Quantum Internet: a step closer

Demonstrations and applications using diamond qubits

Mariagrazia Iuliano

QUANTUM INTERNET: A STEP CLOSER

DEMONSTRATIONS AND APPLICATIONS USING DIAMOND QUBITS

QUANTUM INTERNET: A STEP CLOSER

DEMONSTRATIONS AND APPLICATIONS USING DIAMOND QUBITS

Dissertation

for the purpose of obtaining the degree of doctor
at Delft Univeristy of Technology,
by the authority of the Rector Magnificus, Prof. dr. ir. H. Bijl,
chair of the Board of Doctorates,
to be defended publicly on
Monday 2nd February 2026 at 15:00

by

Mariagrazia IULIANO

Master of Science in Physics,
Sapienza University of Rome, Italy,
born in Benevento, Italy

Printed by: Proefschriftspecialist.nl

Cover design: Mariagrazia Iuliano & Firefly Vector 4

ISBN 978-94-6518-233-9

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

From sprinkler splashes to fireplace ashes
I gave my blood, sweat, and tears for this
I hosted parties and starved my body
Like I'd be saved by a perfect kiss
The jokes weren't funny, I took the money
My friends from home don't know what to say
I looked around in a blood-soaked gown
And I saw something they can't take away
'Cause there were pages turned with the bridges burned
Everything you lose is a step you take
So make the friendship bracelets, take the moment and taste it
You've got no reason to be afraid
You're on your own, kid

You're On Your Own, Kid - Taylor Swift

To all the people who left home to find a new home, on their own.

Contents

Summary

The Quantum Internet is a complementary tool to the widely spread classical Internet, which has already revolutionized our everyday life. The promise is that the Quantum Internet will unlock new unprecedented capabilities and applications that span from secure communication, to distributed quantum computation and enhanced quantum sensing. The realization of such a powerful tool is the result of a joint effort among several fields, like computer science, physics, engineering, and materials science, which all rely on the fundamentals of quantum mechanics. The introduction of a new computational unit, the qubit, allows for the creation of superposition and entangled states, and the possibility of measuring such states. On a practical level, we can envision the Quantum Internet as a network of interconnected heterogeneous platforms aimed at solving different tasks, such as the processing of quantum information at the end nodes, and the storing and retrieval of quantum information in between end nodes to bridge long distances. The quantum information routing is governed and optimized by a dedicated software architecture that facilitates the user interface, removing the requirement of knowing the hardware's physical principles for a general user.

In the hardware framework, the Nitrogen-Vacancy center in diamond represents a viable platform as processing end node, thanks to the high quality of its qubits and the capability of generating remote entanglement with other nodes in the network via its optical interface. These properties can be engineered to utilize the NV center as a test-bed for demonstrating crucial steps towards the Quantum Internet final goal.

We first employ a two-node NV quantum network in the laboratory to demonstrate the elementary building-blocks of distributed quantum computation: the generation of a distributed 4-partite Greenberger-Horne-Zeilinger state and the realization of a non-local Controlled-NOT gate between physically separated and non-interacting qubits.

In the long distance scenario, we use the NV center platform to study the photonic interface of solid-state qubits with time-bin qubits compatible with the emission from quantum memory platforms, such as Rubidium gas or Thulium-doped crystals. The interface is benchmarked with a quantum teleportation experiment. Quantum teleportation is the ultimate protocol that enables the transfer of quantum information from one physical point to another. We teleport a photonic time-bin qubit to the communication qubit of the NV center platform, establishing the primary form of communication between heterogeneous platforms in a quantum network.

Finally, the two-node NV network is used as reliable setup to demonstrate the first operating system for quantum network applications, QNodeOS. QNodeOS can schedule and manage quantum network applications in a multitasking fashion. It constitutes a software interface which enables facilitated access for users, boosting the research in quantum network applications and making a first step towards the deployment of such technology into society.

SAMENVATTING

Het kwantuminternet is een complementair hulpmiddel aan het wijdverbreide klassieke internet, dat ons dagelijks leven al ingrijpend heeft veranderd. De belofte is dat het kwantuminternet nieuwe, ongekende mogelijkheden en toepassingen zal ontsluiten, variërend van veilige communicatie tot gedistribueerde kwantumberekeningen en verbeterde kwantumsensoren. De realisatie van een dergelijk krachtig hulpmiddel is het resultaat van een gezamenlijke inspanning van verschillende disciplines, zoals informatica, natuurkunde, techniek en materiaalkunde, die allemaal steunen op de fundamenten van de kwantummechanica. De introductie van een nieuwe computationele eenheid, de qubit, maakt het mogelijk om superpositie- en verstrengelde toestanden te creëren en deze te meten. Op praktisch niveau kunnen we het kwantuminternet beschouwen als een netwerk van onderling verbonden heterogene platforms die gericht zijn op het uitvoeren van verschillende taken, zoals de verwerking van kwantuminformatie in de eindknooppunten en het opslaan en ophalen van kwantuminformatie tussen de eindknooppunten om grote afstanden te overbruggen. De routering van kwantuminformatie wordt aangestuurd en geoptimaliseerd door een toegewijde softwarearchitectuur die de gebruikersinterface vereenvoudigt, waardoor van een algemene gebruiker geen kennis van de fysieke principes van de hardware vereist is.

Binnen het hardwarekader vormt het stikstof-vacantiecentrum (NV) in diamant een geschikt platform als verwerkend eindknooppunt, dankzij de hoge kwaliteit van zijn qubits en het vermogen om via zijn optische interface verstrengeling op afstand te genereren met andere knooppunten in het netwerk. Deze eigenschappen kunnen zodanig worden ontworpen dat het NV-centrum kan worden gebruikt als testbed voor het demonstreren van cruciale stappen richting het uiteindelijke doel van het kwantuminternet.

We maken eerst gebruik van een tweeknoops NV-kwantumnetwerk in het laboratorium om de elementaire bouwstenen van gedistribueerde kwantumberekening te demonstreren: de generatie van een gedistribueerde vierpartijen Greenberger–Horne–Zeilinger-toestand en de realisatie van een niet-lokale Controlled-NOT-gate tussen fysiek gescheiden en niet-interagerende qubits.

In het scenario met lange afstanden gebruiken we het NV-centrumplatform om de fotonische interface van vaste-stofqubits te bestuderen met tijdsbin-qubits die compatibel zijn met de emissie van kwantumgeheugenplatforms, zoals rubidiumgas of met thulium gedoteerde kristallen. Deze interface wordt geëvalueerd met een kwantumteleportatie-experiment. Kwantumteleportatie is het ultieme protocol dat de overdracht van kwantuminformatie van het ene fysieke punt naar het andere mogelijk maakt. We teleporteren een fotonische tijdsbin-qubit naar de communicatiequbit van het NV-centrumplatform, waarmee de primaire vorm van communicatie tussen heterogene platforms in een kwantumnetwerk wordt gerealiseerd.

Ten slotte wordt het tweeknoops NV-netwerk gebruikt als een betrouwbaar opstelling om het eerste besturingssysteem voor kwantumnetwerktoepassingen te demonstreren:

QNodeOS. QNodeOS kan kwantumnetwerktoepassingen plannen en beheren op een multitaskende manier. Het vormt een software-interface die gebruikers een vereenvoudigde toegang biedt, het onderzoek naar kwantumnetwerktoepassingen stimuleert en een eerste stap zet richting de inzet van deze technologie in de samenleving.

Sommario

L'Internet quantistico è uno strumento complementare al largamente diffuso Internet classico, il quale ha rivoluzionato tutti gli aspetti della vita quotidiana. Lo scopo dell'Internet quantistico è quello di introdurre nuove capacità e applicazioni che non hanno eguali. Alcuni esempi sono una comunicazione più sicura, l'uso di risorse computazionali fisicamente distribuite in più luoghi, e una rete avanzata di sensori con sensibilità aumentata grazie ai principi della fisica quantistica. La realizzazione dell'Internet quantistico riguarda diversi campi, come l'informatica, la fisica, l'ingegneria e la scienza dei materiali. Sul piano pratico, questo si traduce in diverse piattaforme hardware, tutte connesse tra loro in una rete, dove un'architettura software dedicata facilita l'accesso ad utenti meno esperti sui principi fisici che governano l'hardware e si occupa di gestire il traffico dell'informazione. Le piattaforme hardware devono realizzare diversi task, come la computazione quantistica o memorizzare informazione quantistica e rilasciarla al momento opportuno, specialmente per il raggiungimento di grandi distanze tra i nodi della rete. Alla base di tutto ciò troviamo i principi cardine della fisica quantistica e l'introduzione di una nuova unità di calcolo, il qubit, che permette di creare stati di sovrapposizione ed entangled, e di effettuare misure sullo stato stesso.
Tra le piattaforme hardware, i centri azoto-vacanza (NV) nel diamante rappresentano una delle piattaforme più promettenti, grazie alle eccellenti proprietà dei qubit e alla possibilità di generare entanglement a distanza con altri nodi della rete attraverso la loro interfaccia con i fotoni. Queste proprietà possono essere ingegnerizzate per usare gli NV come un setup affidabile per le dimostrazioni nell'ambito dell'Internet quantistico.
Come prima dimostrazione, utilizziamo una rete quantistica basata sugli NV, a due nodi in laboratorio per creare uno stato Greenberger-Horne-Zeilinger a quattro qubit distribuiti (due qubit per nodo) e per realizzare un gate Controlled-NOT a distanza, tra due qubit che non possono interagire tra loro e si trovano fisicamente in due luoghi separati. Queste dimostrazioni rappresentano le risorse primarie per la computazione quantistica distribuita su più device, necessaria per raggiungere un vantaggio sulla computazione classica e facilitare la costruzione di computer quantistici, che possono, dunque, essere di dimensioni più piccole.
Un'altra dimostrazione si focalizza sul raggiungimento di grandi distanze tra le piattaforme dell'Internet quantistico. In questo caso studiamo l'interfaccia fotonica tra gli NV e qubit basati sui fotoni che sono compatibili con l'emissione da parte di piattaforme per memorie quantistiche. Alcuni esempi per memorie quantistiche sono i gas di atomi di rubidio o cristalli drogati con ioni di tulio, una terra rara. L'interfaccia è collaudata con un esperimento di teletrasporto quantistico, il protocollo fondamentale per il trasferimento di informazione quantistica da un punto all'altro di una rete. L'esperimento di teletrasporto trasferisce l'informazione dal qubit fotonico nel qubit dell'NV.
L'ultima dimostrazione utilizza la rete NV a due nodi come base hardware per testare il primo sistema operativo per reti quantistiche, QNodeOS. QNodeOS può organizzare e

programmare applicazioni per reti quantistiche in maniera multitasking. Rappresenta un’interfaccia software che facilita l’accesso a queste tecnologie per un utente medio, dando spazio alla ricerca di nuove applicazioni per queste reti quantistiche e costituisce un primo passo per la diffusione su larga scala dell’Internet quantistico.

1 Introduction

I'm so sick of running as fast as I can
Wondering if I'd get there quicker if I was a man

The Man - Taylor Swift

Curiosity about nature motivates humanity to investigate, model, and reproduce natural phenomena. As scientific questions grow in complexity, so does the need for computational resources capable of capturing nature mechanisms and explanations. Classical computers struggle with such tasks because simulating quantum systems generally requires resources that grow exponentially with system size. This limitation has fueled a long-standing pursuit of more powerful computational paradigms.
Another big driver is the desire to communicate. It reflects a fundamental human need to exchange knowledge and information, and preserve them across time. Communication in all its forms is not always intended for everybody to read, see or hear, tightly bounding the progress of communication to the necessity for secrecy and privacy. This gives rise to cryptography. The term originates from the ancient Greek $\kappa\rho\upsilon\pi\tau$ὸ$\varsigma$ ("hidden") and
$\gamma\rho\alpha\phi\epsilon\iota\nu$ ("to write"). Documented examples of cryptography date back to the Roman Empire [1], while its systematic study accelerated with the development of modern cryptologic methods in the 19th and 20th centuries. Today, cryptography is anchored in mathematical complexity theory: many widely used protocols rely on the presumed hardness of problems, such as factoring [2], and push advancement in several complementary fields, including physics, engineering and computer science.
Both these drivers converge in the rise of the Internet, that dramatically expanded the scale and speed of communication, enabling global information exchange and the outsourcing of computational workloads to distributed cloud infrastructures [3, 4]. However, this growth has introduced significant vulnerabilities. Critical sectors, such as finance, defense and politics, depend on secure communication, yet classical cryptographic security is not unconditional. It rests on computational assumptions that may fail if adversaries gain sufficient processing power or discover new algorithms. Moreover, as hardware scales down, classical computing approaches physical and thermodynamic limits.
It is in this context that quantum mechanics introduces a paradigm shift. As argued by R. Feynman and D. Deutsch, classical computers are fundamentally inefficient for simulating quantum systems, whereas a quantum computer, governed by quantum physical laws, can efficiently perform such tasks [5, 6]. This insight led to the emergence of quantum algorithms, such as Shor's factoring algorithm [7] and Grover's search algorithm [8], which threaten classical cryptographic protocols and simultaneously offer computational speedups for scientific applications.

Building on these developments, the Quantum Internet has emerged as a complementary infrastructure to the classical Internet. It leverages quantum superposition, entanglement, and quantum teleportation [9] to distribute quantum information across networks. In such networks, communication is mediated by qubits, a new computational unit counterpart of the classical bit. We can distinguish two species of qubits [10]: the so-called *stationary qubit* and the *flying qubit*. The *stationary qubit* is a qubit that can store quantum information in a fixed location, process and manipulate such information, when required, via quantum gates, and finally readout the state with high fidelity. The *flying qubit* has the task of transporting quantum information between remote stationary qubits by exploiting entanglement with stationary qubits. An example

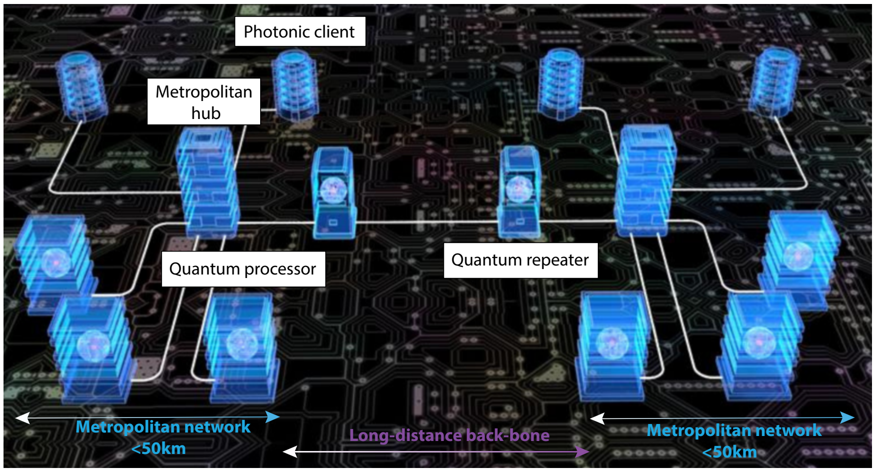


Figure 1.1: **The Quantum Internet: hardware visualization** Each metropolitan network, which expands up to ~50km, is connected via a backbone of quantum repeaters, being able to cover distances of hundreds of kilometers. Adapted from Quantum Internet Alliance website.

of a *flying qubit* is a single photon that can be transmitted via optical fibers or in free-space.
Prominent applications include unconditionally secure communication via quantum key distribution (QKD) [11, 12], distributed quantum computing assisted by entanglement [13], and quantum-enhanced sensor networks [14]. However, achieving such a network requires overcoming fundamental challenges. Quantum states are fragile and easily disturbed by noise, decoherence, and photon loss in transmission channels. Therefore, the Quantum Internet must rely on a heterogeneous hardware ecosystem, including [15]:

- **Quantum processors**. End-nodes must support coherent multi-qubit registers, high-fidelity control operations and the capability to generate remote entanglement with other quantum platforms in the network utilizing optical interfaces. Leading candidates include trapped ions [16], neutral atoms [17], color centers in diamond [18], and integrated photonic qubits [19, 20]. Each platform currently presents trade-offs in scalability, experimental rates, coherence times, and integration.

- **Quantum repeaters**. Due to exponential photon loss in fibers or high information traffic and temporary unavailability of the receiving end-nodes, direct long-distance quantum communication is not feasible beyond a few hundred kilometers [21, 22]. Quantum repeaters provide entanglement storage, entanglement swapping, and purification, enabling multi-hop communication [23, 24]. Achieving long-coherence, high-fidelity quantum memories still remains a central challenge. Examples of quantum repeater platforms are rare-earth ions-doped crystals [25, 26] or atom gases [27–30].

- **Photonic clients**. Lightweight devices capable of generating, measuring, or receiving photonic qubits allow users to access quantum network functionalities without owning full-scale quantum processors. Protocols for remote state preparation, delegated or blind quantum computation [31–36] make quantum resources accessible, enabling scalable, secure, and widespread adoption.

- **Metropolitan (quantum) hubs.** In classical communication, a hub is a hardware device whose task is to connect multiple nodes in the network at the physical layer. Similarly, a metropolitan hub in a quantum network serves as an aggregation node that integrates heterogeneous devices, both classical and quantum, within a city-scale quantum network (up to ~50km). These hubs act as switching and coordination centers, providing routing, entanglement management, and resource allocation for multiple users and applications within a metropolitan area. They can typically host several key capabilities [37, 38]: multiplexed quantum memories enabling simultaneous entanglement distribution, quantum-classical control systems ensuring synchronization and time-stamping across city-scale fiber infrastructures, and photon detection devices.

Hardware alone cannot realize a functional Quantum Internet. Quantum communication introduces different constraints from classical networking: entanglement must be established and consumed on demand, routing must account for decoherence, and many protocols require tight temporal coordination. As a result, dedicated quantum network control stacks and software layers [39–41] are essential for managing entanglement distribution, scheduling, error handling, and hybrid classical–quantum communication. Ultimately, the software architecture enables abstractions of the physical layers, a necessary step to bring the technology to society, allowing hardware non-experts to contribute to the development of novel quantum network applications. Each software layer has a specific functionality and it is envisioned such that it can work independently from the layers right below. As proposed by A. Dahlberg et al. [39], a possible configuration, inspired by classical TCP/IP protocols, includes, from bottom to top:

- a **physical layer**: the hardware that has quantum capabilities, such as remote entanglement generation and local quantum information processing, but it needs to be instructed on when to execute these capabilities;

- a **link layer**: an entanglement service that ensures that the physical layer generates robust entanglement on demand. It does not require information on how the entanglement instruction is physically executed at the control level.

- a **network layer**: responsible to distribute entanglement across a network between non-neighboring nodes;

- a **transport layer**: responsible for deterministically transferring the qubit information via a teleportation protocol, utilizing a previously-generated entangled state and subsequent feed-forwarded instructions.

In this thesis, the aim is to provide a series of proof-of-concept demonstrations within the emerging landscape of the Quantum Internet by leveraging a quantum network based on nitrogen-vacancy (NV) centers in diamond. Among the many physical platforms available for quantum information processing, NV centers have distinguished themselves as a particularly powerful and versatile testbed for networked quantum technologies. Their unique combination of properties, like an optically accessible spin [42], a robust photonic interface enabling remote entanglement generation [43], exceptionally long coherence times [44], and the possibility of creating larger qubit registers by controlling the surrounding nuclear spins [45, 46], makes them ideally suited for studying interesting applications and interfaces. These advantages have translated into several landmark experimental achievements. For example, quantum teleportation between non-neighboring quantum network nodes has been realized [47], posing groundwork for long-range quantum communication mediated by entanglement swapping. More recently, entanglement has been distributed at metropolitan distances using deployed fiber-based links connected to NV processors [48], illustrating the suitability of such platforms for real-world quantum networking scenarios. Together, these results establish NV centers not only as a convenient laboratory system, whose results mark a path toward deployment into the world. Against this backdrop, the present thesis explores several core building blocks required for scalable quantum networking, with NV hardware serving as a test-bed. The structure of the thesis is as follows:

- **Chapter 2** introduces the NV-center platform in detail, reviewing its physical properties, photonic interface, control techniques, and the role of its local memory qubits within quantum network architectures.
- **Chapter 3** presents an experimental demonstration for the generation of distributed 4-partite entangled states in a network and the realization of a quantum gate teleportation protocol. Such demonstrations form the foundation of more advanced distributed quantum computing protocols.
- **Chapter 4** extends the focus to heterogeneous quantum networks by investigating the photonic interface between an NV center end-node and photonic time-bin qubits compatible with emission from quantum repeater platforms, such as Thulium-doped crystals or Rubidium gas. This aims to address the challenge of interconnecting disparate quantum hardware into a unified and scalable network.
- **Chapter 5** introduces the world's first quantum network operating system, developed to manage, schedule, and orchestrate quantum network operations. The NV-center platform enables its benchmarking through controlled, high-fidelity experiments, providing insight into the co-design of hardware and software necessary for scalable quantum networking applications.
- **Chapter 6** summarizes the key findings and outlines a perspective on future developments in this rapidly evolving research field.

REFERENCES

[1] C. S. Tranquillus. *De vita Caesarum.* 138CE.

[2] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of the ACM* 21.2 (Feb. 1978), pp. 120–126. DOI: 10.1145/359340.359342.

[3] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff. “A brief history of the internet”. In: *ACM SIGCOMM Computer Communication Review* 39.5 (Oct. 2009), pp. 22–31. DOI: 10.1145/1629607.1629613.

[4] S. Lukasik. “Why the Arpanet Was Built”. In: *IEEE Annals of the History of Computing* 33.3 (Mar. 2011), pp. 4–21. DOI: 10.1109/MAHC.2010.11.

[5] R. P. Feynman. “Simulating physics with computers”. In: (1982).

[6] D. Deutsch. “Quantum theory, the Church–Turing principle and the universal quantum computer”. In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 400.1818 (July 1985), pp. 97–117. DOI: 10.1098/rspa.1985.0070.

[7] P. W. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *FOCS.* IEEE, 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[8] L. K. Grover. “A Fast Quantum Mechanical Algorithm for Database Search”. In: *STOC.* ACM, 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[9] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. “Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels”. In: *Phys. Rev. Lett.* 70 (13 Mar. 1993), pp. 1895–1899. DOI: 10.1103/PhysRevLett.70.1895.

[10] D. P. DiVincenzo. “The Physical Implementation of Quantum Computation”. In: *Fortschritte der Physik* 48.9-11 (Sept. 2000), pp. 771–783. DOI: 10.1002/1521-3978(200009)48:9/11<771::AID-PROP771>3.0.CO;2-E.

[11] C. H. Bennett and G. Brassard. “Quantum Cryptography: Public Key distribution and Coin Tossing”. In: *Theor. Comput. Sci.* 560.1 (2014), pp. 7–11. DOI: 10.1016/j.tcs.2014.05.025.

[12] A. K. Ekert. “Quantum cryptography based on Bell’s theorem”. In: *Physical Review Letters* 67.6 (Aug. 1991), pp. 661–663. DOI: 10.1103/PhysRevLett.67.661.

[13] J. I. Cirac, A. K. Ekert, S. F. Huelga, and C. Macchiavello. “Distributed quantum computation over noisy channels”. In: *Physical Review A* 59.6 (June 1999). Publisher: American Physical Society, pp. 4249–4254. DOI: 10.1103/PhysRevA.59.4249.

[14] P. Kómár, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin. “A quantum network of clocks”. In: *Nature Physics* 10.8 (Aug. 2014), pp. 582–587. DOI: 10.1038/nphys3000.

[15] S. Wehner, D. Elkouss, and R. Hanson. “Quantum internet: A vision for the road ahead”. In: *Science* 362.6412 (Oct. 2018). DOI: 10.1126/science.aam9288.

[16] P. Drmota, D. Main, D. Nadlinger, B. Nichol, M. Weber, E. Ainley, A. Agrawal, R. Srinivas, G. Araneda, C. Ballance, et al. "Robust quantum memory in a trapped-ion quantum network node". In: *Physical Review Letters* 130.9 (2023). Publisher: APS, p. 090803. DOI: 10.1103/PhysRevLett.130.090803.

[17] J. P. Covey, H. Weinfurter, and H. Bernien. "Quantum networks with neutral atom processing nodes". In: *npj Quantum Information* 9.1 (Sept. 2023), pp. 1–12. DOI: 10.1038/s41534-023-00759-9.

[18] M. Ruf, N. H. Wan, H. Choi, D. Englund, and R. Hanson. "Quantum networks based on color centers in diamond". In: *Journal of Applied Physics* 130.7 (Aug. 2021), p. 070901. DOI: 10.1063/5.0056534.

[19] J. Wang, F. Sciarrino, A. Laing, and M. G. Thompson. "Integrated photonic quantum technologies". In: *Nature Photonics* 14.5 (May 2020), pp. 273–284. DOI: 10.1038/s41566-019-0532-1.

[20] H. Wang, T. C. Ralph, J. J. Renema, C.-Y. Lu, and J.-W. Pan. "Scalable photonic quantum technologies". In: *Nature Materials* (Aug. 2025). DOI: 10.1038/s41563-025-02306-7.

[21] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller. "Quantum Repeaters: The Role of Imperfect Local Operations in Quantum Communication". In: *Physical Review Letters* 81.26 (Dec. 1998), pp. 5932–5935. DOI: 10.1103/PhysRevLett.81.5932.

[22] W. Dür, H.-J. Briegel, J. I. Cirac, and P. Zoller. "Quantum repeaters based on entanglement purification". In: *Physical Review A* 59.1 (Jan. 1999), pp. 169–181. DOI: 10.1103/PhysRevA.59.169.

[23] K. Azuma, S. E. Economou, D. Elkouss, P. Hilaire, L. Jiang, H.-K. Lo, and I. Tzitrin. "Quantum repeaters: From quantum networks to the quantum internet". In: *Reviews of Modern Physics* 95.4 (Dec. 2023), p. 045006. DOI: 10.1103/RevModPhys.95.045006.

[24] C. Liorni, H. Kampermann, and D. Bruß. "Quantum repeaters in space". In: *New Journal of Physics* 23.5 (May 2021), p. 053021. DOI: 10.1088/1367-2630/abfa63.

[25] D. Lago-Rivera, J. V. Rakonjac, S. Grandi, and H. d. Riedmatten. "Long distance multiplexed quantum teleportation from a telecom photon to a solid-state qubit". In: *Nature Communications* 14.1 (Apr. 2023), p. 1889. DOI: 10.1038/s41467-023-37518-5.

[26] W. Tittel, M. Afzelius, A. Kinos, L. Rippe, and A. Walther. "Quantum networks using rare-earth ions". In: *Quantum Science and Technology* 10.3 (Oct. 2025), p. 033002. DOI: 10.1088/2058-9565/addd93.

[27] C. Simon, H. De Riedmatten, and M. Afzelius. "Temporally multiplexed quantum repeaters with atomic gases". In: *Physical Review A* 82.1 (July 2010), p. 010304. DOI: 10.1103/PhysRevA.82.010304.

[28] N. Sangouard, C. Simon, H. de Riedmatten, and N. Gisin. "Quantum repeaters based on atomic ensembles and linear optics". In: *Reviews of Modern Physics* 83.1 (Mar. 2011), pp. 33–80. DOI: 10.1103/RevModPhys.83.33.

[29] X.-H. Bao, X.-F. Xu, C.-M. Li, Z.-S. Yuan, C.-Y. Lu, and J.-W. Pan. “Quantum teleportation between remote atomic-ensemble quantum memories”. In: *Proceedings of the National Academy of Sciences* 109.50 (Dec. 2012), pp. 20347–20351. DOI: 10.1073/pnas.1207329109.

[30] Y. Wang, A. N. Craddock, R. Sekelsky, M. Flament, and M. Namazi. “Field-Deployable Quantum Memory for Quantum Networking”. In: *Physical Review Applied* 18.4 (Oct. 2022), p. 044058. DOI: 10.1103/PhysRevApplied.18.044058.

[31] C. H. Bennett, D. P. DiVincenzo, P. W. Shor, J. A. Smolin, B. M. Terhal, and W. K. Wootters. “Remote State Preparation”. In: *Physical Review Letters* 87.7 (July 2001), p. 077902. DOI: 10.1103/PhysRevLett.87.077902.

[32] A. Broadbent, J. Fitzsimons, and E. Kashefi. “Universal Blind Quantum Computation”. In: *FOCS*. IEEE, 2009, pp. 517–526. DOI: 10.1109/FOCS.2009.36.

[33] T. Morimae and K. Fujii. “Blind quantum computation protocol in which Alice only makes measurements”. In: *Physical Review A* 87.5 (May 2013), p. 050301. DOI: 10.1103/PhysRevA.87.050301.

[34] C. Greganti, M.-C. Roehsner, S. Barz, T. Morimae, and P. Walther. “Demonstration of measurement-only blind quantum computing”. In: *New Journal of Physics* 18.1 (Jan. 2016), p. 013020. DOI: 10.1088/1367-2630/18/1/013020.

[35] Q. Li, C. Liu, Y. Peng, F. Yu, and C. Zhang. “Blind quantum computation where a user only performs single-qubit gates”. In: *Optics & Laser Technology* 142 (Oct. 2021), p. 107190. DOI: 10.1016/j.optlastec.2021.107190.

[36] J. Van Dam, G. Avis, T. B. Propp, F. Ferreira Da Silva, J. A. Slater, T. E. Northup, and S. Wehner. “Hardware requirements for trapped-ion-based verifiable blind quantum computing with a measurement-only client”. In: *Quantum Science and Technology* 9.4 (Oct. 2024), p. 045031. DOI: 10.1088/2058-9565/ad6eb2.

[37] G. Avis, F. Rozpędek, and S. Wehner. “Analysis of multipartite entanglement distribution using a central quantum-network node”. In: *Physical Review A* 107.1 (Jan. 2023), p. 012609. DOI: 10.1103/PhysRevA.107.012609.

[38] S. Gauthier, G. Vardoyan, and S. Wehner. “A Control Architecture for Entanglement Generation Switches in Quantum Networks”. In: *Proceedings of the 1st Workshop on Quantum Networks and Distributed Quantum Computing*. New York NY USA: ACM, Sept. 2023, pp. 38–44. DOI: 10.1145/3610251.3610552.

[39] A. Dahlberg et al. “A Link Layer Protocol for Quantum Networks”. In: *SIGCOMM*. ACM, 2019, pp. 159–173. DOI: 10.1145/3341302.3342070.

[40] W. Kozlowski, A. Dahlberg, and S. Wehner. “Designing a quantum network protocol”. In: *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*. Barcelona Spain: ACM, Nov. 2020, pp. 1–16. DOI: 10.1145/3386367.3431293.

[41] M. Pompili et al. “Experimental Demonstration of Entanglement Delivery Using a Quantum Network Stack”. In: *npj Quantum Information* 8.1 (2022), p. 121. DOI: 10.1038/s41534-022-00631-2.

[42] L. Robledo, H. Bernien, I. van Weperen, and R. Hanson. “Control and Coherence of the Optical Transition of Single Nitrogen Vacancy Centers in Diamond”. In: *Physical Review Letters* 105.17 (Oct. 2010), p. 177403. DOI: 10.1103/PhysRevLett.105.177403.

[43] H. Bernien et al. “Heralded Entanglement Between Solid-State Qubits Separated by Three Metres”. In: *Nature* 497 (2013), pp. 86–90. DOI: 10.1038/nature12016.

[44] M. H. Abobeih, J. Cramer, M. A. Bakker, N. Kalb, M. Markham, D. J. Twitchen, and T. H. Taminiau. “One-Second Coherence for a Single Electron Spin Coupled to a Multi-Qubit Nuclear-Spin Environment”. In: *Nature Commun.* 9.1 (2018), pp. 1–8. DOI: 10.1038/s41467-018-04916-z.

[45] C. E. Bradley, J. Randall, M. H. Abobeih, R. C. Berrevoets, M. J. Degen, M. A. Bakker, M. Markham, D. J. Twitchen, and T. H. Taminiau. “A Ten-Qubit Solid-State Spin Register with Quantum Memory up to One Minute”. In: *Phys. Rev. X* 9.3 (2019), pp. 031045-1–031045-12. DOI: 10.1103/PhysRevX.9.031045.

[46] G. L. Van De Stolpe, D. P. Kwiatkowski, C. E. Bradley, J. Randall, M. H. Abobeih, S. A. Breitweiser, L. C. Bassett, M. Markham, D. J. Twitchen, and T. H. Taminiau. “Mapping a 50-spin-qubit network through correlated sensing”. In: *Nature Communications* 15.1 (Mar. 2024), p. 2006. DOI: 10.1038/s41467-024-46075-4.

[47] S. L. N. Hermans, M. Pompili, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson. “Qubit teleportation between non-neighbouring nodes in a quantum network”. In: *Nature* 605.7911 (May 2022), pp. 663–668. DOI: 10.1038/s41586-022-04697-y.

[48] A. J. Stolk et al. “Metropolitan-scale heralded entanglement of solid-state qubits”. In: *Science Advances* 10.44 (Nov. 2024), eadp6442. DOI: 10.1126/sciadv.adp6442.

2

METHODS

In this chapter, an overview of the platform and its control methods is given, in perspective of the demonstrations that are discussed throughout this thesis. First of all, we introduce the NV center as a color defect in diamond and explain its properties from a solid-state physics background. Then, such properties can be engineered and controlled for the realization of quantum network applications.

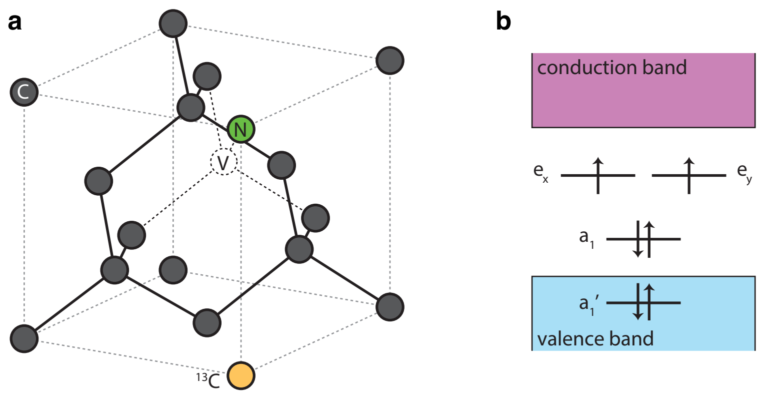


Figure 2.1: **The NV center in diamond** a) Lattice unit of the diamond containing the NV point defect. The orange atom in the lattice represents the Carbon-13 isotope, which can be used as an additional data qubit. b) Molecular orbitals of NV$^-$ obtained from the overlap of the nitrogen and carbon atomic orbitals. Adapted from Pfaff, Bernien and Kalb [4–6]

.

2.1. THE NV CENTER IN DIAMOND

The Nitrogen-Vacancy center in diamond is a point defect in the crystalline structure of the diamond, where two neighboring carbon (C) atoms are replaced by a nitrogen (N) atom and a vacancy. The first optical evidence of this defect dates back to the 60s and 70s of the last century [1, 2], but it is only in the late 90s that the spin properties were discovered [3] and made this defect interesting for the quantum information field, given its optically-active electronic spin.
The NV can occur in two charge forms: NV^0 (neutral) and NV$^-$ (negative). In the neutral case, there are five outbound electrons, where two are obtained from the nitrogen atom and three correspond to unpaired electrons from the neighboring carbon atoms, as visualized in Fig. 2.1a. For the negatively charged state, there is an additional electron captured from the environment. Throughout this thesis, the actual qubit platform is based on NV$^-$, which will be referred to as NV. However, the neutral form is interesting to consider to understand the charge trapping mechanism, as explained in the following sections.

2.2. THE NV$^-$ ELECTRONIC STRUCTURE

The molecular orbitals of the NV are obtained via a linear combination of the involved atomic orbitals according to the LCAO theory and the C_{3v} symmetry group of the defect [7–9]. The carbon atomic orbitals are sp^3 hybridized, and the combination with the nitrogen atomic orbital results in the molecular orbitals displayed in Fig. 2.1b: $a_i' a_i e$. Considering the Pauli exclusion principle, the occupation of such orbitals becomes: $a_i'^2 a_i^2 e^2$, with a total spin number of 1. For the excitation, one a_1 electron can be promoted in one of the two degenerate e states, changing the occupation configuration in $a_i'^2 a_i^1 e^3$. Given that the unpaired electrons, both in the ground state and in the excited state, are in the

diamond band-gap, the NV can be seen as an atom-like system, despite being hosted in a solid-state material.
Additionally, we have to also consider the Coulomb repulsion between the electrons and the electrons and nuclei. This results in non-degenerate singlet and triplet ground and excited states, as depicted in Fig. 2.2a. Transitions between the triplet ground and excited states can occur at an energy of 1.945eV, corresponding to the absorption and spontaneous emission of single photons (in the Zero-Phonon Line) at ~637nm. However, only 3% of the emitted light from the NV is a single photon. In the remaining 97%, the emission is composed by a photon and a phonon, constituting the Phonon-Side Band. It is also possible to excite the emitter off-resonantly, illuminating the defect with a high-energy laser field, like a green laser at 535nm. For the spin states m_s=±1 of the triplet, it is also possible to observe decays via the singlet state, emitting photons at 1042nm. Cool-

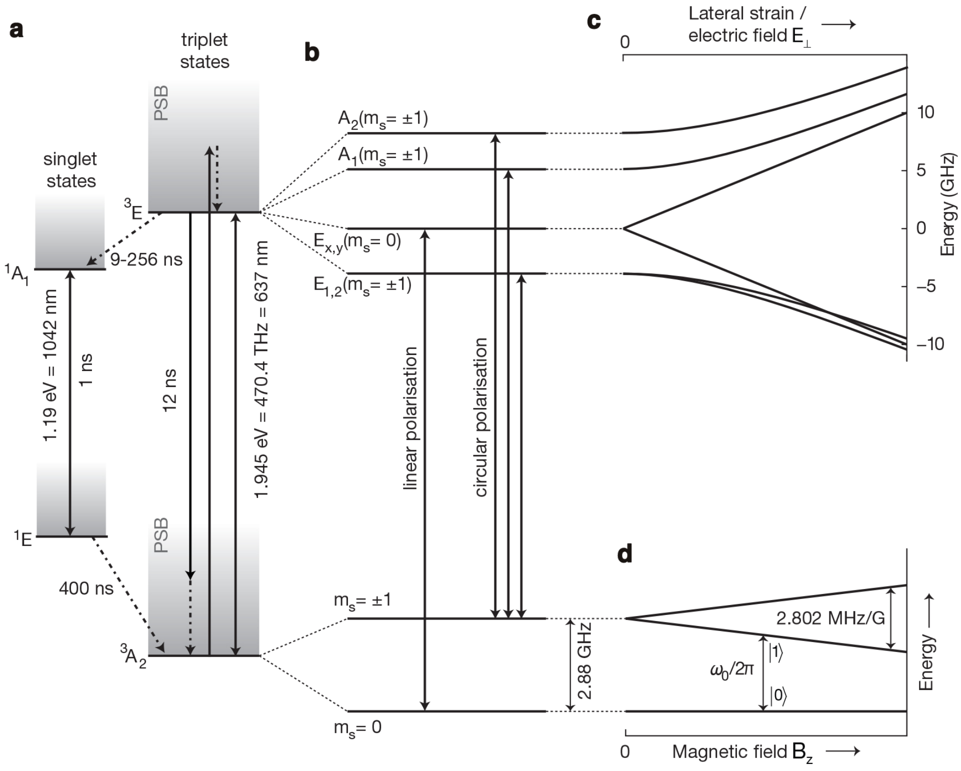


Figure 2.2: **Energy levels of the NV at cryogenic temperatures** a) The NV can form triplet and singlet states, with the triplet states showing lower energy. Excitation from the triplet ground state can be done resonantly or off-resonantly. The emission occurs via the Phonon-Side Band, in which a phonon and a photon are emitted, or via the Zero-Phonon Line, in which only single photon at 637nm are emitted. There is also a non-zero probability of decaying from the triplet excited states via the singlet states, called intersystem crossing. b) At cryogenic temperatures, the triplet ground and excited states present a well-defined structure and spin-selective transitions. c) The excited states become non-degenerate with lateral strain and external electric field. d) For the ground state, the degeneracy is lifted when an external magnetic field is applied (Zeeman effect). Adapted from Kalb [6]

ing down the NV center to cryogenic temperatures (~4K), the fine structure of its electronic levels can be observed (Fig. 2.2b). For the triplet states, including the spin-spin interaction and the spin-orbit interaction, we can describe the electron ground states via the Hamiltonian

$$H_{GS}/h = D_{gs} S_z^2 + \gamma_e S \cdot B \tag{2.1}$$

where D_{gs} ≃2.88GHz is the zero-field spitting between m_s=0 and m_s=±1 and S_i is the spin-1 matrix along the axis i. Additionally, the degeneracy of the spin states m_s=±1 can be lifted via the Zeeman effect with a gyromagnetic factor γ_e=2.802MHz/G (Fig. 2.2d) when an external magnetic field B is applied. In this way, it is possible to obtain three distinct ground state levels, in which two of them can be used to encode the qubit space ($|0\rangle \equiv$m$_s$=0 and $|1\rangle \equiv$m$_s$=-1 or m$_s$=+1). The choice on the physical $|1\rangle$ state is based on experimental considerations, for instance on the frequency of the microwave field necessary to manipulate the qubit, which we tend to confine in the range around 2-4GHz.
For the excited states, the spin-spin interaction results in four distinct energy levels that are accessible from the ground state via spin-selection rules (Fig. 2.2b). The interaction due to lateral strain and additional external electric field (Stark effect) lifts the degeneracy for the $E_{x,y}$ and $E_{1,2}$ states (Fig. 2.2c), obtaining a total of six distinct excited states together with the possibility of transition-tuning, which will become crucial for quantum network experiments [10–12].

2.3. NV DEVICE AND CONTROL

2.3.1. SAMPLE PREPARATION

Although sample fabrication is beyond the scope of this dissertation, a brief description of the NV devices and their fabrication process is useful to justify certain choices in the setup design and control.
The NV center can be naturally found in low concentrations in diamonds that have gone through the chemical vapor deposition (CVD) growth process. For the devices used in this thesis, no additional processes regarding the manipulation of the NV concentration were pursued. The diamonds are type-IIa, namely they have the lowest available concentration of nitrogen defects, and they are cut along the ⟨111⟩ crystal orientation. After localizing the position of the NVs in the bulk diamond sample using a green laser at room temperature, a solid-immersion lens (SIL) is milled around each NV via a focused ion beam (Fig. 2.3a-b). The SIL improves the collection efficiency of the NV photons, reducing the reflection at the diamond-air interface [13, 14]. An additional deposition of anti-reflection coating (Al_2O_3) on the diamond surface further reduces the reflections. The NVs used for the experiments can typically be found at a depth of ~10μm with respect to the SIL surface. Gold strip-lines for the conduction of radiofrequency waves, as well as the application of DC fields at the NV, are then fabricated around the SILs via a lithography process. Ultimately, the sample is mounted on a printed circuit board (PCB) and bondwires connect gold pads on the diamond to the PCB (Fig. 2.3c).

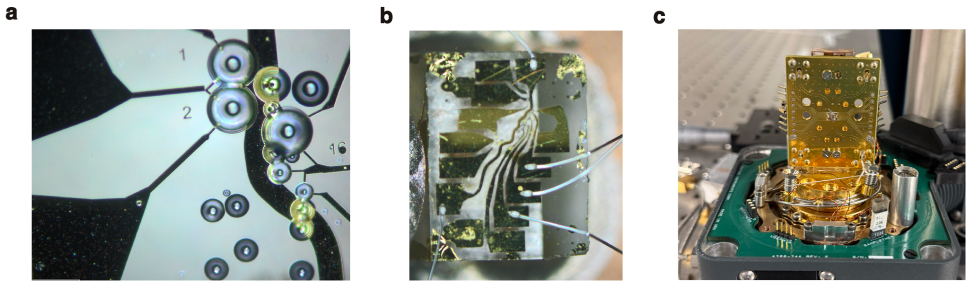


Figure 2.3: **A NV center-based device** a) Microscope image of SILs fabricated around single NV defects. The dark parts represent the gold striplines and gate voltages used for microwave and DC delivery, respectively. b) Microscope image of samples with bondwires to the PCB c) Picture of an NV sample mounted on a PCB.

2.3.2. OPTICAL CONTROL

The NV has an optically active spin that can be controlled. In this section, all-optical control techniques that are relevant for quantum information processing and quantum networks are discussed.

CHARGE-RESONANCE CHECK

At the basis of any NV experiment in the scope of this thesis, there is the so-called Charge-Resonance check [15]. It is crucial that the NV is in its right charge state and that the laser fields are on resonance with the addressed transitions, generating the need for a control and validation procedure.

For the addressed transition, a choice has to be made beforehand on which excited state (E_x or E_y) is addressed from the ground state m_s=0. For quantum network experiments, this choice is made such that the resonance frequency for this transition is tunable (or even equal) to the resonance frequency of the node we want to generate entanglement with[1].

Regarding the charge state, during an experiment, the use of laser light can induce an ionization process [16], which converts NV^- into NV^0 via a two-photon absorption process. In addition, the charge environment around the NV can fluctuate, causing spectral diffusion. This means that the frequencies of the addressed transitions might change during the experiment, constituting a criticality for quantum network experiments, which rely on photon indistinguishability, as it will be explained in Section 2.4.

The Charge-Resonance check has therefore the duty of counteracting to these phenomena. To probe the overlap of the laser fields with the desired transitions, weak (1-10nW) long (50μs) pulses on resonance with the m_s=0→E_x/E_y and $m_s = \pm1 \rightarrow E_{1,2}$ are sent, and the number of emitted photons in the PSB during the illumination is recorded on an avalanche photodiode. If the number of photons is above a certain pass threshold, the NV is in the right charge-resonance state. If the counts are, instead, below the threshold, there are two possible scenarios. When the counts are between the threshold and the

[1]In the case of long-distance entanglement (several km), generally frequency conversion to telecom wavelengths is preferable due to the high loss in optical fibers in the visible range. Therefore, a tuning mechanism at the single emitters is not necessary, as the pump lasers for the frequency conversion are tuned to generate converted photons at the same frequency.

repump threshold, the resonant illumination is repeated until the threshold is passed. If the counts are even below the repump threshold, this is an indication that the NV is not in the right charge state anymore or very far from the ideal resonance. To restore NV^-, it is necessary to ionize back from NV^0 (this process is called repumping). One solution is to use off-resonant excitation via a green laser at ~515nm at several μW of power. Besides restoring the negative charge state, the green laser also reshuffles the charge environment, which can help to re-establish the optimal resonance frequencies by exploiting spectral diffusion. However, it is not always possible to use green light for the repumping. As stated before, one possibility for tuning the resonance frequency of the NV is via the DC Stark effect [10]. When a DC field is applied, the use of the green laser can cause a build-up of charges inside the diamond, which could ultimately end up in damage to the emitter itself. Hence, another repumping mechanism is needed. A two-photon process addressing resonantly the ZPL transition of NV^0 can be used [17]. The ZPL transition of NV^0 is at ~575nm, corresponding to a laser field of yellow color. This method deterministically recharges the NV, when proper duration and power of the yellow laser is used, and does not cause spectral diffusion, as it happens when using the green laser, instead. A summary of the control logic for the CR check is reported in Figure 2.4a,b. As stated previously, an experiment can start only when the CR check is passed. Analogously, a round of CR check performed after the experiment can validate the experiment itself. A low or no counts in the CR check round right after the experiment can indicate that the emitter was not at the right configuration during the experiment, hence a filtering on the results can be introduced.

INITIALIZATION

Given the spin-selective nature of the NV optical transitions, they can be exploited to achieve high-fidelity initialization. In this regard, we use both ground states m_s=±1, paired with the excited states $E_{1,2}$. At each cycle, these transitions have a 40% probability of decaying via the singlet states (inter-system crossing). When decaying back to the triplet ground states, there is a preferred decay ending up in the m_s=0 state. In this way, it is possible to achieve initialization probabilities of more than 99.7%, when using a strong short resonant pulse (~ μW for ~ μs), typically during an entanglement attempt. Alternatively, when time constraints are not so strict, the initialization can be achieved via weak long pulses (~nW for 100s μs), which reduce the ionization probability. In this dissertation, the second option becomes relevant when additional qubits are introduced in the experiment. We usually refer to this process also as "reset" or "spin pumping" and the transition $m_s=\pm1 \rightarrow E_{1,2}$ is also called "spin-pump" transition.

READ-OUT

Another crucial element for quantum information processing is the ability to measure the quantum state. For the NV center, this can be achieved optically by using, once again, the spin-selective transitions [19, 20]. Particularly, we address one of the two transitions (with few nW for tens of μs) $m_s=0 \rightarrow E_x/E_y$, whose choice depends on the possibility to match the other node, in case of network experiments, or on the cyclicity of the transition. Such a transition is also referred to as "read-out transition", however it is also used for entanglement generation. The cyclicity of the transition determines the read-out fidelity, and given the high cyclicity of these transitions it is possible to perform the

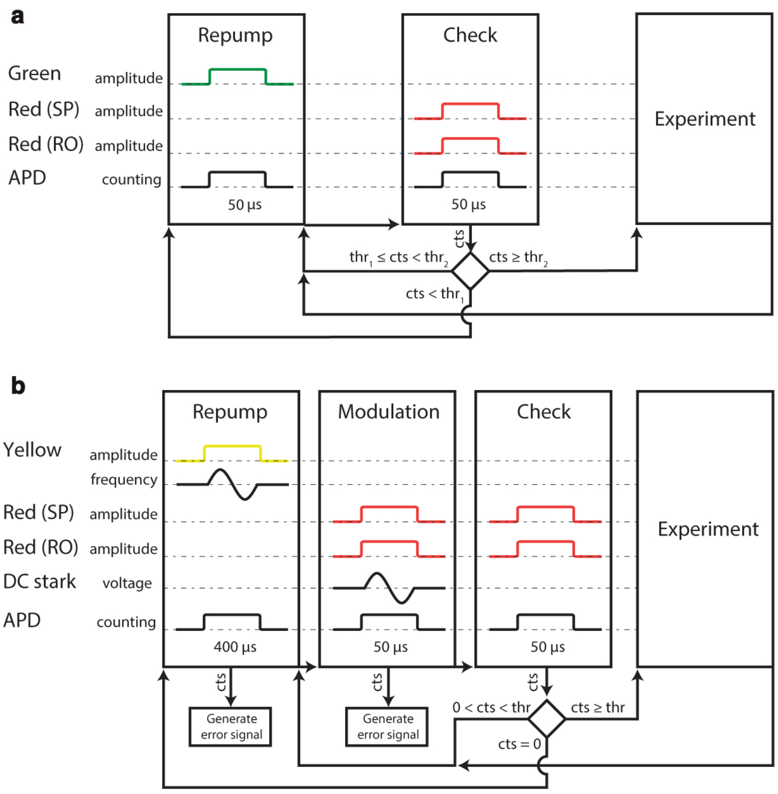


Figure 2.4: **Charge-Resonance check logic** a) Flow-chart including a repump process via green laser field. b) Flow-chart when Stark tuning is involved. In this case, there is a first Check sequence that is used to assess the error signal that modulates the applied DC voltage. Adapted from Hermans [18].

read-out in one shot. Namely, when at least one photon is detected during the readout, the outcome of the measurement is assigned to $|0\rangle$. If no photon is detected, the outcome is assigned to $|1\rangle$. No photon detection, though, might also be caused by photon loss in the collection. With this method, there is a non-zero probability that after the readout, there is a spin-flip. This process becomes relevant when the readout is performed mid-circuit and the outcome determines transformations on other qubits in the register. In this case, we aim for a non-destructive single-shot readout, that is performed by reducing the power of the laser field by up to a tenth and interrupting the illumination the moment a single photon is detected. For the NV used in this thesis, typical average read-out fidelities $\mathscr{F}_{avg}$ are in the range 90%-96%.

2.3.3. Spin control

The ultimate property for the NV to be used as a quantum processor is the capability of manipulating its spin state to perform quantum gates. Having defined in Section 2.2 the qubit space, single-qubit rotations are performed by addressing the transition $|0\rangle \rightarrow |1\rangle$

using microwaves. Particularly in this thesis, we use skewed Hermite pulses [21, 22], whose frequency is on resonance with the $|0\rangle \rightarrow |1\rangle$ transition. The phase of the pulse determines the axis of rotation of the single-qubit gate. The duration of the pulse is determined by the Rabi frequency, that is typically ~5MHz for power levels around 40dBm (42W). The pulse error on a single-qubit gate is estimated to be in the range 0.1%-1%, and it is mostly caused by electrical noise and unsuppressed laser fields.

2.4. REMOTE ENTANGLEMENT GENERATION

The capability that makes the NV a suitable platform for quantum communication, rather than just a computing platform, is the emission of narrow-band single photons that can be used for remote entanglement generation. In this section, we discuss two largely used protocols for entanglement generation. A general setup for remote entanglement generation comprises at least two network nodes with quantum processing capabilities (the NV), a classical mid-point constituted by a beam-splitter for photon interference purposes, photon detectors and simple logic for real-time heralding, a transmission link that can be made of optical fibers suited for the employed photon wavelength. In this dissertation, we only use optical fibers suitable for the transmission of 637nm light.

2.4.1. SINGLE-CLICK PROTOCOL

The first method was introduced by Cabrillo et al. [23] and Bose et al. [24], and it is based on the detection of a single photon at the midpoint, therefore the nomenclature "single-click".

Starting from a single node perspective, the requirement is the capability of generating a spin-photon entangled state. The electron spin at each node is initialized in a superposition state in the form:

$$|\psi\rangle_{A/B} = \sqrt{\alpha}\,|0\rangle + \sqrt{1-\alpha}\,|1\rangle \tag{2.2}$$

where α represents the bright state population. Sending a short optical pulse on resonance with the transition $|0\rangle \rightarrow \mathrm{E}_x/\mathrm{E}_y$, the NV can spontaneously emit a single photon. Considering the joint spin-photon state $|J\rangle$, we obtain:

$$|J\rangle_{A/B} = \sqrt{\alpha}\,|0\rangle_s|1\rangle_p + \sqrt{1-\alpha}\,|1\rangle_s|0\rangle_p \tag{2.3}$$

where s (p) refers to the spin (photon) state. The photons constitute flying qubits that are entangled with our spin state, or communication qubit. The flying qubits from each node are then sent to the mid-point, and by interfering we erase the which-path information, de facto entangling the photons. The detection of a single-click at one of the output ports of the beam-splitter, where the detectors are, heralds the following entangled state at each node:

$$|\Psi^{\pm}\rangle = \frac{|01\rangle \pm e^{i\theta}|10\rangle}{\sqrt{2}} \tag{2.4}$$

where θ is a phase due to the optical path difference between the two nodes that needs to be stabilized before a round of entanglement generation. The phase stabilization protocol utilized in this thesis is based on the one reported in Ref. [25], in which the phase is stabilized in two steps, via a local and a global interferometer. The sign $\pm$, instead, is due to the π phase difference between the two detectors, namely depending on which detector clicked, we herald a different Bell state. This means that for real-time heralding, a two-bit message needs to be communicated from the mid-point back to the end-nodes. The maximum achievable fidelity, in the case of a perfect setup, is limited by the choice of α, namely $F_{\Psi^{\pm}}^{max}$=1-α. The probability of success per attempt is given by $2\alpha p_{det}$, where p_{det} represents the detection probability of a single photon, which for the setup we use is in the range 10^{-4}-10^{-3}. This means that we can reach high fidelity (above the classical bound of 0.5) with rate of ~10Hz [26].

2.4.2. DOUBLE-CLICK PROTOCOL

The double-click protocol was first proposed by Barrett & Kok [27] and it is based on the consecutive detection of two photons within one entanglement attempt. It is straightforward to notice the first difference with the single-click protocol, which results in a lower generation rate. The sequence goes as follows: first at each node we prepare the electron spin qubit in a balanced superposition state and we send a short optical pulse on resonance with $|0\rangle \rightarrow E_x/E_y$, when we can spontaneously emit a photon that we call "early". Subsequently, we rotate the electron spin state by π and send another short optical pulse, spontaneously emitting a "late" photon. The joint spin-photon state at each node results as follows:

$$|\psi\rangle_{A/B} = \frac{|0\rangle_s |l\rangle_p + |1\rangle_s |e\rangle_p}{\sqrt{2}} \tag{2.5}$$

The early and late photons travel towards the mid-point, where they can interfere with the photons coming from the second node. When a photon is detected at the mid-point in the early and late time-window, this event heralds the following entangled state at the electron spin qubits:

$$|\Psi^{\pm}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{2.6}$$

Another immediate difference compared to the single-click protocol is that there is no optical phase difference in this case, removing the complexity of implementing classical phase stabilization protocols. However, the entanglement success probability is given by $0.5p_{det}^2$, which corresponds to rates around tens of mHz. For the fidelity, in a perfect scenario, this protocol has no limitations on the maximum achievable fidelity.

2.5. ADDITIONAL QUBITS

So far, the discussion has focused solely on the definition and control of the electron spin qubit. Diamonds naturally give the possibility to enlarge the qubit register by exploiting the hyperfine coupling of the electronic spin with the surrounding nuclear spin bath. The nitrogen atom of the NV, in both its possible isotopes ^{15}N and ^{14}N, possesses a nuclear spin that can be controlled [28, 29]. Similarly, the carbon atoms, and specifically the isotope ^{13}C, that is present in the diamond lattice with a natural abundance of 1.1%, can also be used as an additional data qubit or memory qubit. In this dissertation, we focus specifically on the control of single ^{13}C nuclear spins.
At first glance, this means that diamonds provide an intrinsically million-qubit platform with an optical interface in the visible range at almost zero cost in terms of engineer-

ing challenges compared to other computing platforms such as superconducting qubits, spin qubits in semiconductors or neutral atoms. Reality hits hard, though, as the disposition of such additional qubits in diamond is probabilistic and their simultaneous control poses many challenges.

2.5.1. CARBON-13 NUCLEAR SPIN CONTROL

To describe the hyperfine interaction between the electron spin and the ^{13}C nuclear spin we have to introduce the electron-nuclear Hamiltonian:

$$H/h = \gamma_c I \cdot B + A_\parallel S_z I_z + A_\perp S_z I_x \tag{2.7}$$

where γ_c=1.07084kHz/G is the nuclear gyromagnetic factor for the ^{13}C, I$_i$ are the spin-1/2 Pauli operators. $A_\parallel$ and $A_\perp$ represent the parallel and perpendicular hyperfine couplings, respectively. Such couplings identify each single ^{13}C nuclear spin in the lattice [30, 31]. To control the nuclear spin as a qubit, in the experiments in Chapter 3, we employ two techniques: dynamical decoupling (DD) [32] and dynamical decoupling radio-frequency (DDRF) [29, 33].
First of all, it is necessary to rewrite the Hamiltonian to uncover how to control the nuclear spin via the DD method. The second term in Eq. 2.7 can be rewritten considering that S_z can be either 0 or ±1 (one at the time, not both) and for simplicity the external magnetic field is aligned along the B$_z$ axis. This results in:

$$H = |0\rangle\langle 0| H_0 + |\pm 1\rangle |\pm 1\rangle H_{\pm 1} \tag{2.8}$$

$$H_0 = \omega_L I_z \tag{2.9}$$

$$H_{\pm 1} = (\omega_L \pm A_\parallel) I_z \pm A_\perp I_x \tag{2.10}$$

where ω_L=$\gamma_c B_z$ is the Larmor frequency of the nuclear spin. This rewritten version of the Hamiltonian shows that depending on the state of the electron spin, the nuclear spin precesses at a different frequency, specifically at ω_L when the electron spin is in $|0\rangle$ and $\omega_1 = \sqrt{(\omega_L \pm A_\parallel)^2 + A_\perp^2}$, along different axes. As a consequence, it is possible to perform single and two-qubit gates on the nuclear spin via the electron spin dynamics. We use tailored DD sequences of the form $(\tau - X_\pi - 2\tau - Y_\pi - \tau)^{N/2}$ [34], in which τ and N are specifically chosen to address the targeted nuclear spin.
In the DDRF method, the goal is to control the nuclear spin via direct RF driving, while the electron spin undergoes DD sequences to preserve its coherence. For this, we can rewrite Eq. 2.8 by introducing the RF field. The RF pulse has Rabi frequency Ω, phase ϕ and frequency ω. The frequency must be on resonance with the target nuclear spin, so $\omega = \omega_1$. For simplicity of this example, we assume that $A_\perp = 0$ and $(\omega_L - \omega_1) \gg \Omega$. Then, Eq. 2.8 becomes:

$$H = |0\rangle\langle 0| (\omega_L - \omega_1) I_z + |1\rangle\langle 1| \Omega(\cos(\phi) I_x + \sin(\phi) I_y) \tag{2.11}$$

This means that when the electron spin state is $|0\rangle$, the nuclear spin precesses around the z axis of the Bloch sphere with frequency $(\omega_L - \omega_1)$, while it is rotating around an axis in the x-y plane with a phase ϕ when the electron spin is $|1\rangle$. From this, conditional and unconditional gates can be performed.
Gates and control of ^{13}C nuclear spins are presented and used in Chapter 3.

2.6. QUBIT TIMESCALES

One of the principal qubit properties that must be taken into account when building a full-stack quantum network node is its operation timescales. In this regard, the community identifies three parameters to describe such timescales: the relaxation time T_1, the transverse relaxation time T_2^* and the total coherence time T_{coh}.
T_1 represents the decay time for a qubit initialized in one of the two eigenstates ($|0\rangle/|1\rangle$) into a mixed state of both $|0\rangle$ and $|1\rangle$.
T_2^* is the decay time of a generic superposition state in the form $1/\sqrt{2}(|0\rangle+|1\rangle)$ under dephasing mechanism.
T_{coh} is related to T_2^* and represents the dephasing time when coherence protection mechanisms are involved, such as tailored dynamical decoupling sequences. Therefore, $T_{coh} > T_2^*$.
The following table summarizes the state-of-the-art parameters for the NV qubit register, with distinction whether the NV was in a setup dedicated to quantum networking or computation. The table is not exhaustive, given that more exotic qubit types are effectively available [35, 36], such as P1 centers, ^{13}C and P1 pairs; however, their suitability for quantum network applications has not been fully investigated at the time of this dissertation.

Qubit type	T_1	T_2^*	T_{coh}
NV Electron spin	1h [37] (5min)	5μs	1s [37] (560ms)[38]
^{13}C Nuclear spin	~s - ~h [37]	[5-20]ms [39, 40],Ch. 3	[4-25]s [39] (21ms) [38]
^{14}N Nuclear spin	~s - ~h	23ms [39]	63s [39]

Table 2.1: NV center qubit register timescale summary. The value in parentheses refers to the actual value for a network setup (i.e. it can generate remote entanglement), while the first value represents the maximum reported in literature for setups that are mainly optimized for local information processing (if no quantum communication reference is given). Note that the T_1 of the nuclear spins is not a measured value, as the coherence time measurements are a more interesting parameter and they are not limited by T_1.

As it will be discussed in Chapter 5, a quantum network application needs the right qubit to be allocated at the right time for a given instruction. Information on qubit decay times is therefore an essential first step in building optimized schedulers for quantum networks [41, 42]. In a full-stack quantum network, the quantum hardware platform needs to interface with the software stack, which typically works on timescales of milliseconds. Hence, this sets boundaries in how the available qubits are used in the various applications requested by the users, as it will be discussed in Chapter 5. In this context, it is also necessary to discuss which factors are actually limiting the qubit timescales. For T_1, the main limitation is coming from unwanted interactions between the qubit and residual control fields (lasers and microwaves), which are particularly noticeable in a network setup, as included in Table 2.1. For T_2^*, the electron spin is limited by the interaction with the nuclear spin bath, which causes dephasing. For this reason, DD sequences can prolong the coherence time (T_{coh}) of this qubit, isolating it from the bath. The limitations of this method are mainly due to the imperfections of the applied microwave pulses. For the Carbon and Nitrogen single nuclear spins, T_2^* is longer than the one of the electron spin due to weaker interactions with surrounding nuclei, and DD sequences on the nuclear spins can also prolong the coherence time [38, 43].

2.7. EXPERIMENTAL SETUP

In this section, an overview of a typical network node experimental setup is shown in Figure 2.5, while a complete description in relation to their functionality is present in Chapter 3, 4 and 5. Additionally, an image of the optical setup used for each node is inserted in Figure 2.6.

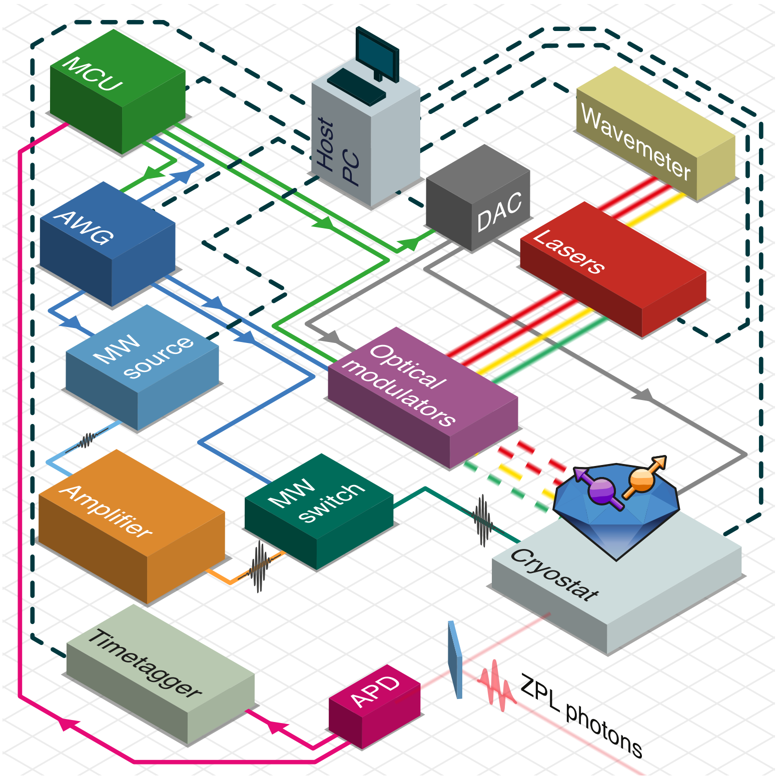


Figure 2.5: **Schematic of a single quantum network node setup.** The NV platform is controlled and manipulated both optically (via Lasers, whose frequency is locked with a wavemeter and switched on/off via optical modulators) and electronically via microwaves (MW). Photons are detected with either Avalanche Photo-Diodes or superconducting nanowire single-photon detectors (not in figure). Their arrival time is recorded with a Timetagger, whose signal is elaborated to herald entanglement. All these devices are orchestrated in the experiment via a MicroController Unit. The user (in this case, a PhD student) can interface with the hardware via a regular PC, coding the experiment utilizing several programming languages (Python, ADBasic, C++). Adapted from Pompili [44].

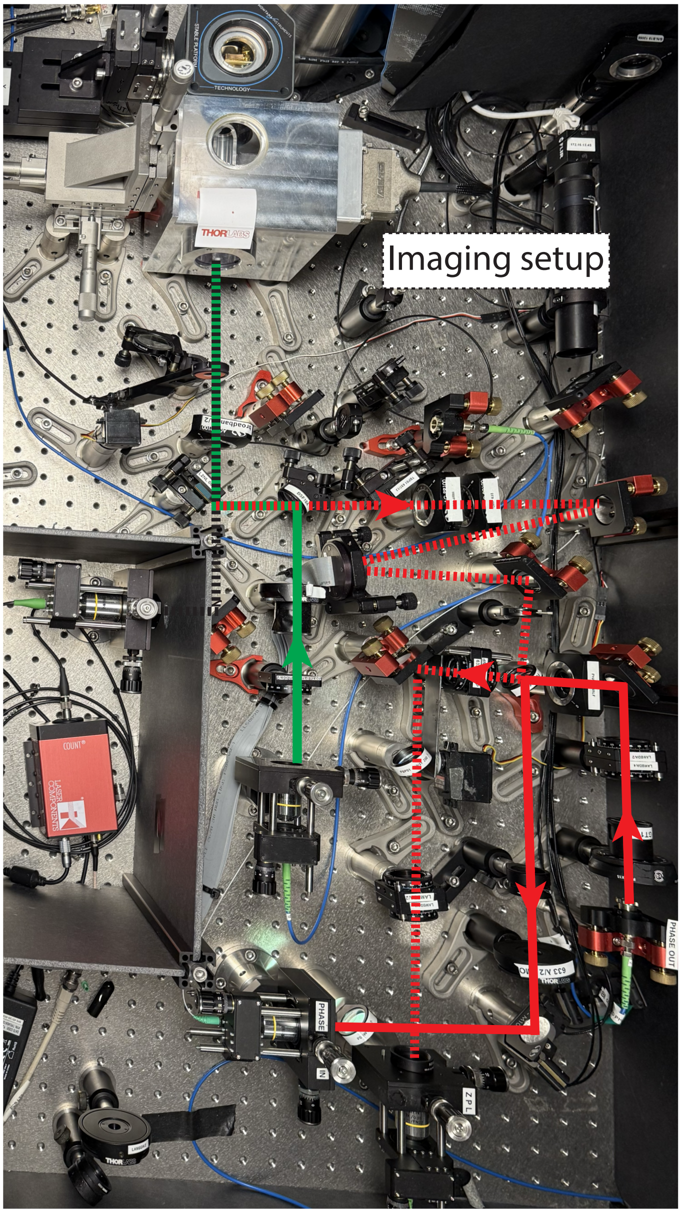


Figure 2.6: **Picture of the optical setup of a single network node.** Excitation light, in green, travels towards the objective case (metal box) to reach the NV chip. The PSB+ZPL photons are emitted (black dashed line) and a razor-cut dichroic mirror separates the two contributions (black dashed line for PSB and red dashed line for ZPL). The PSB photons are immediately detected with an APD. The ZPL photons pass through a narrow-band filter and a set of motorized waveplates that are used to suppress unwanted excitation light via cross-polarization. The unwanted laser field is actually collected in another path for the phase stabilization (plain red line). The single photons are collected via another coupler, whose fiber is connected to the mid-point. Additionally, an imaging setup with a white light is used to position the objective on the SIL we want to address. Courtesy of generations of PhDs and PostDocs from 2012 to 2025.

REFERENCES

[1] L. Du Preez. "Electron paramagnetic resonance and optical investigations of defect centres in diamond". PhD thesis. University of the Witwatersrand, Johannesburg, Sept. 1965.

[2] G. Davies and M. F. Hamer. "Optical studies of the 1.945 eV vibronic band in diamond". In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 348.1653 (Feb. 1976). Publisher: Royal Society, pp. 285–298. DOI: 10.1098/rspa.1976.0039.

[3] A. Gruber, A. Dräbenstedt, C. Tietz, L. Fleury, J. Wrachtrup, and C. V. Borczyskowski. "Scanning Confocal Optical Microscopy and Magnetic Resonance on Single Defect Centers". In: *Science* 276.5321 (June 1997), pp. 2012–2014. DOI: 10.1126/science.276.5321.2012.

[4] W. Pfaff. "Quantum measurement and entanglement of spin quantum bits in diamond". PhD thesis. Delft University of Technology, 2013.

[5] H. Bernien. "Control, measurement and entanglement of remote quantum spin registers in diamond". PhD thesis. Technische Universtiteit Delft, 2014.

[6] N. Kalb. "Diamond-based quantum networks with multi-qubit nodes". PhD thesis. Delft University of Technology, 2018.

[7] M. W. Doherty, N. B. Manson, P. Delaney, and L. C. L. Hollenberg. "The negatively charged nitrogen-vacancy centre in diamond: the electronic solution". In: *New Journal of Physics* 13.2 (Feb. 2011), p. 025019. DOI: 10.1088/1367-2630/13/2/025019.

[8] M. W. Doherty, N. B. Manson, P. Delaney, F. Jelezko, J. Wrachtrup, and L. C. Hollenberg. "The nitrogen-vacancy colour centre in diamond". In: *Physics Reports* 528.1 (July 2013), pp. 1–45. DOI: 10.1016/j.physrep.2013.02.001.

[9] J. R. Maze, A. Gali, E. Togan, Y. Chu, A. Trifonov, E. Kaxiras, and M. D. Lukin. "Properties of nitrogen-vacancy centers in diamond: the group theoretic approach". In: *New Journal of Physics* 13.2 (Feb. 2011), p. 025025. DOI: 10.1088/1367-2630/13/2/025025.

[10] P. Tamarat et al. "Stark Shift Control of Single Optical Centers in Diamond". In: *Physical Review Letters* 97.8 (Aug. 2006), p. 083002. DOI: 10.1103/PhysRevLett.97.083002.

[11] L. C. Bassett, F. J. Heremans, C. G. Yale, B. B. Buckley, and D. D. Awschalom. "Electrical Tuning of Single Nitrogen-Vacancy Center Optical Transitions Enhanced by Photoinduced Fields". In: *Phys. Rev. Lett.* 107 (26 Dec. 2011), pp. 266403-1–266403-5. DOI: 10.1103/PhysRevLett.107.266403.

[12] B. Hensen et al. "Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres". In: *Nature* 526.7575 (Oct. 2015), pp. 682–686. DOI: 10.1038/nature15759.

[13] S. M. Mansfield and G. S. Kino. "Solid immersion microscope". In: *Applied Physics Letters* 57.24 (Dec. 1990), pp. 2615–2616. DOI: 10.1063/1.103828.

[14] J. P. Hadden, J. P. Harrison, A. C. Stanley-Clarke, L. Marseglia, Y.-L. D. Ho, B. R. Patton, J. L. O'Brien, and J. G. Rarity. "Strongly enhanced photon collection from diamond defect centers under microfabricated integrated solid immersion lenses". In: *Applied Physics Letters* 97.24 (Dec. 2010), p. 241901. DOI: 10.1063/1.3519847.

[15] L. Robledo, H. Bernien, I. van Weperen, and R. Hanson. "Control and Coherence of the Optical Transition of Single Nitrogen Vacancy Centers in Diamond". In: *Physical Review Letters* 105.17 (Oct. 2010), p. 177403. DOI: 10.1103/PhysRevLett.105.177403.

[16] N. Aslam, G. Waldherr, P. Neumann, F. Jelezko, and J. Wrachtrup. "Photo-induced ionization dynamics of the nitrogen vacancy defect in diamond investigated by single-shot charge state detection". In: *New Journal of Physics* 15.1 (Jan. 2013). Publisher: IOP Publishing, p. 013064. DOI: 10.1088/1367-2630/15/1/013064.

[17] P. Siyushev, H. Pinto, M. Vörös, A. Gali, F. Jelezko, and J. Wrachtrup. "Optically Controlled Switching of the Charge State of a Single Nitrogen-Vacancy Center in Diamond at Cryogenic Temperatures". In: *Physical Review Letters* 110.16 (Apr. 2013), p. 167402. DOI: 10.1103/PhysRevLett.110.167402.

[18] S. Hermans. "Quantum Networks using Spins in Diamond". PhD thesis. Delft University of Technology, 2022.

[19] L. Robledo, L. Childress, H. Bernien, B. Hensen, P. F. A. Alkemade, and R. Hanson. "High-fidelity projective read-out of a solid-state spin quantum register". In: *Nature* 477.7366 (Sept. 2011). Publisher: Nature Publishing Group, pp. 574–578. DOI: 10.1038/nature10401.

[20] D. M. Irber, F. Poggiali, F. Kong, M. Kieschnick, T. Lühmann, D. Kwiatkowski, J. Meijer, J. Du, F. Shi, and F. Reinhard. "Robust all-optical single-shot readout of nitrogen-vacancy centers in diamond". In: *Nature Communications* 12.1 (Jan. 2021), p. 532. DOI: 10.1038/s41467-020-20755-3.

[21] S. W. Warren. "Effects of arbitrary laser or NMR pulse shapes on population inversion and coherence". In: *The Journal of Chemical Physics* 81 (1984). DOI: 10.1063/1.447644.

[22] L. M. K. Vandersypen and I. L. Chuang. "NMR techniques for quantum control and computation". In: *Rev. Mod. Phys.* 76 (4 Jan. 2005), pp. 1037–1069. DOI: 10.1103/RevModPhys.76.1037.

[23] C. Cabrillo, J. I. Cirac, P. García-Fernández, and P. Zoller. "Creation of entangled states of distant atoms by interference". In: *Physical Review A* 59.2 (Feb. 1999), pp. 1025–1033. DOI: 10.1103/PhysRevA.59.1025.

[24] S. Bose, P. L. Knight, M. B. Plenio, and V. Vedral. "Proposal for Teleportation of an Atomic State via Cavity Decay". In: *Physical Review Letters* 83.24 (Dec. 1999), pp. 5158–5161. DOI: 10.1103/PhysRevLett.83.5158.

[25] M. Pompili et al. "Realization of a Multinode Quantum Network of Remote Solid-State Qubits". In: *Science* 372.6539 (2021), pp. 259–264. DOI: 10.1126/science.abg1919.

[26] P. C. Humphreys, N. Kalb, J. P. J. Morits, R. N. Schouten, R. F. L. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson. "Deterministic delivery of remote entanglement on a quantum network". In: *Nature* 558.7709 (June 2018), pp. 268–273. DOI: 10.1038/s41586-018-0200-5.

[27] S. D. Barrett and P. Kok. "Efficient high-fidelity quantum computation using matter qubits and linear optics". In: *Physical Review A* 71.6 (June 2005), p. 060310. DOI: 10.1103/PhysRevA.71.060310.

[28] W. Pfaff et al. "Unconditional quantum teleportation between distant solid-state quantum bits". In: *Science* 345.6196 (Aug. 2014), pp. 532–535. DOI: 10.1126/science.1253512.

[29] C. E. Bradley, J. Randall, M. H. Abobeih, R. C. Berrevoets, M. J. Degen, M. A. Bakker, M. Markham, D. J. Twitchen, and T. H. Taminiau. "A Ten-Qubit Solid-State Spin Register with Quantum Memory up to One Minute". In: *Physical Review X* 9.3 (Sept. 2019), p. 031045. DOI: 10.1103/PhysRevX.9.031045.

[30] M. H. Abobeih, J. Randall, C. E. Bradley, H. P. Bartling, M. A. Bakker, M. J. Degen, M. Markham, D. J. Twitchen, and T. H. Taminiau. "Atomic-scale imaging of a 27-nuclear-spin cluster using a quantum sensor". In: *Nature* 576.7787 (Dec. 2019), pp. 411–415. DOI: 10.1038/s41586-019-1834-7.

[31] G. L. Van De Stolpe, D. P. Kwiatkowski, C. E. Bradley, J. Randall, M. H. Abobeih, S. A. Breitweiser, L. C. Bassett, M. Markham, D. J. Twitchen, and T. H. Taminiau. "Mapping a 50-spin-qubit network through correlated sensing". In: *Nature Communications* 15.1 (Mar. 2024), p. 2006. DOI: 10.1038/s41467-024-46075-4.

[32] T. H. Taminiau, J. Cramer, T. van der Sar, V. V. Dobrovitski, and R. Hanson. "Universal control and error correction in multi-qubit spin registers in diamond". In: *Nature Nanotechnology* 9.3 (Mar. 2014), pp. 171–176. DOI: 10.1038/nnano.2014.2.

[33] H. van Ommen, G. van de Stolpe, N. Demetriou, H. Beukers, J. Yun, T. Fortuin, M. Iuliano, A.-P. Montblanch, R. Hanson, and T. Taminiau. "Improved Electron-Nuclear Quantum Gates for Spin Sensing and Control". In: *PRX Quantum* 6.2 (Apr. 2025). Publisher: American Physical Society, p. 020309. DOI: 10.1103/PRXQuantum.6.020309.

[34] G. De Lange, Z. H. Wang, D. Ristè, V. V. Dobrovitski, and R. Hanson. "Universal Dynamical Decoupling of a Single Solid-State Spin from a Spin Bath". In: *Science* 330.6000 (Oct. 2010), pp. 60–63. DOI: 10.1126/science.1192739.

[35] H. P. Bartling, M. H. Abobeih, B. Pingault, M. J. Degen, S. J. H. Loenen, C. E. Bradley, J. Randall, M. Markham, D. J. Twitchen, and T. H. Taminiau. "Entanglement of Spin-Pair Qubits with Intrinsic Dephasing Times Exceeding a Minute". In: *Physical Review X* 12.1 (Mar. 2022), p. 011048. DOI: 10.1103/PhysRevX.12.011048.

[36] H. Bartling. "Quantum control of interacting nuclear and electron spins in diamond". PhD thesis. Delft University of Technology, 2023. DOI: 10.4233/UUID:5C27F652-137C-46EC-B31B-BA01367323B4.

[37] M. H. Abobeih, J. Cramer, M. A. Bakker, N. Kalb, M. Markham, D. J. Twitchen, and T. H. Taminiau. “One-Second Coherence for a Single Electron Spin Coupled to a Multi-Qubit Nuclear-Spin Environment”. In: *Nature Commun.* 9.1 (2018), pp. 1–8. DOI: 10.1038/s41467-018-04916-z.

[38] S. L. N. Hermans, M. Pompili, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson. “Qubit teleportation between non-neighbouring nodes in a quantum network”. In: *Nature* 605.7911 (May 2022), pp. 663–668. DOI: 10.1038/s41586-022-04697-y.

[39] C. E. Bradley, J. Randall, M. H. Abobeih, R. C. Berrevoets, M. J. Degen, M. A. Bakker, M. Markham, D. J. Twitchen, and T. H. Taminiau. “A Ten-Qubit Solid-State Spin Register with Quantum Memory up to One Minute”. In: *Phys. Rev. X* 9.3 (2019), pp. 031045-1–031045-12. DOI: 10.1103/PhysRevX.9.031045.

[40] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. W. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson. “Entanglement Distillation Between Solid-State Quantum Network Nodes”. In: *Science* 356.6341 (2017), pp. 928–932. DOI: 10.1126/science.aan0070.

[41] T. R. Beauchamp, H. Jirovská, S. Gauthier, and S. Wehner. *A Modular Quantum Network Architecture for Integrating Network Scheduling with Local Program Execution.* arXiv:2503.12582 [quant-ph]. Mar. 2025.

[42] G. Ni, L. Ho, and H. Claussen. *Entanglement Request Scheduling in Quantum Networks Using Deep Q-Network.* arXiv:2505.12461 [quant-ph]. May 2025. DOI: 10.48550/arXiv.2505.12461.

[43] C. E. Bradley et al. “Robust quantum-network memory based on spin qubits in isotopically engineered diamond”. In: *npj Quantum Information* 8.1 (Oct. 2022), p. 122. DOI: 10.1038/s41534-022-00637-w.

[44] M. Pompili. “Multi-Node Quantum Networks with Diamond Qubits”. PhD thesis. Delft University of Technology, 2021.

3
Unconditionally teleported quantum gates between remote solid-state qubit registers

This project has completely confiscated my life, darling.
Consumed me as only hero work can.
My best work, I must admit.
Simple, elegant, yet bold.

Edna Mode - The Incredibles

M. Iuliano, N. Demetriou, H. B. van Ommen, C. Karels, T. H. Taminiau, R. Hanson

Quantum networks connecting quantum processing nodes via photonic links enable distributed and modular quantum computation. In this framework, quantum gates between remote qubits can be realized using quantum teleportation protocols. The essential requirements for such non-local gates are remote entanglement, local quantum logic within each processor, and classical communication between nodes to perform operations based on measurement outcomes. Here, we demonstrate an unconditional Controlled-NOT quantum gate between remote diamond-based qubit devices. The control and target qubits are Carbon-13 nuclear spins, while NV electron spins enable local logic, readout,

The results of this chapter are in preparation for a peer-reviewed publication. A pre-print is available at arXiv:2601.04848

and remote entanglement generation. We benchmark the system by creating a Greenberger-Horne-Zeilinger state, showing genuine 4-partite entanglement shared between nodes. Using deterministic logic, single-shot readout, and real-time feed-forward, we implement non-local gates without post-selection. These results demonstrate a key capability for solid-state quantum networks, enabling exploration of distributed quantum computing and testing of complex network protocols on fully integrated systems.

3.1. INTRODUCTION

Quantum networks can connect separate quantum processors to unlock capabilities and applications that do not have a classical counterpart. Examples range from long-range secure communication and distributed quantum computing to enhanced quantum sensing [1, 2]. In particular, distributed quantum computing exploits quantum links between small quantum processors to build larger networks that allow the system to scale in size or distance [3, 4]. Key to such modular architectures are non-local quantum operations, which can be performed using quantum teleportation protocols [5, 6]. Quantum gate teleportation (QGT) poses stringent requirements on the qubit platform, including distribution of remote entanglement, executing local operations within a multi-qubit register and performing non-local feed-forwarded operations within the coherence time of its qubit register. To avoid low gate success probabilities and ensure scalability, QGT should run unconditionally on the outcomes of the mid-circuit measurements of the teleportation protocol. This implies that once entanglement is shared between the processors, the gates should operate deterministically.

Pioneering experiments have demonstrated probabilistic remote QGT in purely photonic systems [7, 8] as well as with photonic systems combined with quantum memories [9–11]. These demonstrations are readily extensible to longer distances, but could not achieve unconditional operation as they inherently rely on post-selection. Unconditional (and even fully deterministic) QGT has recently been achieved within a single cryogenic system with superconducting qubits [12, 13], within a segmented ion trap system [14] and between nearby trapped ion systems [15].

Here, we implement unconditional QGT between solid-state qubits across an extensible optical link. In particular, we employ Nitrogen-Vacancy (NV) spin qubits in diamond. This platform has previously enabled heralded entanglement generation over 10 km distance using 25 km of deployed fiber [16], as well as the realization of basic network protocols on a three-node network [17, 18].

We first generate a 4-qubit Greenberger–Horne–Zeilinger state using two independently controlled two-qubit registers. Each register consists of an NV center electron spin qubit and a ^{13}C nuclear spin qubit, housed in separate cryostats (Fig. 5.2a). We then perform the teleportation of a Controlled-NOT quantum gate between the two remote nuclear spin qubits. In both cases, we exclude post-selection and data filtering, unconditionally accepting all intermediate measurement outcomes, and use real-time feedforward operations within the registers' coherence time. This demonstration of unconditional QGT is made possible by several innovations compared to previous NV center network experiments [17–19], including tuning of the optical transition frequency at high-magnetic-field, different tailored control methods for the nuclear spin qubits in the two nodes (Dynamical Decoupling [20] and Dynamical Decoupling-Radio Frequency [21, 22]) in combination with remote entanglement generation and node synchronization, and novel nuclear spin qubit phase tracking strategies (see Sec. 3.2.3) during network activity.

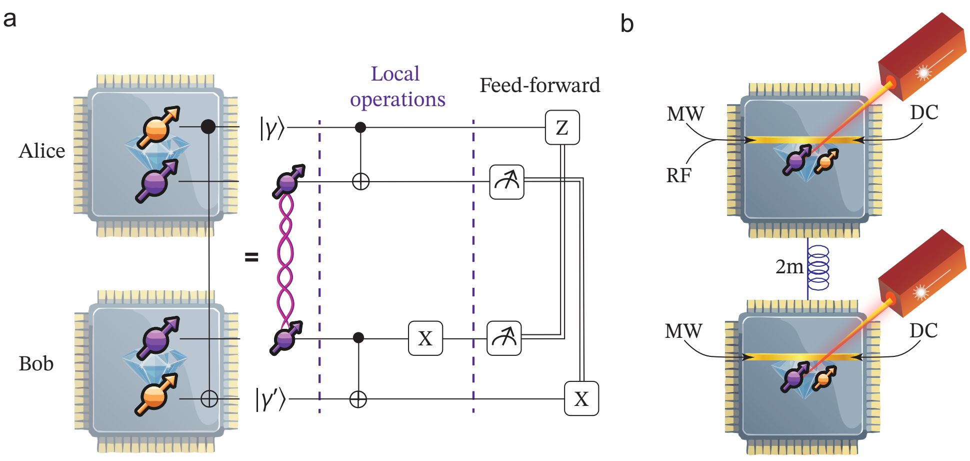


Figure 3.1: **Experiment and resources overview.** a) C-NOT quantum gate teleportation: we use two separated nodes based on NV-center defects in diamond. Each node is composed of two qubits: one communication qubit (in purple), obtained via controlling the electron spin of the NV, and one data qubit (in yellow), made by controlling a single ^{13}C nuclear spin. To realize a non-local C-NOT gate between the data qubits, a teleportation protocol is used, including the generation of remote entanglement, local operations and feed-forward operations. b) Concept of the control setup for the two-qubit register on each node, separated by 2m of optical fibers. The qubits are manipulated via MicroWaves (in GHz range), and additionally RadioFrequency (MHz) waves for Alice's data qubit, sent along a gold stripline. Preparation, readout and entanglement generation require optical control via red (637nm) and yellow (575nm) lasers, whose outputs are combined in a single excitation optical path. A DC voltage is applied to use the Stark effect for tuning the emitted photon frequency of the two nodes.

3.2. Results

We employ two setups (Alice and Bob) hosting diamond NV centers that are physically separated by 2m of optical fiber in a lab and cooled down to T_{Alice}= 3.9K, T_{Bob}= 3.4K (Fig. 5.2b). The electron spin qubits, referred to hereafter as communication qubits, are manipulated using microwave (MW) pulses delivered on-chip via gold striplines. Initialization and single-shot readout of these qubits are performed via spin-selective optical transitions [23].

3.2.1. Nuclear spin control

In addition to the communication qubit, each node employs a hyperfine-coupled ^{13}C nuclear spin as a data qubit. The Hamiltonian that describes the interaction between the electron spin qubit and the nuclear spin qubit is approximated by [20]:

$$H = \omega_L I_z + A_\parallel S_z I_z + A_\perp S_z I_x \tag{3.1}$$

where ω_L=γB$_z$ is the Larmor frequency of the nuclear spin in the external magnetic field B_z. The external magnetic field for Alice (Bob) is 189mT (31mT). S_i and I_i are the spin operators for the electron spin and the nuclear spin, respectively. $A_\parallel$ and $A_\perp$ are the parallel and perpendicular hyperfine coupling parameters (more details in the Supplementary).

We optimize the control of the data qubits by using two different techniques. At Alice, we use the DDRF method [21, 22], in which the data qubit is directly driven via phase-controlled RF pulses, interleaved with Dynamical Decoupling sequences to protect the communication qubit from decoherence. Bob's data qubit, instead, is manipulated using tailored DD sequences, therefore achieving control via the communication qubit dynamics [20, 24, 25]. For Alice, the high magnetic field regime provides significant advantages in qubit control. In this regime, the DDRF technique enables control of nuclear spins with small $A_\perp$ (compared to ω_L). The DDRF gates bring versatility and multi-qubit control while showing similar gate fidelity as the DD gates used on this qubit in Ref.[18]. Here, we exploit the feature that the gate duration is easily adaptable to timing constraints set by the other node, contributing to optimized experimental rates and higher overall system fidelity. Additionally, when DDRF is combined with remote entanglement generation (see Sec. 3.2.3 and Supplementary Information), this enables less complex and more efficient phase tracking of the data qubit.

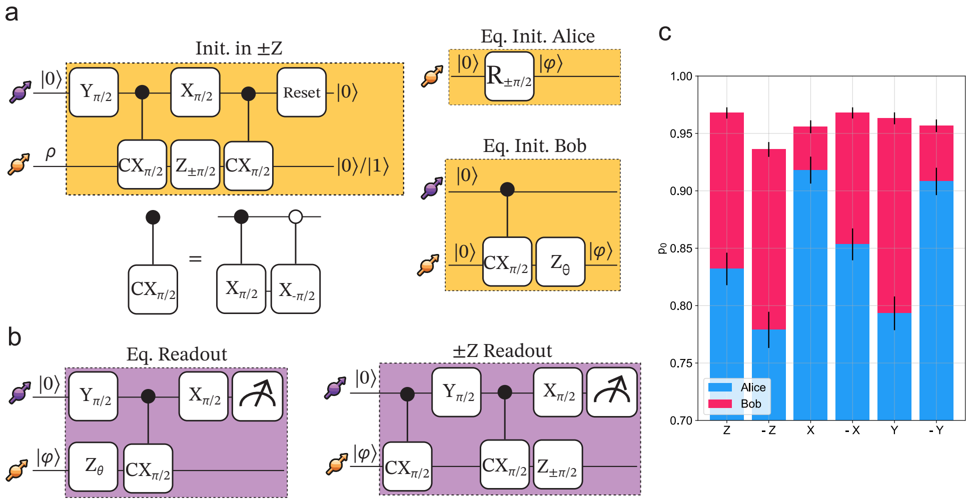


Figure 3.2: **Data qubit preparation.** a) Data qubit initialization sequence. $\pm Z$ initialization with the electron spin qubit in $|0\rangle$ deterministically enables the initialization in one of the two eigenstates. The initialization gate is completed when the electron spin qubit is optically reset to the state $|0\rangle$. Initialization on the equatorial plane is obtained by adding an unconditional gate for Alice along a tailored combination of $\hat{x}$ and $\hat{y}$ axes when initialized in $|0\rangle$, or using a conditional gate and a phase gate with an arbitrary angle θ for Bob. b) Readout of the data qubit. The state of the data qubit is mapped on the communication qubit and then optically read out. c) Measured fidelity with the ideal state for a set of unbiased initial states along the Bloch sphere.

In Fig. 5.3a-b, we show the gate sequences to initialize and read out the data qubits. The sequences are control technique-independent, unless otherwise specified. Both the initialization and read-out sequences are assisted by the communication qubit. The Z-gates on the data qubit for Alice are performed by updating the phase on the local oscillator of the RF field, and for Bob by either waiting a certain amount of time or playing a specific DD sequence based on the phase we want to imprint. In Fig. 5.3c, we show the measured fidelity of each data qubit with the ideal state for initialization in six unbiased states along the Bloch sphere: $\pm Z$, $\pm X$, $\pm Y$. We achieve average fidelity, corrected for known tomography errors on the communication spin [17], of 85(1)% for Alice and 96(1)% for Bob.

The main sources of infidelity are pulse errors on the communication qubit, leakage of laser light causing communication qubit dephasing, errors in the mapping of the state of the data qubit onto the communication qubit, and the imperfect decoupling of the communication qubit from the surrounding nuclear spin bath.

3.2.2. REMOTE ENTANGLEMENT GENERATION

For the generation of remote entanglement, the emission of indistinguishable photons from the remote communication qubits is critical. We introduce DC Stark tuning [26] on both setups to achieve indistinguishability in photon frequency, together with charge repumping using 575nm light on resonance with the Zero-Phonon Line of the neutral charge state (NV^0) to counteract ionization. The novelty of DC Stark tuning at high magnetic field is enabled by efficient charge repumping using a strongly power-broadened 575nm pulse (see Supplementary) together with operating in favorable strain conditions.

Remote entanglement between the two nodes is generated using photonic number-state encoding [27–29]. The experimental sequence, depicted in Fig. 3.3a, involves the generation of electron spin-photon entangled states at each node, in the form of $\sqrt{\alpha}\,|0\rangle_c|1\rangle_p + \sqrt{1-\alpha}\,|1\rangle_c|0\rangle_p$, where $|i\rangle_c$ and $|i\rangle_p$ are the communication qubit and photonic qubit states, respectively, and α is a parameter set in experiment. The spontaneously emitted photons travel towards a mid-point station, composed of a 50:50 in-fiber beam-splitter, whose output ports are connected to Superconducting Nanowire Single-Photon Detectors (SNSPDs). The detection of a single photon heralds, in a perfect scenario, the two-qubit state $(|01\rangle_c \pm e^{i\phi}|10\rangle_c)/\sqrt{2}$, with probability (and hence state fidelity) of 1-α. Here ϕ is the optical phase difference between the two paths at the beam splitter, which is actively stabilized before entanglement generation [17]. The sign of the entangled state depends on which detector clicked.

In Fig. 3.3a we report the measured values of the entangled state correlators along with their simulated values for the states Ψ^+ and Ψ^-. We obtain state fidelities of 77(2)% and 76(2)% for Ψ^+ and Ψ^- respectively, with an average α=0.045 between the two setups. For comparison, the average simulated state fidelity is 79%. Detailed explanations about the protocol, the source of errors and the simulated values are discussed in Ref. [29].

3.2.3. DATA QUBIT COHERENCE DURING NETWORKING

The data qubits, encoded in nuclear spins, possess a long intrinsic coherence time (tens of milliseconds for the current devices). However, during entanglement attempts, the coherence of the data qubit undergoes a faster decay due to its coupling to the communication qubit, whose state cannot be perfectly tracked in entanglement attempts [30].

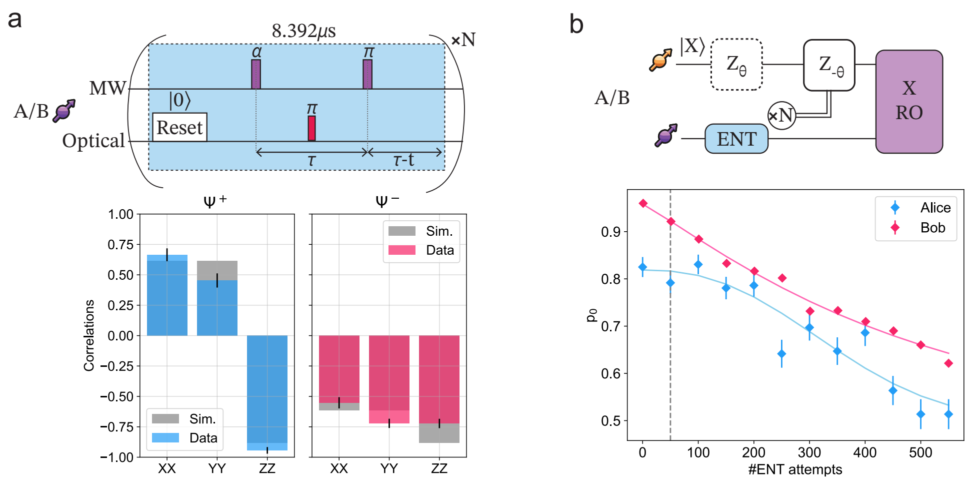


Figure 3.3: **Network activity characterization.** a) Remote entanglement generation and entangled state fidelity. At each node, a single attempt includes a reset pulse to initialize the communication qubit in $|0\rangle$, a MW α-pulse, which brings the qubit in an unbalanced superposition state; a short (1ns) optical π-pulse that excites the population in the $|0\rangle$ state to the excited state, enabling spontaneous emission of a single photon; a MW π-pulse played at a time τ after the α-pulse and τ before the next reset pulse in the subsequent attempt, hence a distance $\tau - t$ from the end of a single attempt. The total duration of a single attempt is 8.392μs (details in the Methods section), which is repeated N times. The lower panel shows measured and simulated correlations. b) Characterization of the nuclear spin dephasing during entanglement attempts. During each entanglement attempt, the nuclear spin gains a deterministic phase, which we correct based on the number of repetitions N before entanglement is heralded. Additional stochastic phases, e.g. due to the spin reset, cause decoherence. The plot shows the state fidelity of the nuclear spin state, initialized in a superposition state, for different numbers of entanglement attempts. The dashed grey line represents the chosen timeout for entanglement generation N_{max}=50.

The dephasing time under network activity is parametrized by the number of entanglement attempts $N_{1/e}$ after which the fidelity contrast of the state stored in the data qubit has decreased by $1/e$. During an entanglement attempt, the time that the communication qubit is in $|0\rangle$ versus $|1\rangle$ is not deterministic, decomposing the total phase acquired by the data qubit in a static offset plus stochastic variations. Therefore, real-time tracking of the phase becomes critical (Fig. 3.3b) and $N_{1/e}$ is thus affected by the accuracy of the nuclear spin evolution phase tracking.

For Alice, the phase tracking is executed on the local oscillator of the data qubit RF driving field, updating the phase of the next RF pulse. The average phase picked up during a single entanglement attempt is calibrated beforehand. For Bob's data qubit, the rephasing after entanglement attempts is achieved via an XY8 DD sequence on the electron spin, in which the inter-pulse delay is tailored to result in the specific phase we want to imprint on the nuclear spin evolution [18]. Additionally, it is important to protect the communication qubit during this process and therefore it is key to avoid inter-pulse delays for which the communication qubit couples to other nuclear spins in its environment. The optimized inter-pulse delays are also calibrated beforehand and compiled in a look-up table for the control device (Supplementary Material).

In Fig. 3.3b we report the fidelity of the input state on the data qubit as a function of the number of entanglement attempts while employing the above-mentioned rephasing techniques. We extract the parameter $N_{1/e}$ by fitting the data to the exponential decay curve $A \cdot e^{-(n/N_{1/e})^d} + 0.5$, where A is related to the initial fidelity and d is the exponential decay. We obtain a $N_{1/e}$ of 391(31) (479(19)) for Alice (Bob) with d of 2.4(7) (1.1(1)) and A equals 0.32(2) (0.46(1)). Based on these results, we set the timeout for the entanglement generation to 50 attempts before re-initializing the data qubit. The choice of the timeout is a trade-off between the experiment rates and corresponding fidelities. We note that the coherence time during entanglement attempts may be further prolonged by introducing dynamical decoupling pulses for the data qubit, as shown in Ref. [18, 19].

3.2.4. 4-QUBIT GHZ STATE

Next, we combine all the above techniques for the creation of a 4-qubit GHZ state distributed over 2 nodes. Besides demonstrating the generation of a crucial resource state for quantum information protocols [31], this experiment serves as a system benchmark for the non-local C-NOT gate, as it utilizes the same gate set for local operations, together with fixed sequences for initialization, remote entanglement generation, rephasing of the data qubit after entanglement using real-time feedforward, mid-circuit readout of the communication qubit and data qubit readout.

The circuit diagram in Fig. 5.4a shows the gate sequence for the creation of the state $\Psi_{GHZ} = 1/\sqrt{2}\ (|0\rangle_{Ad}|1\rangle_{Ac}|1\rangle_{Bc}|0\rangle_{Bd} - |1\rangle_{Ad}|0\rangle_{Ac}|0\rangle_{Bc}|1\rangle_{Bd})$, with A (B) indicating the node Alice (Bob) and c (d) the communication (data) qubit in each node. The initialization of the data qubit is achieved via the circuits shown in Fig. 5.3a. To ensure that both nodes enter the remote entanglement generation sequence at the same time, the initialization of the two data qubits is synchronized by delaying the start of the initialization of the fastest node. After successful entanglement generation, Bob's data qubit is rephased based on the number of entanglement attempts used. In case the generated remote entangled state is Ψ^-, the midpoint communicates this to Alice, where an extra phase gate is added in real time to the tomography pulses of the data qubit. Effectively, this ensures that the remote entangled state is Ψ^+ irrespective of the photon detection pattern. Next, Ψ^+ is transformed into Φ^+ by a Pauli correction gate applied at Alice. Subsequently, local operations on the qubit registers are performed that entangle the data qubits with the communication qubits. Phase gates on the data qubits at the end of the protocol are compiled into the final tomography pulses. Experimental details of the tomography are discussed in the Methods section.

In Fig. 5.4b we report the measurement results of the 4-qubit correlators, along with the predicted values from simulations using measured parameters. This data as well as data presented below is corrected for known tomography errors (see Supplementary material for details). We obtain a state fidelity F_{GHZ}=64(4)%, in good agreement with the value predicted from simulations of F^{sim}_{GHZ}=66%. The observed value of F_{GHZ} exceeding 0.5 proves the generation of genuine four-partite entanglement across the two nodes [32]. We emphasize that this state is generated without any post-selection, constituting to the best of our knowledge, the largest heralded GHZ state across optically connected solid-state network nodes demonstrated so far.

The GHZ state fidelity is mainly limited by imperfections in the remote entangled

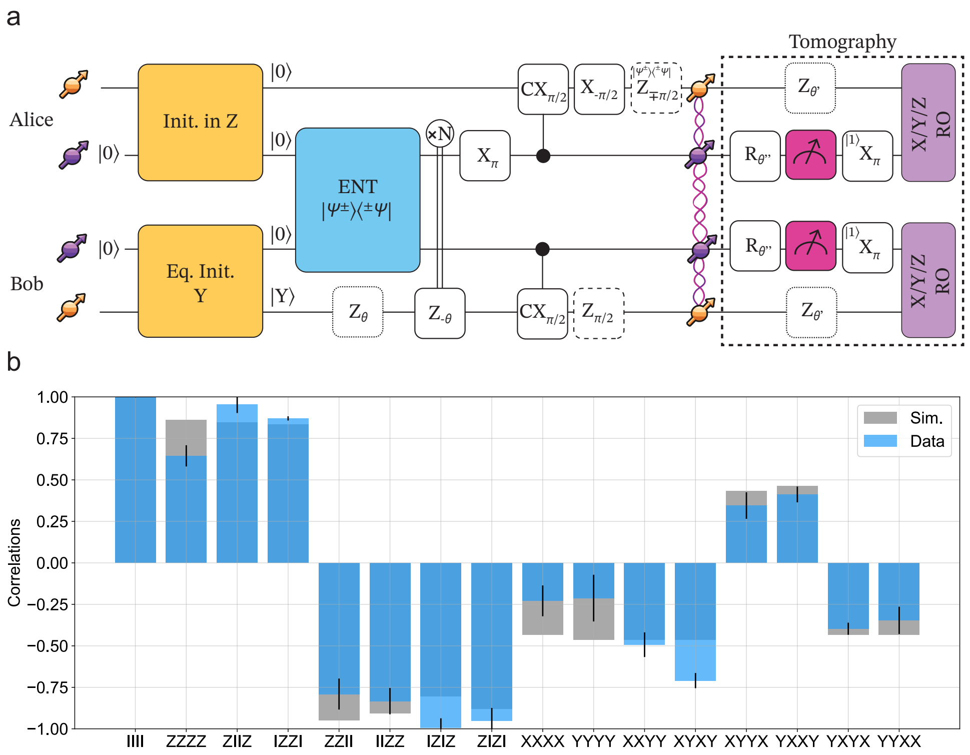


Figure 3.4: **Realization of a remote 4-qubit GHZ state.** a) Circuit diagram. The data qubits are initialized using the sequence in Fig. 5.3a. After heralding entanglement, a rephase gate is played on Bob's data qubit. Subsequently, a set of local operations completes the generation of the GHZ state. Dashed gates represent gates that are not individually executed, but are compiled in the readout sequence. To measure the correlators in b), we first measure the electron spin state, using single-qubit gates for the measurement basis selection and a non-destructive optical readout (highlighted in magenta). Every outcome is accepted. If the outcome is $|1\rangle$, a π-pulse flips the state to ensure the assisted-readout always starts with the communication qubit in $|0\rangle$. During the readout of the electron spin qubit, the data qubit picks up another phase θ' depending on the measurement outcome, whose rephasing is also compiled in the subsequent assisted-readout. b) GHZ correlator results and corresponding simulated values.

state generation and initialization of the data qubits. Separately, incorrect state assignment of the communication qubit measurement outcome in the tomography leads to a wrong rephasing sequence applied to the data qubit. We estimate that this occurs for ~5% (~9%) of the measured $|1\rangle$ outcomes for Alice (Bob), causing tomography errors that reduce the observed state fidelity by ~7%. We thus estimate that the actual GHZ state fidelity is about 71%.

3.2.5. C-NOT GATE TELEPORTATION

We realize a C-NOT gate between the data qubits of the two remote nodes, using the gate circuit shown in Fig. 5.2a. Compared to the GHZ state generation, we add real-time feed-forwarded operations based on the exchange of classical information between the nodes. In Fig. 5.5a, we report the circuit diagram presented in Fig. 5.2a translated into native gates of our platform. Note that of the local operations (gates depicted with a purple boundary), the single-qubit gates on Alice's data qubit are executed right after the initialization and right after the mid-circuit measurement. This compilation optimizes the synchronization between the nodes, taking into account the different gate durations on the two nodes. This synchronization is required not only during the entanglement attempts (as in the GHZ case) but also when exchanging classical information for the real-time feed-forward operations.

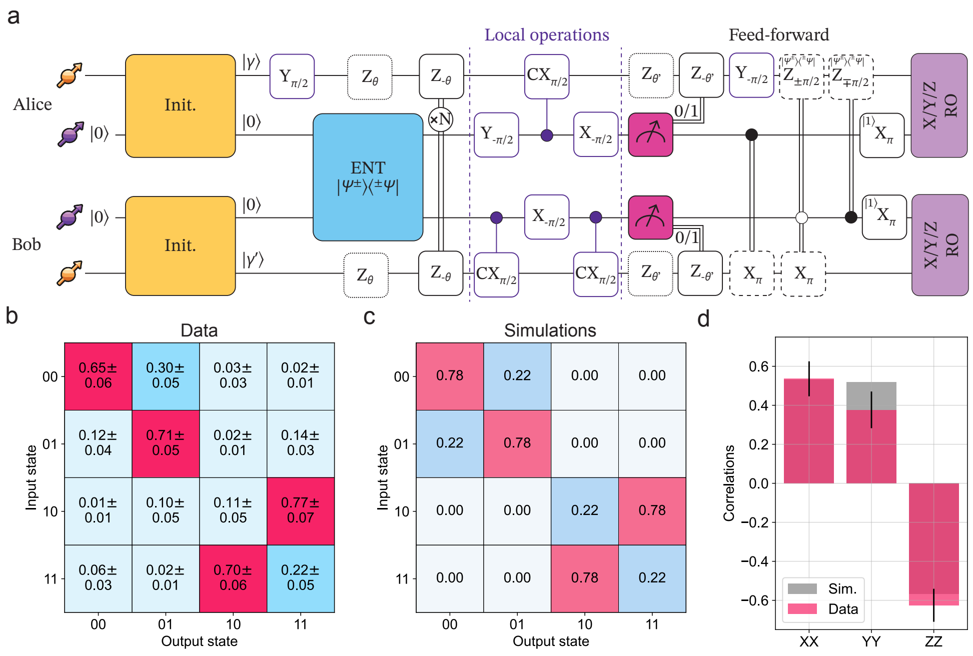


Figure 3.5: **Non-local C-NOT gate**. a) Circuit diagram using native NV gates. The gates in purple compile a local C-NOT gate. For Alice, the unconditional gates on the data qubit are performed before entanglement and after the mid-circuit readout for synchronization purposes. The feed-forward operation (dashed gates) is compiled in the readout sequence. The magenta mid-circuit measurement indicates a non-destructive readout. b) Measured classical truth-table. The initial states on the data qubit are the eigenstates and we report the non-local two-qubit state fidelity. As expected, we see a bit-flip in Bob's state when Alice's input state is $|1\rangle$. c) Simulated classical truth table. d) Generation of an entangled state via the non-local C-NOT gate. We prepare the data qubits in $|X\rangle_A$ and $|1\rangle_B$, to obtain the entangled state Ψ^+. The histogram shows the correlator expectation values together with their simulated values.

We first reconstruct the classical truth table of the C-NOT gate. For this, the initial states prepared on each data qubit are the two eigenstates $|0\rangle$ and $|1\rangle$. On Bob's side, this results in the qubit not being subjected to additional dephasing during the entanglement attempts. In contrast, on Alice's side the data qubit is in a superposition state during the network activity, due to the local gate being executed before the entanglement generation as discussed above; therefore, the dephasing mechanisms and the phase tracking reported in Fig. 3.3b are relevant.

The results of the truth table measurements are displayed in Fig. 5.5b. For comparison, we include in Fig. 5.5c the simulated truth table. The results show the correct gate action with the four two-qubit fidelities being above 70% on average, in reasonable quantitative agreement with the simulations.

Subsequently, we show the quantum-coherent nature of the non-local C-NOT gate by generating an entangled state between the data qubits. Specifically, we prepare Alice's data qubit in $|X\rangle$ and Bob's data qubit in $|1\rangle$. Application of the non-local C-NOT generates the two-qubit entangled state Ψ^+ in the ideal case. We analyze the resulting state by measuring the two-qubit correlators $\langle XX\rangle$, $\langle YY\rangle$ and $\langle ZZ\rangle$. The experimental results are shown in Fig. 5.5d, together with the simulated values. We then extract the state fidelity F_{Ψ^+}=(1+$\langle XX\rangle$+$\langle YY\rangle$-$\langle ZZ\rangle$)/4, where $\langle ii\rangle$ represents the measured correlator. We find a state fidelity F_{Ψ^+} = 63(4)%, in good agreement with its simulated value of F_{sim}=65%, demonstrating entanglement between the remote data qubits.

The main sources of error for the experiments in this section are the same as in the GHZ state generation. In addition, wrong assignment of the mid-circuit readout results in a wrong feed-forward operation on the data qubit and an error to the gate. To quantify the corresponding infidelity, we simulate the scenario of accepting only $|00\rangle_c$ mid-circuit readout results. We find that, in this case, the expected average fidelity for the classical truth table outcomes is 90%, while the expected entangled state fidelity reaches 76% [33], indicating that an improved readout would yield significant gains in gate performance.

3.3. Discussion and Outlook

This work demonstrates the realization of heralded genuine four-partite entanglement and the implementation of an unconditionally teleported quantum gate, adding key capabilities for solid-state quantum network testbeds that open up several new avenues. Taking the current platform as a basis, the number of data qubits per node can be further increased. In particular, the DDRF control method, integrated here with a network link, enables extension to multi-qubit control [21], enabling the generation of larger resource states that could be used, for instance, for exploring error correction on a distributed processor [34].

Another interesting direction is towards fully deterministic non-local gate operation, without imposing a timeout on the entanglement generation attempts and re-initializing the data qubits when the entanglement generation does not succeed within the timeout. This requires an active link efficiency exceeding one [35], meaning that the data qubit coherence time under network activity has to exceed the time required to generate one (or more) entangled states. The active link efficiency can be improved both by extending the data qubit coherence and by enhancing the remote entanglement generation rate. For the former, recent experiments on a weakly coupled ^{13}C nuclear spin [35] as well as on a data qubit encoded in a pair of nearby ^{13}C nuclear spins [36] promise orders of magnitude improvement in coherence under networking activity. Integrating such data qubits into non-local protocols directly benefits from the phase tracking developed here. For entanglement generation, both cavity enhancement [37, 38] and employing more efficient communication qubits [39–44] can lead to substantial rate enhancements. The techniques and methods developed in the current work can aid and accelerate the development of other communication qubits, such as the DDRF techniques pioneered on the current platform being adopted to diamond group-IV qubits [45].

Following earlier integration tests of this platform with software control layers [46, 47], the current work also impacts quantum network stack development. Both the 4-qubit GHZ resource state and the non-local gate operations expand the set of network protocols that can be explored and tested using higher layers of the stack. Scaling the number of available qubits and enabling more complex applications also opens the way to experimentally investigate optimal network synchronization and classical communication strategies, as well as network application compilers [48–50].

3.4. Methods

3.4.1. Experimental setup

The setup utilized in this work is based on the setup of Bob and Charlie nodes in Refs. [17, 18]. More details are included in the Supplementary Information.

3.4.2. Remote entanglement generation duration

As shown in Fig. 3.3a, the duration of a single remote entanglement attempt is 8.392μs. The length of a single entanglement attempt L is set by the required decoupling time τ, the duration of the reset pulse t_{reset}, and the time t necessary to reset the electron spin state. t_{reset} includes the actual on-time of the laser field and the response time of the acousto-optical modulator to make sure that the reset pulse is completely off when the first microwave pulse is applied. This constrains $L = 2\tau + t_{reset} - t$ and L must be the same for both nodes. Consequently, a free parameter for each node is τ. Given that Bob experiences a lower magnetic field compared to Alice, its minimum τ is ~3.0μs, which effectively sets the minimum allowed duration as $L \geq 2\tau$. Additionally, τ must be chosen to avoid undesired coupling to surrounding nuclear spins. As a result, for Alice the value of τ is adapted to fulfill the duration L set by Bob.

3.4.3. Qubit readout

The tomography basis-selection on the electron spin is executed via a single MW pulse with axis and angle depending on the chosen readout basis, while the optical readout is performed using long weak laser pulses (~0.1nW for up to 190μs) with a dynamical stop on the laser field when a single photon is detected. This method ensures a non-destructive readout, crucial for avoiding additional dephasing on the data qubit. Both outcomes, $|0\rangle$ and $|1\rangle$, are accepted, but for outcome $|1\rangle$, the communication qubit is afterward flipped to ensure it is always in the $|0\rangle$ state for the assisted-readout of the data qubit. During the readout of the communication qubit, the data qubits are picking up a phase depending on the outcome of the readout and its duration. This phase is also compiled in the communication qubit-assisted readout for the data qubit tomography. For the final readout on the communication qubit, after mapping the state of the data qubit on it, we use a shorter and higher power pulse (~1nW for up to 40μs), without dynamical stop.

3.4.4. GHZ STATE FIDELITY

The 4-qubit GHZ state fidelity, provided that the measured correlators C_i signs are in accordance with the expected state, is calculated as:

$$F_{GHZ} = \frac{\sum_{i=1}^{16} |C_i|}{16} \tag{3.2}$$

while the error is propagated as:

$$\sigma_{GHZ} = \left(\sqrt{\sum_{i=2}^{16} \sigma_{C_i}^2 + 2 \cdot \sum_{i=2}^{8} \sum_{\substack{j=2 \\ j \neq i}}^{8} Cov(C_i, C_j)} \right) / 16 \tag{3.3}$$

where the covariance term takes into account the full correlations among the Z terms, as they are directly extracted from the $\langle ZZZZ \rangle$ measurement.

3.5. SUPPLEMENTARY INFORMATION

3.5.1. EXPERIMENTAL SETUP & OPERATIONS

The experimental setup for Alice and Bob is similar; hence a single common description is provided here. The NV center platform is composed of a type IIa chemical vapor deposition diamond, cut along the $\langle 111 \rangle$ crystal orientation (Element Six), where Solid Immersion Lenses are fabricated around single defects to improve the collection efficiency together with anti-reflection coating. Lithographically deposited gold on the diamond surface acts as a stripline for microwave, DC voltage and radio-frequency delivery. The sample is mounted on a PCB and placed on a sample holder in a closed-cycle cryostat (Montana Cryostation). In the back of the sample holder, a static neodymium magnet is inserted. Additional magnets for magnetic field alignment purposes are placed outside the sample chamber at room temperature. Optical access to the diamond sample is obtained with a room-temperature confocal microscope objective that is mounted on a three-axis piezo stage. A detailed schematic of the optics used for excitation and collection can be found in Ref. [17].

The negatively-charged state of the NV-center is a spin-1 system, whose ground state is fully non-degenerate in the presence of an external magnetic field [51]. In Fig.3.6, we include a schematic of the optical and microwave transitions that are relevant for this work. To achieve photon indistinguishability, a DC voltage (range ±15V) is applied to exploit the DC Stark effect that effectively tunes the optical transitions and brings the transition m$_s$=0→ $E_{x/y}$ of the two nodes in resonance with each other at 470.4550THz. Spectral wandering over time is compensated by a Proportional-Integral-Derivative control loop on the applied DC voltage, whose error signal is computed on the average photon counts during the Charge-Resonance check.

The excitation of the optical transitions stimulates the NV to emit single photons (Zero-Phonon Line) or photons+phonons (Phonon-Side Band) according to the Debye-Waller factor. The ZPL photons are used to generate remote entanglement. Using narrow-band filters and cross-polarization techniques, the ZPL photons are separated from the

PSB and the excitation light and directed towards the midpoint. Photon detection is achieved via Superconducting Nanowire Single Photon Detectors (PhotonSpot), connected to the output ports of a 50:50 (effectively measured 45:55) in-fiber beam splitter, and show a dark count rate ≤1Hz each. The PSB is used to read out the qubit state in single-shot mode by state-dependent excitation and discriminating on whether zero or non-zero PSB photons were detected. The detection is achieved using an Avalanche PhotoDiode at each node (Laser Components, Count FC 10C/20C) that shows a dark count rate of 15Hz for Alice and 6Hz for Bob.

The transitions denoted with "Reset" are used to initialize the qubit state in $|0\rangle$. In Alice, these two transitions are separated by 480MHz, hence we use two separate red lasers (Toptica TA-SHG and DL Pro) to address each one of them and achieve an efficient reset process. For Bob, the separation is efficiently covered by the power broadening of a single laser pulse parked in the middle of the two transitions.

To keep the NV in the desired charge state (NV^-), a recharging mechanism is needed. In this case, we exploit a two-photon process when addressing the ZPL transition of the neutral charge state (NV^0) [52], that deterministically ionizes to NV^-. For this we use a single laser per node around 575nm (Toptica DL-SHG pro). For the high-magnetic field setup, Alice, the frequency splitting between the relevant NV^0 transitions is ~200MHz [53], and also in this case we exploit the power broadening of the pulse to effectively address both transitions simultaneously. Typical power values utilized for effective recharging are 400nW (30nW) for Alice (Bob) for hundreds of μs.

The single-qubit gate on the electron spin state is performed by applying microwave pulses to the spin transitions denoted with the purple cycle in Fig. 3.6. The microwave signals' source is provided by the R&S SGS100A and is IQ-modulated via the Zurich Instruments HDAWG. The signal is amplified up to 42W (20W) (AR 40S1G4) before reaching the sample for Alice (Bob).

The HDAWG is used for nanosecond-precision signals and part of the experimental logic. On a higher level, the experimental sequences and logic are orchestrated by a multi-module microcontroller unit (Jäger ADwin-Pro II T12), including the classical information exchange between the nodes via the TiCo module.

For the DDRF method, the RF signal is generated by the HDAWG and amplified before being mixed with the microwave signal and delivered to the chip. The RF signal is a square pulse, whose rise and fall transitions incorporate a $\sin^2(t)$ signal to reduce transient oscillations. Experimental details and theoretical considerations on the DDRF method can be found in Ref. [22].

3.5.2. CALIBRATION ROUTINE

Before being able to run the experiments described in the main text, it is necessary to prepare the setup and calibrate the relevant parameters for each qubit. In this section, we describe the general calibration routine, differentiating whether the calibration is tar-

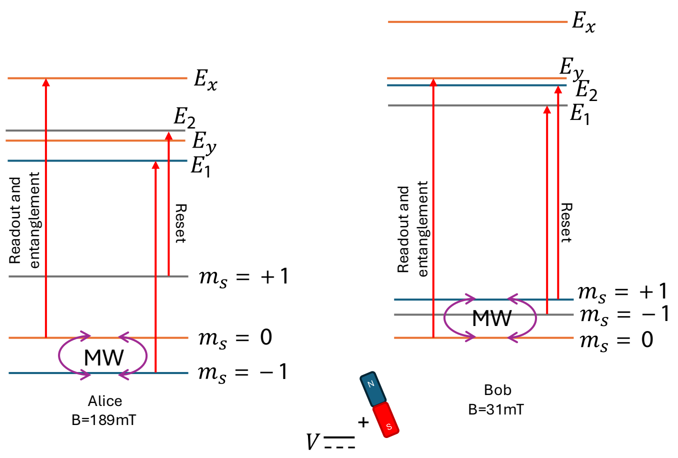


Figure 3.6: Energy level diagram (not in scale). Alice and Bob are biased at different magnetic fields resulting in different energy splittings in the electronic ground state. In Alice's case, the m_s=-1 state crossed the m_s=0 state and becomes the lowest energy state. The crossing happens at 100mT. The excited states are tuned via an external DC field, ensuring that the m_s=0→E$_x$ transition of Alice has the same frequency as m_s=0→E$_y$ of Bob.

geted at the physical setup, at the electron spin qubit or the nuclear spin qubit. The calibration routine and experimental sequences are backed up by the QMI software package [54].

SETUP CALIBRATION

The setup calibration starts with the calibration of the laser power levels, by sweeping the amplitude of the RF tone that drives the acousto-optical modulators for each laser and detecting with a power meter the corresponding laser power at the setup. Subsequently, we calibrate the position of the microscope objective with respect to the emitter. To do so, we use green laser photoluminescence, collecting the emitted PSB photons when scanning the objective position along the three axes.

Other relevant setup calibrations regard the cross-polarization alignment to ensure that the laser photons are properly rejected in the ZPL collection path. For this, a set of automatized half and quarter waveplates placed in the ZPL path is scanned. The optimal position corresponds to the minimum amount of photon count rate in the SNSPDs when the resonant red laser is on. A similar procedure is followed to enable homodyne interference between the two setups at the mid-point in the global phase stabilization scheme. Namely, a motorized half-waveplate at each setup ensures that the same amount of coherent light is sent from each node to the midpoint. Details on the phase stabilization setup and procedures are explained in Ref. [17].

ELECTRON SPIN CALIBRATION

Provided that the NV is in the right charge state and the laser frequencies are on resonance with the relevant transitions (validated via the Charge-Resonance check), the calibration starts with the microwave pulses for the single-qubit gates on the communication qubit. The summary of the calibrated parameters with their typical values is inserted in Table 3.1. To obtain an arbitrary rotation along a specific axis (α-pulse), apart from the $\pi/2$ rotation, we use the same duration of the π pulse and reduce the amplitude accordingly to the desired angle of rotation.

Next, the routine focuses on the calibration of the remote entanglement generation parameters, which are summarized in Table 3.2, additionally including the optical phase stabilization and the entangled state phase measurement, whose values change over time due to setup alignment and ambient conditions.

Parameter	**Alice**	**Bob**
Frequency	2.414GHz	3.733GHz
Power	42W	20W
π Duration	215ns	205ns
π Amplitude (fraction)	0.88	0.94
π Skewness	$9.85 \cdot 10^{-9}$	$-3.34 \cdot 10^{-9}$
π: $P(\|0\rangle)$ after 7 pulses	4%	0.5%
$\pi/2$ Duration	150ns	135ns
$\pi/2$ Amplitude (fraction)	0.40	0.52
$\pi/2$ Skewness	$1.28 \cdot 10^{-8}$	$-5.24 \cdot 10^{-9}$
$\pi/2$: $P(\|0\rangle)$ after 6 pulses	2%	0.5%

Table 3.1: Relevant parameters for the calibration of the microwave pulses. The microwave pulse shape is a skewed Hermite pulse. The amplitude is reported as a fraction of the maximum output voltage of the IQ modulation channels.

Parameter	**Alice**	**Bob**
Counts per shot p	$0.9 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$
Bright state population α	0.06	0.03
Entanglement attempt duration		8.392μs
Detection window		7ns

Table 3.2: Remote entanglement relevant parameters. The α_B and p parameters reported are typical values, as they can fluctuate based on external conditions. Particularly, α_B is set to fulfill the expression $p_A\alpha_A = p_B\alpha_B$. The ratio α_B/α_A is in the range 0.5±0.1.

NUCLEAR SPIN CALIBRATION

The third part of the calibration focuses on the data qubit, namely the control of single nuclear spins. Despite the use of two different methods for the control, the routine is very similar. The preliminary step is to identify a well-isolated ^{13}C. In the case of DD method, this is obtained by sweeping the interpulse delay in a repeated XY8 sequence, when the electron spin is initialized in a superposition state. Interpulse delays τ that are on resonance with a single nuclear spin result in a coherent inversion of the electron spin state [20]. For the selected nuclear spin, we obtain τ=12.452μs. For the DDRF method, we sweep the frequency ω_{RF} of the RF field, while the repeated XY8 sequence has a fixed interpulse delay of τ=21.8μs.

Once the target nuclear spin is selected, we can calibrate the conditional and unconditional gates. This is achieved by tuning the amount of XY8 pulses. For the DD case, we obtain N_{cond}^{DD}=48, while for the DDRF the number of pulses can be tuned for time and synchronization reasons by changing the amplitude of the driving RF field, with an upper limit set by heating.

As the electron spin dynamics affect the precession of the nuclear spin due to the hyperfine interaction, it is necessary for effective control to characterize these precession frequencies. For the DD method (Bob), these frequencies can be extrapolated from a detuned Ramsey-type experiment using the nuclear spin initialized in a superposition state and the electron spin in an eigenstate. From fitting the data, we can extrapolate the frequencies and the T_2^* value. The values of the precession frequencies for the two possible electron spin eigenstates are then used to calculate the phase that the nuclear spin state picks up under the electron spin dynamics, provided that it is known how much time the electron spin spends in such states.

In the case of DDRF (Alice), we use a Ramsey-type experiment with electron spin in $|0\rangle$ to determine the precession frequency ω_0 of the nuclear spin around the $\hat{z}$ axis. When the electron spin is in $|1\rangle$, the nuclear spin is driven by the RF field in the $\hat{x}$-$\hat{y}$ plane. The Ramsey experiments are shown in Fig. 3.7, while the results are summarized in Table 3.3.

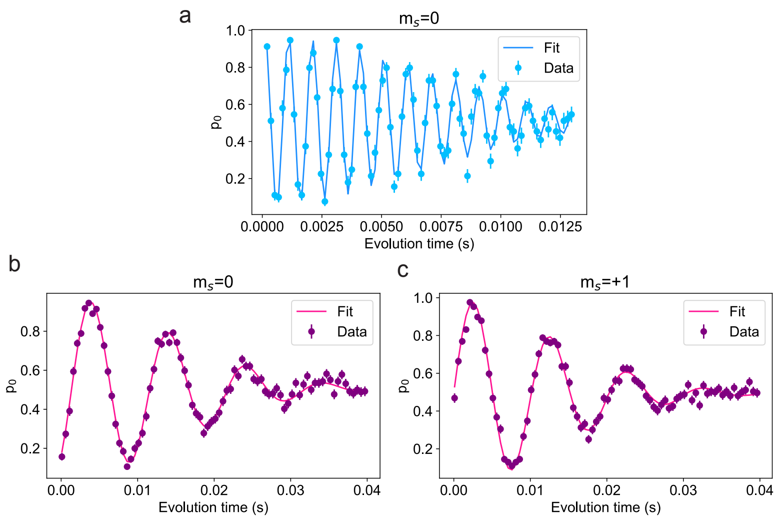


Figure 3.7: Ramsey measurement. a) Experiment for Alice nuclear spin when the electron spin is initialized in $|0\rangle$. From the fit, we obtain ω_0=2.021MHz. b) Experiment for Bob nuclear spin with the electron spin in $|0\rangle$. The resulting frequency is ω_0=327.1kHz. In c) we report the experiment when the electron spin is in $|1\rangle$, resulting in ω_1=355.6kHz.

Node	$A_\parallel$ ($\times 2\pi$)	$A_\perp$ ($\times 2\pi$)	T_2^* (avg)	ω_0	ω_1
Alice	-30.0kHz	~ 0	9.4(4)ms	2.021MHz	ω_{RF}=2.051MHz
Bob	28.2kHz	11.9kHz	19.6(5)ms	327.1kHz	355.6kHz

Table 3.3: Nuclear spin characteristic parameters. For DDRF, $A_\perp \ll \omega_{RF}$, hence we neglect this term. In the reference frame of the RF field, $\omega_0 = A_\parallel = \omega_L - \omega_{RF}$ and $\omega_1 = 0$. The relative error on the obtained frequencies is 0.5% for Alice and 0.2% for Bob [33].

3.5.3. NUCLEAR SPIN PHASE EVOLUTION DURING ENTANGLEMENT ATTEMPTS

A separate discussion is needed on the evolution of the nuclear spin during network activity. During an entanglement attempt, and specifically during the reset pulse, the electron spin state undergoes a stochastic process, given that the exact moment when it flips back to $|0\rangle$ is probabilistic. From the nuclear spin perspective, this results in a dephasing mechanism. As reported in the main text, the phase that the nuclear spin acquires during the entanglement attempts needs a proper separate calibration. The resulting phase per entanglement attempt can be seen as an average phase due to the stochasticity of the process. To calibrate such a phase, we follow two different protocols due to the use of two different control techniques.

For the DDRF setup (Alice), the calibration process is faster, since the phase acquired during the entanglement attempt is fed to the local oscillator of the RF field and used to update the phase of the next RF pulse. We first characterize a pre-entanglement global phase that ensures that without any entanglement attempts, the nuclear spin is correctly rephased for the readout measurement. This phase is independent of the initial state of the nuclear spin, so for consistency, we initialize it in the $|X\rangle$ state. Subsequently, we can characterize the single entanglement attempt phase. To do so, we initialize the nuclear spin state in $|X\rangle$, sweep the number of entanglement attempts (e.g. from 1 to 25), and then measure in the X basis. Given that the local RF oscillator was not updated, we obtain a sine-type signal, from which we can extract the average phase for a single entanglement attempt. A typical value for the phase of a single attempt is 54°.

For the DD setup (Bob), the rephasing is executed via a tailored XY8 sequence. The calibration of such sequences comprises several steps. First of all, we compile a table of interpulse delays for the XY8 sequence where we ensure that no coupling to surrounding nuclear spins is involved (1% tolerance on the electron spin coherence loss). A typical range for the interpulse delay is [2.8μs-3.2μs]. The next step includes finding the optimal rephasing interpulse delay for a certain number of entanglement attempts. For this, we first initialize the nuclear spin in $|X\rangle$, we sweep the number of entanglement attempts (e.g. from 1 to 10) and for each number of entanglement attempts we sweep the total duration of the rephasing XY8 sequence by using the precompiled table of optimal interpulse delays, and finally we measure the nuclear spin in the X basis. For each number of entanglement attempts, we obtain a sine-like signal over the interpulse delays. We jointly fit these curves by imposing the same frequency as a fit parameter, and from that we extract the phase acquired for each number of entanglement attempts. In the next step, we fit the obtained phases with a linear function to extrapolate the general phase rule for N number of entanglement attempts, bounded between 0 and 2π. We then convert the phase into an XY8 duration knowing the evolution frequency, and we compile the corresponding interpulse delays, chosen among those that pass the non-coupling check, into a look-up table in the HDAWG that can be used in real-time during the experiment. In this method, the main source of errors comes from the fit error and from the necessity of using a discrete set of XY8 durations, while for the DDRF method the only source of error is the curve fit.

3.5.4. READOUT CORRECTION ON NUCLEAR SPIN STATE MAPPING

As illustrated in the main text, the readout of the nuclear spin state is assisted by mapping such a state into the electron spin state. Hence, when reading out, infidelity is caused by the known tomography errors during the single-shot readout of the electron spin and the errors that occur during the mapping of the nuclear spin state into the electron spin state. To estimate and correct for the latter, we adopt a combination of the strategies reported in Refs. [55, 56]. During the mapping, the electron spin is subjected, among other sources of errors, to dephasing that is faster than the optimal read-out time, measured in number of microwave pulses N_{RO} necessary to complete the mapping. To characterize this dephasing, we perform the experiment displayed in Fig. 3.8a, during which the target nuclear spin is left uninitialized, but the interaction with the electron spin is activated via the repeated XY8 sequence similar to that used for the electron-nuclear conditional gate. Hence, in the case of Alice, this is also interleaved with RF pulses. The result is a damped oscillation in the number of XY8 repetitions due to repeated entangling and disentangling of the electron with the nuclear spin, displayed in Figs. 3.8a-b. Doing the calibration this way we avoid the introduction of additional errors due to the initialization process of the nuclear spin state, which is also assisted by the electron spin, separating the readout sequence from it. An imperfect initialization process can, in principle, lead to correct readout results, as the mapping process is not symmetric and, therefore, the readout might compensate for incorrect initialization and, at the same time, generate a correlated error on the electron spin, obscuring the dephasing only given by the readout. We fit this curve to the function:

$$\langle\sigma_y^e\rangle(x,\delta,d,N_0,\beta) = \delta\exp[-(x/N_0)^d]\cos(\beta x) \tag{3.4}$$

in which δ represents the maximum contrast achieved by the signal, N$_0$ and n characterize the exponential decay due to the dephasing of the electron spin; the cosine function represents the oscillating behaviour that the signal should have under perfect conditions. The parameter β refers to the electron-nuclear coupling. From this, it is possible to extract the correction C_{en} defined as:

$$C_{en} = \delta\exp[-(N_{RO}/N_0)^d]\sin(\beta N_{RO}) \tag{3.5}$$

that we use to rescale the single-shot readout corrected expectation values obtained from the electron-assisted nuclear spin tomography as $1/C_{en}$. We obtain correction factors of $1/C_{en}^{Alice}$=1.08(2) and $1/C_{en}^{Bob}$=1.05(3).

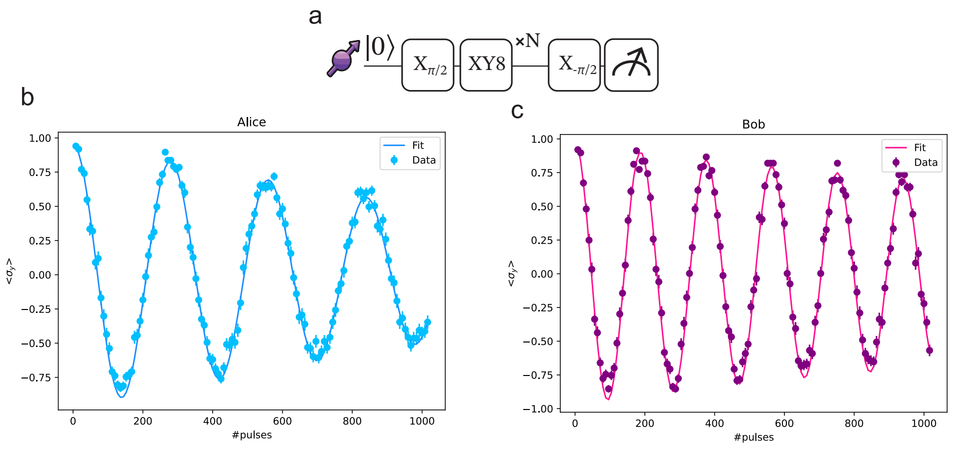


Figure 3.8: Nuclear spin readout error characterization. a) Experimental sequence executed on the electron spin to isolate the assisted-readout errors from any nuclear spin initialization imperfections. For Alice, the inter-pulse delay is filled with RF pulses on resonance with the target qubit. b) and c) display the recorded signal and the corresponding fit to extract the relevant parameters for the computation of the correcting factors. In b), we obtain the following parameters from the curve fitting: δ_{Alice}=0.95(1), N_0^{Alice}=1442(97), d_{Alice}=1.2(1), β_{Alice}=0.0224(3). For c) we obtain: δ_{Bob}=0.98(2), N_0^{Bob}=3495(954), d_{Bob}=0.9(2), β_{Bob}=0.0334(2)

3.5.5. EXPERIMENT SIMULATIONS

The simulated outcomes of the two experiments can be found at [33]. For the simulation of the remote entangled state, the simulation is adapted from [18].
The GHZ experiment simulation includes errors from the dephasing on the data qubits from the generation of the remote entangled state; the depolarization of the communication qubits after entangling with their local data qubit; the dephasing on the data qubit caused by wrong readout assignment of the communication qubit.
The simulation for the non-local C-NOT gate includes the same errors of the GHZ case, with the exception of the last dephasing error, which is substituted with the incorrect feedback operation on the data qubit corresponding to the incorrect readout assignment probabilities.

3.5.6. DATA ACQUISITION

The setup can be fully operated remotely. For the two network experiments, data are acquired in batches of 1h, interleaved with partial calibration of the setup. The partial calibration is focused on the entanglement generation parameters, particularly the measurement of the phase of the entangled state and the optimal cross-polarization point. These parameters are affected by small drifts in the optical setup that are mainly due to the degradation of the vacuum of the sample chamber (leading to the formation of layers of ice), as well as due to vibrations, temperature and humidity fluctuations of the laboratory.

The average experimental rate is in the range of (23-42)mHz, with a total number of data points of: 360 for the GHZ experiment, 400 for the classical truth table of the C-NOT gate and 234 points for the creation of the remote entangled state via the non-local C-NOT gate. The variation in the experimental rate is due to the daily fluctuations in counts per shot of the two NVs, which directly affect the rate of entanglement generation, and to the charge fluctuations due to the DC Stark tuning, which affect the number of CR checks required to bring both nodes on resonance with each other, increasing experiment overhead time. Besides the rate, such fluctuations directly affect the maximum achievable fidelity. During entanglement generation, for all experiments, we keep the bright state population parameter α_A of Alice fixed at 0.06, while α_B of Bob is adapted to fulfill the equality $p_A\alpha_A = p_B\alpha_B$. However, during the experiment, such conditions might not be fulfilled at all times. The simulations do not take this variation into account. On the other hand, variations in the DC field necessary to keep both nodes at the same resonance frequency during the entanglement attempts affect the overall indistinguishability of the single photons, and therefore the fidelity.

REFERENCES

[1] H. J. Kimble. "The quantum internet". In: *Nature* 453.7198 (June 2008), pp. 1023–1030. DOI: 10.1038/nature07127.

[2] S. Wehner, D. Elkouss, and R. Hanson. "Quantum internet: A vision for the road ahead". In: *Science* 362.6412 (Oct. 2018). DOI: 10.1126/science.aam9288.

[3] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti. "Distributed quantum computing: A survey". In: *Computer Networks* 254 (Dec. 2024), p. 110672. DOI: 10.1016/j.comnet.2024.110672.

[4] D. Barral et al. "Review of Distributed Quantum Computing: From single QPU to High Performance Quantum Computing". In: *Computer Science Review* 57 (Aug. 2025), p. 100747. DOI: 10.1016/j.cosrev.2025.100747.

[5] J. I. Cirac, A. K. Ekert, S. F. Huelga, and C. Macchiavello. "Distributed quantum computation over noisy channels". In: *Physical Review A* 59.6 (June 1999). Publisher: American Physical Society, pp. 4249–4254. DOI: 10.1103/PhysRevA.59.4249.

[6] J. Eisert, K. Jacobs, P. Papadopoulos, and M. B. Plenio. "Optimal local implementation of nonlocal quantum gates". In: *Physical Review A* 62.5 (Oct. 2000). Publisher: American Physical Society, p. 052317. DOI: 10.1103/PhysRevA.62.052317.

[7] Y.-F. Huang, X.-F. Ren, Y.-S. Zhang, L.-M. Duan, and G.-C. Guo. "Experimental Teleportation of a Quantum Controlled-NOT Gate". In: *Physical Review Letters* 93.24 (Dec. 2004), p. 240501. DOI: 10.1103/PhysRevLett.93.240501.

[8] W.-B. Gao et al. "Teleportation-based realization of an optical quantum two-qubit entangling gate". In: *Proceedings of the National Academy of Sciences* 107.49 (Dec. 2010), pp. 20869–20874. DOI: 10.1073/pnas.1005720107.

[9] S. Daiss, S. Langenfeld, S. Welte, E. Distante, P. Thomas, L. Hartung, O. Morin, and G. Rempe. "A quantum-logic gate between distant quantum-network modules". In: *Science* 371.6529 (Feb. 2021), pp. 614–617. DOI: 10.1126/science.abe3150.

[10] X. Liu et al. "Nonlocal photonic quantum gates over 7.0 km". In: *Nature Communications* 15.1 (Oct. 2024). Publisher: Nature Publishing Group, p. 8529. DOI: 10.1038/s41467-024-52912-3.

[11] Y.-C. Wei et al. "Universal distributed blind quantum computing with solid-state qubits". In: *Science* 388.6746 (May 2025), pp. 509–513. DOI: 10.1126/science.adu6894.

[12] K. S. Chou, J. Z. Blumoff, C. S. Wang, P. C. Reinhold, C. J. Axline, Y. Y. Gao, L. Frunzio, M. H. Devoret, L. Jiang, and R. J. Schoelkopf. "Deterministic teleportation of a quantum gate between two logical qubits". In: *Nature* 561.7723 (Sept. 2018). Publisher: Nature Publishing Group, pp. 368–373. DOI: 10.1038/s41586-018-0470-y.

[13] J. Qiu et al. "Deterministic quantum state and gate teleportation between distant superconducting chips". In: *Science Bulletin* 70.3 (Feb. 2025), pp. 351–358. DOI: 10.1016/j.scib.2024.11.047.

[14] Y. Wan et al. “Quantum gate teleportation between separated qubits in a trapped-ion processor”. In: *Science* 364.6443 (May 2019), pp. 875–878. DOI: 10.1126/science.aaw9415.

[15] D. Main, P. Drmota, D. P. Nadlinger, E. M. Ainley, A. Agrawal, B. C. Nichol, R. Srinivas, G. Araneda, and D. M. Lucas. “Distributed quantum computing across an optical network link”. In: *Nature* 638.8050 (Feb. 2025). Publisher: Nature Publishing Group, pp. 383–388. DOI: 10.1038/s41586-024-08404-x.

[16] A. J. Stolk et al. “Metropolitan-scale heralded entanglement of solid-state qubits”. In: *Science Advances* 10.44 (Nov. 2024), eadp6442. DOI: 10.1126/sciadv.adp6442.

[17] M. Pompili et al. “Realization of a multinode quantum network of remote solid-state qubits”. In: *Science* 372.6539 (Apr. 2021). Publisher: American Association for the Advancement of Science, pp. 259–264. DOI: 10.1126/science.abg1919.

[18] S. L. N. Hermans, M. Pompili, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson. “Qubit teleportation between non-neighbouring nodes in a quantum network”. In: *Nature* 605.7911 (May 2022), pp. 663–668. DOI: 10.1038/s41586-022-04697-y.

[19] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. W. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson. “Entanglement distillation between solid-state quantum network nodes”. In: *Science* 356.6341 (June 2017), pp. 928–932. DOI: 10.1126/science.aan0070.

[20] T. H. Taminiau, J. J. T. Wagenaar, T. van der Sar, F. Jelezko, V. V. Dobrovitski, and R. Hanson. “Detection and Control of Individual Nuclear Spins Using a Weakly Coupled Electron Spin”. In: *Physical Review Letters* 109.13 (Sept. 2012), p. 137602. DOI: 10.1103/PhysRevLett.109.137602.

[21] C. E. Bradley, J. Randall, M. H. Abobeih, R. C. Berrevoets, M. J. Degen, M. A. Bakker, M. Markham, D. J. Twitchen, and T. H. Taminiau. “A Ten-Qubit Solid-State Spin Register with Quantum Memory up to One Minute”. In: *Physical Review X* 9.3 (Sept. 2019), p. 031045. DOI: 10.1103/PhysRevX.9.031045.

[22] H. van Ommen, G. van de Stolpe, N. Demetriou, H. Beukers, J. Yun, T. Fortuin, M. Iuliano, A.-P. Montblanch, R. Hanson, and T. Taminiau. “Improved Electron-Nuclear Quantum Gates for Spin Sensing and Control”. In: *PRX Quantum* 6.2 (Apr. 2025). Publisher: American Physical Society, p. 020309. DOI: 10.1103/PRXQuantum.6.020309.

[23] L. Robledo, L. Childress, H. Bernien, B. Hensen, P. F. A. Alkemade, and R. Hanson. “High-fidelity projective read-out of a solid-state spin quantum register”. In: *Nature* 477.7366 (Sept. 2011). Publisher: Nature Publishing Group, pp. 574–578. DOI: 10.1038/nature10401.

[24] N. Zhao et al. “Sensing single remote nuclear spins”. In: *Nature Nanotechnology* 7.10 (Oct. 2012), pp. 657–662. DOI: 10.1038/nnano.2012.152.

[25] S. Kolkowitz, Q. P. Unterreithmeier, S. D. Bennett, and M. D. Lukin. “Sensing Distant Nuclear Spins with a Single Electron Spin”. In: *Physical Review Letters* 109.13 (Sept. 2012). Publisher: American Physical Society, p. 137601. DOI: 10.1103/PhysRevLett.109.137601.

[26] P. Tamarat et al. “Stark Shift Control of Single Optical Centers in Diamond”. In: *Physical Review Letters* 97.8 (Aug. 2006), p. 083002. DOI: 10.1103/PhysRevLett.97.083002.

[27] C. Cabrillo, J. I. Cirac, P. García-Fernández, and P. Zoller. “Creation of entangled states of distant atoms by interference”. In: *Physical Review A* 59.2 (Feb. 1999), pp. 1025–1033. DOI: 10.1103/PhysRevA.59.1025.

[28] S. Bose, P. L. Knight, M. B. Plenio, and V. Vedral. “Proposal for Teleportation of an Atomic State via Cavity Decay”. In: *Physical Review Letters* 83.24 (Dec. 1999), pp. 5158–5161. DOI: 10.1103/PhysRevLett.83.5158.

[29] S. L. N. Hermans, M. Pompili, L. D. Santos Martins, A. R-P Montblanch, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson. “Entangling remote qubits using the single-photon protocol: an in-depth theoretical and experimental study”. In: *New Journal of Physics* 25.1 (Jan. 2023), p. 013011. DOI: 10.1088/1367-2630/acb004.

[30] N. Kalb, P. C. Humphreys, J. J. Slim, and R. Hanson. “Dephasing mechanisms of diamond-based nuclear-spin memories for quantum networks”. In: *Physical Review A* 97.6 (June 2018), p. 062330. DOI: 10.1103/PhysRevA.97.062330.

[31] M. Ghaderibaneh, H. Gupta, and C. Ramakrishnan. “Generation and Distribution of GHZ States in Quantum Networks”. In: *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Bellevue, WA, USA: IEEE, Sept. 2023, pp. 1120–1131. DOI: 10.1109/QCE57702.2023.00127.

[32] O. Gühne and G. Tóth. “Entanglement detection”. In: *Physics Reports* 474.1-6 (Apr. 2009), pp. 1–75. DOI: 10.1016/j.physrep.2009.02.004.

[33] M. Iuliano, N. Demetriou, H. B. Van Ommen, T. H. Taminiau, and R. Hanson. *Data underlying the publication "Unconditionally teleported quantum gates between remote solid-state qubit registers"*. 2025. DOI: 10.4121/a33310e8-a19b-4aac-8057-37ab1363e42e.

[34] S. De Bone, P. Möller, C. E. Bradley, T. H. Taminiau, and D. Elkouss. “Thresholds for the distributed surface code in the presence of memory decoherence”. In: *AVS Quantum Science* 6.3 (Sept. 2024), p. 033801. DOI: 10.1116/5.0200190.

[35] C. E. Bradley et al. “Robust quantum-network memory based on spin qubits in isotopically engineered diamond”. In: *npj Quantum Information* 8.1 (Oct. 2022), p. 122. DOI: 10.1038/s41534-022-00637-w.

[36] H. P. Bartling, M. H. Abobeih, B. Pingault, M. J. Degen, S. J. H. Loenen, C. E. Bradley, J. Randall, M. Markham, D. J. Twitchen, and T. H. Taminiau. “Entanglement of Spin-Pair Qubits with Intrinsic Dephasing Times Exceeding a Minute”. In: *Physical Review X* 12.1 (Mar. 2022), p. 011048. DOI: 10.1103/PhysRevX.12.011048.

[37] D. Riedel, I. Söllner, B. J. Shields, S. Starosielec, P. Appel, E. Neu, P. Maletinsky, and R. J. Warburton. "Deterministic Enhancement of Coherent Photon Generation from a Nitrogen-Vacancy Center in Ultrapure Diamond". In: *Physical Review X* 7.3 (Sept. 2017), p. 031040. DOI: 10.1103/PhysRevX.7.031040.

[38] J. Fischer, Y. Herrmann, C. F. J. Wolfs, S. Scheijen, M. Ruf, and R. Hanson. "Spin-photon correlations from a Purcell-enhanced diamond nitrogen-vacancy center coupled to an open microcavity". In: *Nature Communications* 16.1 (Nov. 2025), p. 11680. DOI: 10.1038/s41467-025-66722-8.

[39] M. Atatüre, D. Englund, N. Vamivakas, S.-Y. Lee, and J. Wrachtrup. "Material platforms for spin-based photonic quantum technologies". In: *Nature Reviews Materials* 3.5 (Apr. 2018), pp. 38–51. DOI: 10.1038/s41578-018-0008-9.

[40] M. Ruf, N. H. Wan, H. Choi, D. Englund, and R. Hanson. "Quantum networks based on color centers in diamond". In: *Journal of Applied Physics* 130.7 (Aug. 2021), p. 070901. DOI: 10.1063/5.0056534.

[41] P.-J. Stas et al. "Robust multi-qubit quantum network node with integrated error detection". In: *Science* 378.6619 (Nov. 2022), pp. 557–560. DOI: 10.1126/science.add9771.

[42] A. E. Rugar, S. Aghaeimeibodi, D. Riedel, C. Dory, H. Lu, P. J. McQuade, Z.-X. Shen, N. A. Melosh, and J. Vučković. "Quantum Photonic Interface for Tin-Vacancy Centers in Diamond". In: *Physical Review X* 11.3 (July 2021), p. 031021. DOI: 10.1103/PhysRevX.11.031021.

[43] R. A. Parker et al. "A diamond nanophotonic interface with an optically accessible deterministic electronuclear spin register". In: *Nature Photonics* 18.2 (Feb. 2024), pp. 156–161. DOI: 10.1038/s41566-023-01332-8.

[44] M. Pasini et al. "Nonlinear Quantum Photonics with a Tin-Vacancy Center Coupled to a One-Dimensional Diamond Waveguide". In: *Physical Review Letters* 133.2 (July 2024), p. 023603. DOI: 10.1103/PhysRevLett.133.023603.

[45] H. K. C. Beukers et al. "Control of Solid-State Nuclear Spin Qubits Using an Electron Spin- 1 / 2". In: *Physical Review X* 15.2 (Apr. 2025), p. 021011. DOI: 10.1103/PhysRevX.15.021011.

[46] M. Pompili et al. "Experimental demonstration of entanglement delivery using a quantum network stack". In: *npj Quantum Information* 8.1 (Oct. 2022), p. 121. DOI: 10.1038/s41534-022-00631-2.

[47] C. Delle Donne et al. "An operating system for executing applications on quantum network nodes". In: *Nature* 639.8054 (Mar. 2025), pp. 321–328. DOI: 10.1038/s41586-025-08704-w.

[48] S. Oslovich, B. v. d. Vecht, and S. Wehner. *Compilation strategies for quantum network programs using Qoala*. arXiv:2505.06162 [quant-ph]. May 2025. DOI: 10.48550/arXiv.2505.06162.

[49] T. R. Beauchamp, H. Jirovská, S. Gauthier, and S. Wehner. *A Modular Quantum Network Architecture for Integrating Network Scheduling with Local Program Execution*. arXiv:2503.12582 [quant-ph]. Mar. 2025.

[50] J. Miguel-Ramiro, J. Illiano, F. Mazza, A. Pirker, J. Freund, A. S. Cacciapuoti, M. Caleffi, and W. Dür. *QPing: a Quantum Ping Primitive for Quantum Networks*. arXiv:2508.03806 [quant-ph]. Aug. 2025. DOI: 10.48550/arXiv.2508.03806.

[51] M. W. Doherty, N. B. Manson, P. Delaney, and L. C. L. Hollenberg. "The negatively charged nitrogen-vacancy centre in diamond: the electronic solution". In: *New Journal of Physics* 13.2 (Feb. 2011), p. 025019. DOI: 10.1088/1367-2630/13/2/025019.

[52] P. Siyushev, H. Pinto, M. Vörös, A. Gali, F. Jelezko, and J. Wrachtrup. "Optically Controlled Switching of the Charge State of a Single Nitrogen-Vacancy Center in Diamond at Cryogenic Temperatures". In: *Physical Review Letters* 110.16 (Apr. 2013), p. 167402. DOI: 10.1103/PhysRevLett.110.167402.

[53] S. Baier, C. E. Bradley, T. Middelburg, V. V. Dobrovitski, T. H. Taminiau, and R. Hanson. "Orbital and Spin Dynamics of Single Neutrally-Charged Nitrogen-Vacancy Centers in Diamond". In: *Physical Review Letters* 125.19 (Nov. 2020), p. 193601. DOI: 10.1103/PhysRevLett.125.193601.

[54] I. T. Raa et al. *QMI - Quantum Measurement Infrastructure, a Python 3 framework for controlling laboratory equipment*. Oct. 2023.

[55] J. Cramer, N. Kalb, M. A. Rol, B. Hensen, M. S. Blok, M. Markham, D. J. Twitchen, R. Hanson, and T. H. Taminiau. "Repeated quantum error correction on a continuously encoded qubit by real-time feedback". In: *Nature Communications* 7.1 (May 2016), p. 11526. DOI: 10.1038/ncomms11526.

[56] J. Randall, C. E. Bradley, F. V. van der Gronden, A. Galicia, M. H. Abobeih, M. Markham, D. J. Twitchen, F. Machado, N. Y. Yao, and T. H. Taminiau. "Many-body–localized discrete time crystal with a programmable spin-based quantum simulator". In: *Science* 374.6574 (Dec. 2021), pp. 1474–1478. DOI: 10.1126/science.abk0603.

4

Qubit teleportation between a memory-compatible photonic time-bin qubit and a solid-state quantum network node

I mean, you could claim that anything's real if the only basis for believing in it is that nobody's proved it doesn't exist!

J.K. Rowling - Harry Potter and the Deathly Hallows

M. Iuliano*, M.-C. Slater*, A. J. Stolk, M. J. Weaver, T. Chakraborty, E. Loukiantchenko, G. C. do Amaral, N. Alfasi, M. O. Sholkina, W. Tittel, R. Hanson

We report on a quantum interface linking a diamond NV center quantum network node and 795nm photonic time-bin qubits compatible with Thulium and Rubidium quantum memories. The interface makes use of two-stage low-noise quantum frequency conversion and waveform shaping to match temporal and spectral photon profiles. Two-photon quantum interference shows high indistinguishability between converted 795nm photons and the native NV center photons. We use the interface to demonstrate quantum teleportation including real-time feedforward from an unbiased set of 795nm photonic qubit

The results of this chapter are published in npj Quantum Information, 10 (2024).
* Equally contributing authors.

input states to the NV center spin qubit, achieving a teleportation fidelity well above the classical bound. This proof-of-concept experiment shows the feasibility of interconnecting different quantum network hardware.

4.1. Introduction

The future quantum internet will leverage the principles of quantum mechanics for ultra-secure communication, enhanced sensing, and distributed quantum computing [1, 2]. Progress in the past decade has led to pioneering experiments on different components of such a network [3–7]. For instance, entanglement generation between separated quantum memory systems based on atomic ensembles has recently been reported [8, 9] and the first multi-node network of rudimentary quantum processors has been realized inside the lab [10, 11]. As different hardware platforms may be optimized for different network tasks, realizing interfaces that enable quantum information transfer between heterogeneous devices is a key challenge.

Here, we report on a proof-of-concept demonstration of a quantum interface between a diamond NV center qubit [12, 13] and photonic time-bin qubits at 795nm that are compatible with Thulium-based solid-state memories [14–16] and Rubidium-based atomic gas memories [17–20]. Such an interface conceptually corresponds to future quantum Internet scenarios such as connecting remote qubit processors via a repeater chain [21] or realizing remote state preparation on a quantum computing server from a photonic client [22]. We validate the quantum nature of the interface by performing quantum teleportation [23, 24] of 795nm time-bin qubits into the NV center spin qubit with state fidelity beating the classical bound.

4.1.1. A quantum interface between heterogeneous devices

A major challenge for linking heterogeneous quantum network hardware is the matching of their corresponding photonic qubits. Many leading hardware platforms for quantum memories and quantum network nodes are based on atom-like systems [25, 26]. The properties of the photonic interface of these platforms, such as temporal profile and wavelength of emitted photons, are therefore largely determined by the atomic properties and vary significantly among the different platforms. Our approach to bridging these differences is depicted in the schematic of our interface in Fig. 4.1.

The interface converts the input 795nm photonic time-bin qubit to match the properties of the NV center photon. In parallel, entanglement is generated between the spin state of the NV center and the temporal mode of a single emitted photon. Then, the converted 795nm photon and the NV photon are interfered on a beam-splitter. Subsequent detection of the photons in different time bins constitutes a Bell state measurement that teleports the original 795nm time-bin qubit state to the NV spin qubit. Real-time feed-forward of the Bell-state measurement outcome and application of the corresponding correction gate on the NV spin qubit completes the action of the interface.

For this interface to function with high fidelity, it is crucial that the converted 795nm photons are indistinguishable from the NV center photons. In particular, the 795nm photons need to match the NV photons' 637nm wavelength, polarization and exponential temporal profile set by NV's 12 ns optical lifetime. In this proof-of-concept work, we create photonic time-bin qubits at 795nm from weak coherent states by using an intensity modulator and a phase modulator. We calibrate the intensity modulator to mimic the NV photon's temporal profile within a 30ns time window. The photonic states obtained through this method are compatible with the storage and retrieval from Thulium-doped solid-state quantum memories [27] as well as Rubidium-gas-based quantum mem-

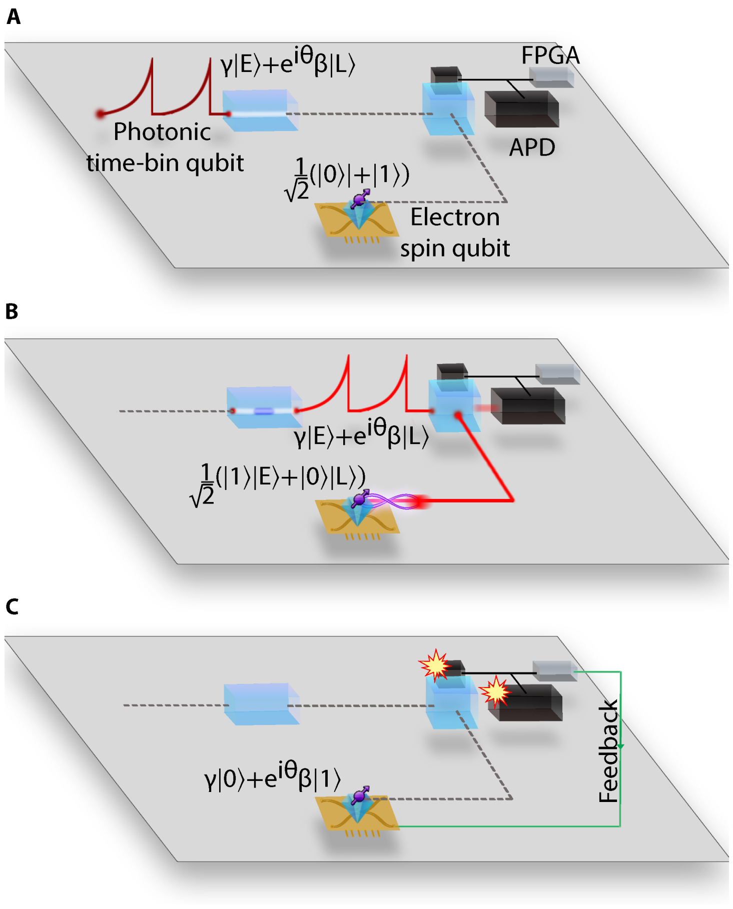


Figure 4.1: **Overview of the quantum interface between 795nm photonic time-bin qubits and an NV center processing node.** The interface consists of a low-noise two-step frequency conversion module including a frequency stabilization module, an interference station containing a balanced beam splitter with output ports connected to avalanche photodiodes (APDs), and an FPGA for real-time feedback. The interface can be visualized in three steps: a) a 795nm time-bin qubit with a temporal shape matching the spontaneous emission profile of the NV center is sent to the input of the interface. The NV spin qubit is prepared in a balanced superposition state. b) The 795 nm photonic qubit is converted to 637 nm, while the NV center generates a 637 nm photonic time-bin qubit entangled with the spin qubit. The generation of the 795 nm photonic qubit is timed to ensure maximum overlap at the beam splitter with the NV center photonic qubit. (c) Upon detecting one photon in each time bin, feedback of the correct phase flip to the NV spin qubit of the NV completes the state teleportation. For the experiments reported in this paper, we employed the NV qubit setup, called “Alice", described in Refs. [10, 11], that includes a microcontroller unit (MCU) and a fast waveform generator (AWG).

ories [28, 29]. These platforms are capable of multiplexing by storage and retrieval of multiple photonic modes in different degrees of freedom [26, 30], and therefore have attracted interest for quantum repeater applications [31].

To achieve wavelength indistinguishability, we employ a low-noise two-stage quantum frequency conversion process [32], depicted in Fig. 4.2a. In the first step, the 795nm shaped weak-coherent state is overlapped with a 1064nm pump laser and coupled into a temperature-stabilized periodically-poled Lithium Niobate (ppLN) waveguide crystal, generating 455nm light via a sum-frequency conversion process with conversion efficiency of 32% (Fig. 4.2b), measured free-space to free-space between the output of the input fiber and before the coupling into the output fiber. Subsequently, the 455nm light is down-converted to 637nm using a 1596nm pump laser, with conversion efficiency of 22%. At the output of each ppLN crystal we include dichroic mirrors and filters to remove residual unconverted light and pump light. The overall process efficiency including in- and outcoupling from fibers and filtering is 3%, which is sufficient for the current proof-of-concept but should be further improved in future designs. Importantly, having both pump lasers red-detuned from the signal photons results in a negligible amount of added noise in the conversion stages, obtaining a signal-to-noise ratio$\geq$1250, when only the conversion setup is considered. To ensure that the converted light precisely matches the NV photon frequency, despite unavoidable component drifts, the frequency of converted 795nm light is locked to the NV excitation laser light. To this end, an identical two-stage conversion setup is employed with 1mW at the input derived from the same 795nm source (Fig. 4.2a). Details on the frequency locking procedure and the employed electronics are discussed in the Supplementary Information. The resulting spread of the beat signal is 75 kHz, pushing the corresponding contribution to teleportation infidelity well below 1% (see below).

4.2. RESULTS

4.2.1. TWO-PHOTON QUANTUM INTERFERENCE BETWEEN CONVERTED 795NM PHOTONS AND NV CENTER PHOTONS

To investigate the degree of indistinguishability of the NV photons and the converted 795nm photons, we perform a two-photon quantum interference (TPQI) experiment, also known as Hong-Ou-Mandel interference. Perfectly indistinguishable photons interfering on a balanced beam-splitter show bosonic coalescence leading to zero probability of detecting photons in both output ports of the beam-splitter [33]. In such an experiment, on one side we employ the NV center in its negatively charged state NV$^-$. The ground state of NV$^-$ is a spin-1 system whose spin sublevels are split by the zero-field splitting and the applied magnetic field of 25.3mT [12]. We employ the m$_S$= 0 (-1) spin state as the $|0\rangle$ ($|1\rangle$) qubit state. The NV optical transitions are spin-dependent, allowing for spin-selective optical excitation and photon emission. In the current work, we use the cycling transition $|0\rangle \rightarrow |g\rangle$, where $|g\rangle$ represent the $|E_x\rangle$ excited state. In the TPQI experiment, the NV center can be modeled as a single-photon source parametrized by the probability of a photon detection per optical excitation p_{NV} (counts per shot).

On the other side, the 795nm photonic states constitute a multi-photon source, featuring Poissonian photon statistics. Up to the second order, the emission can be approx-

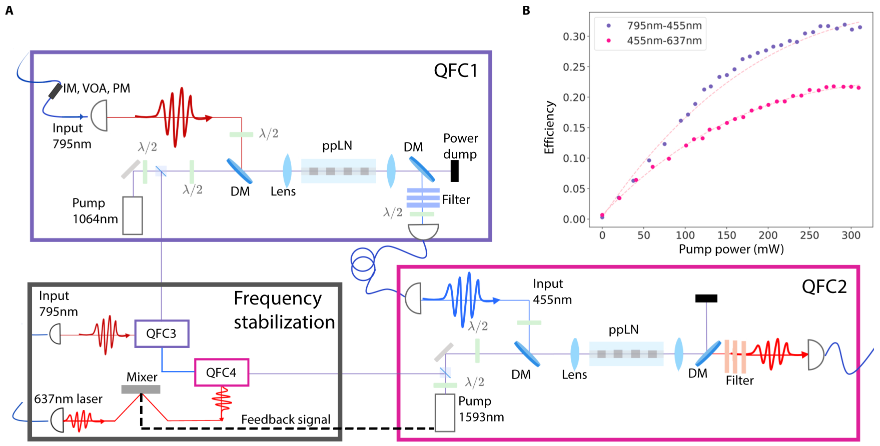


Figure 4.2: **Quantum frequency conversion setup.** a) Overview of the frequency conversion setup. To generate converted weak-coherent states at 637nm, the input light at 795nm undergoes two-step frequency conversion (QFC1 and QFC2), after passing through an intensity modulator (IM) to obtain the typical decay time-shape of the NV spontaneous emission, a variable optical attenuator (VOA) to manipulate the mean photon number, and a phase modulator (PM) for imprinting a phase on the time-bin qubits (only for quantum teleportation experiments). At the output of each conversion step, a dichroic mirror (DM) and a set of filters suppress residual unconverted light and pump light. A copy of the two-step frequency conversion setup (QFC3 and QFC4) is used for frequency stabilization. A higher power tap-off from the 795nm laser is converted and the resulting 637nm light is mixed with the light coming from the excitation laser of the NV. An error signal is computed and fed back to the frequency modulator of the 1593nm pump laser to match the converted light to the excitation wavelength of the NV. b) Measured efficiency for each step of the conversion while sweeping the power of the corresponding pump laser. The dashed lines represent the respective fit of the data points to a saturation curve, to extrapolate the optimal pump power. The relative error on each data point is 1%.

imated through the mean-photon number $|\alpha|^2$ as $|\alpha|^2 + 1/2|\alpha|^4$ [34]. The consequences of having two photonic sources with different statistics are discussed in detail in Sec. 4.5.

In Fig. 4.3a, the experimental sequence for the TPQI experiment is depicted. In the first step, a Charge-Resonance (CR) check is performed [35], which ensures that the NV center is in the correct charge state (namely, NV^{-}) and the lasers are on resonance with the relevant NV transitions. When the CR check threshold is satisfied, the actual TPQI experiment is triggered. Two trains of 10 optical π-pulses each, which we define as 10 different bins, are sent to the NV, which leads to 20 possible emission windows (10 per train). Each train is preceded by an optical spin-reset pulse that prepares the NV in the $|0\rangle$ state. In parallel, two trains of 10 decay-shaped pulses each are sent from the 795nm laser. The mean-photon number can be manipulated via a variable optical attenuator (VOA). As illustrated in Fig. 4.3a, the first train of pulses constitutes the indistinguishable sequence, with the two photonic states overlapping in time on the beam splitter. The second train is the distinguishable sequence: each 795nm photonic state is delayed by 50ns with respect to the corresponding NV photon, rendering the photons fully distinguishable. The sequence of two trains is repeated 100 times before returning to the CR check. The next CR check validates both the previous TPQI sequence and, in case the threshold is satisfied, directly triggers the next sequence. In case the validation round of CR check fails, the experiment iteration is discarded in the analysis, while the CR

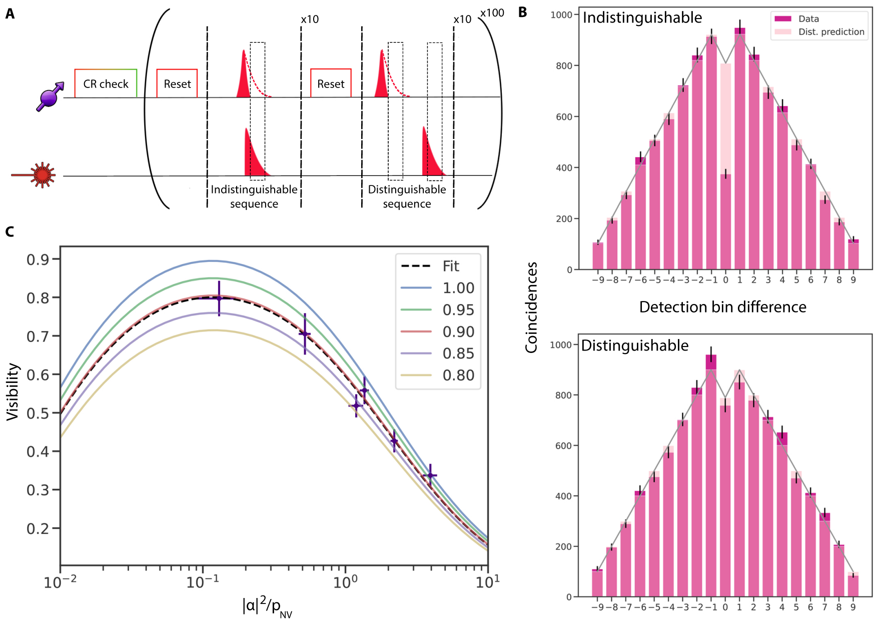


Figure 4.3: **Two-photon quantum interference.** a) Experimental sequence for the two-photon quantum interference experiment. The top line refers to the NV, while the bottom refers to the 795nm weak-coherent state. b) Histograms of the coincident clicks in the two sequences (magenta) for the ratio $x = |\alpha|^2/p_{NV}$ of 1.19±0.14. Each bar of the histograms represents coincident clicks in the two detectors within 30ns windows for all the possible combinations of a given time bin difference. As a reference, we also include, in both histograms, the expected coincidences in the case of perfectly distinguishable photons (in pink). In both diagrams, the grey line connects the expected values of the distinguishable prediction. In the distinguishable case, we consider two pulse windows: one around the NV photons and one around the converted 795nm photonic states. Therefore, the histogram contains the contribution of coincident clicks for three possible cases: coincident clicks in the NV window, in the converted 795nm window and in the combined windows. c) Extracted visibility for different values of x. The values are fitted according to the visibility model included in Sec. 4.5. The dashed line represents the fit result, corresponding to indistinguishability of 0.895±0.019, while the colored lines represent our model of the visibility for different values of indistinguishability.

check is repeated until success to trigger the next experiment iteration. The emitted photons from both sides impinge on a 50:50 in-fiber beam splitter, whose output ports are connected to two single-photon detectors. A timetagger registers the detection times of the photons in the two output ports, enabling the reconstruction of the histograms in Fig. 4.3b. Each bin of the histogram counts the number of coincident clicks

From the histograms in Fig. 4.3b we extract the visibility V=1-$\frac{p_{ind}}{p_{dist}}$, where p_{ind} (p_{dist}) is the probability of a coincidence detection if the photons are indistinguishable (distinguishable) in the 0-bin difference. Taking into account that the photonic states follow different statistics and introducing the indistinguishability η, the visibility can be expressed as

$$V = \frac{\eta x}{\frac{1}{2}g^{(2)}(0) + \frac{1}{2}x^2 + x + \frac{2p_{noise}(1+x)}{p_{NV}} + \frac{2p_{noise}^2}{p_{NV}^2}} \tag{4.1}$$

(see 4.5.3 for the derivation), where x is the ratio $|\alpha|^2/p_{NV}$ and p_{noise} is the probability of a background (noise) click per 30ns window in one detector.

By performing this TPQI experiment and extracting the visibility for 6 different values of x, we can reconstruct the visibility function as shown in Fig. 4.3c. We also plot the expected visibility function, using the independently measured values for the NV $g^{(2)}$ of 0.011±0.004 and p_{NV} of (5.76±0.20)e^{-4}, for several values of η. The value of p$_{noise}$ is discussed in Sec. 4.5. We observe that the data follows the model closely over the full range. From a fit to the data we obtain the indistinguishability η=(0.895±0.019), showing that we have matched all the relevant degrees of freedom of the two photonic states to a high level. The limited indistinguishability can be due to a residual mismatch between the temporal profile of the NV photons and the 795nm photons, as well as an imperfect coherence of the NV photons, which from previous NV-NV TPQI experiments showed limited indistinguishability of 0.9 [10].

4.2.2. QUBIT TELEPORTATION FROM A 795NM PHOTONIC TIME-BIN QUBIT TO THE NV CENTER SPIN QUBIT

Having established the high indistinguishability of the photonic states involved, we exploit the interface to perform quantum teleportation of 795nm time-bin qubits to the NV electron spin qubit, as illustrated in the diagram in Fig. 4.4a. Real-time feed-forward is included to complete the teleportation, enabling the correction of phase-flipped outcomes in the Bell-state measurement and delivery of the teleported state “alive".

The 795nm time-bin qubit is constituted by an early and late weak-coherent state separated by 300 ns and generated in the same way as in the TPQI experiment. Additionally, we include a phase modulator (PM) to manipulate the phase difference between the early and late temporal modes. The resulting qubit state is therefore in the general form of$\gamma|E\rangle + e^{i\theta}\beta|L\rangle)$ with $|E\rangle$ ($|L\rangle$) denoting the early (late) time bin. For this experiment, we prepare time-bin qubits in an unbiased set of states (the cardinal states) that we indicate as: $|Z\rangle$, $|-Z\rangle$, $|X\rangle$, $|-X\rangle$, $|Y\rangle$, $|-Y\rangle$, referring to their position on the Bloch sphere.

On the NV's side, the electron spin qubit is optically initialized in $|0\rangle$. A microwave $\pi/2$ rotation along $\hat{x}$ axis of the Bloch sphere brings the qubit into a balanced superposition state. An optical π-pulse excites the NV's population in $|0\rangle$, enabling the spontaneous emission of a photon in the early time bin. Subsequently, the electron spin goes through a microwave π rotation along $\hat{y}$ axis, and another optical π- pulse enables the NV to spontaneously emit the late time-bin photon. The resulting NV-photon entangled state is $1/\sqrt{2}(|1\rangle|E\rangle \pm |0\rangle|L\rangle)$. Throughout the teleportation experiment we keep the ratio $|\alpha|^2/\mathrm{p}_{NV}$ constant at 1.20±0.24 by regular recalibration.

The converted 795nm photonic state and the NV photon interfere on the balanced beam splitter, erasing the which-path information. Successful teleportation is heralded by the detection of a photon in each of the two time-bins. We can discriminate between the Bell states $|\Psi^+\rangle$ and $|\Psi^-\rangle$ by the double-click pattern: two clicks on the same detector for $|\Psi^+\rangle$ and two clicks on two different detectors for $|\Psi^-\rangle$. The valid detector clicks are detected by an FPGA in a 50ns window around the corresponding photons' time of arrival. In data analysis, we further shorten the valid teleportation time window to 20ns for an improved signal-to-noise ratio. The teleportation sequence is repeated for a maximum of 50 times before going back to the CR check.

When the FPGA detects a valid click pattern, it sends a two-bit message to the AWG. The AWG jumps out from the teleportation attempt sequence (Fig. 4.4a) and starts the feedback and tomography sequence for the electron spin state. This sequence is composed of an XY4 dynamical decoupling sequence [36] followed by a basis selection pulse for the tomography. The latter is selected in real-time, taking the detector click pattern into account by applying a phase-flip correction when necessary. Finally, a single-shot readout of the NV spin qubit is performed. Throughout the measurement, a set of automated measurement and calibration routines detect anomalies in the converted frequency and in the reset frequency, declaring those datasets as failed when the required parameters are not met (more details in Sec. 4.5). In Fig. 4.4b, the results for the teleportation of the six cardinal states are reported together with the average state fidelity. The average fidelity is obtained as $F_{avg}=1/3\bar{Z}+1/3\bar{X}+1/3\bar{Y}$, where $\bar{Z}$, $\bar{X}$, $\bar{Y}$ represent the average fidelity along the respective axis. The resulting fidelity of (75.5 ± 1.0)%. is well above the classical bound, which is set taking into account the use of a multi-photon source (see SI). Additionally, we also calculate the average fidelity for the equatorial states in the

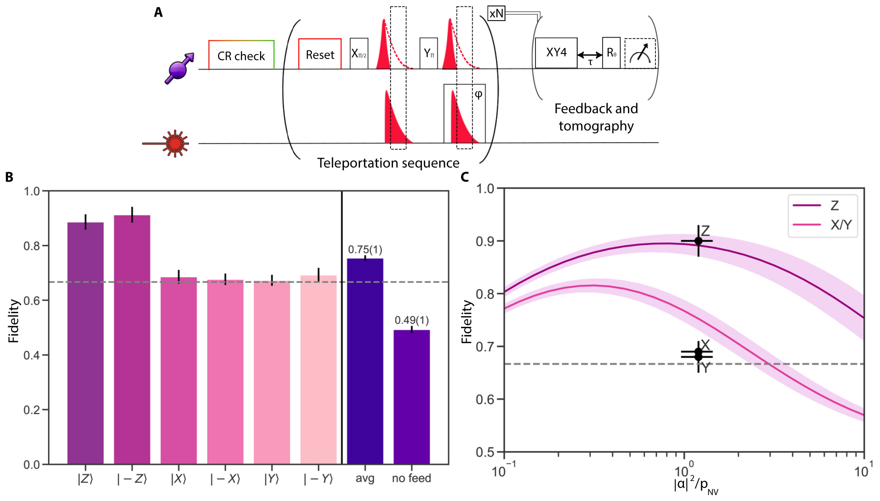


Figure 4.4: **Quantum teleportation of a time-bin qubit into the electron spin of the NV center.** a) Experimental sequence. After passing the CR check, the AWG plays the teleportation sequence as described in the main text. Such a sequence is played N times. The timeout for the teleportation sequence is set to 50 repetitions, after which the NV center goes back to CR check. If a valid click pattern is detected by the FPGA in the Bell-state measurement, the AWG jumps out from the teleportation sequence and starts the feedback and tomography sequence. This sequence contains an XY4 set of pulses, where the first pulse is played after a time τ with respect to the $\pi/2$ rotation pulse in the teleportation sequence. The value τ is a multiple of the Larmor period of the electron spin, optimized taking into account the effects of the spin bath. After the XY4 sequence, the base selection pulse is played taking into account the input state and the Bell-state measurement outcome. Finally, the tomography single-shot readout of the NV electron spin is performed at the MCU level. b) Histogram showing the individual fidelities per each cardinal state, as well as the average and the resulting fidelity when no feedback operations are applied to the NV electron spin qubit. The results are corrected for tomography errors, but not for preparation errors. c) Simulation curves based on the model described in Sec. 4.5. The data points in black represent the average fidelity for the states along the three axes of the Bloch sphere. The dashed line indicates the corrected classical bound.

absence of feed-forward. In this case, the fidelity is consistent with a fully mixed state, confirming the critical role of feedback in the teleportation protocol. In the measured fidelities, we also filter based on the CR check's validation.

In Fig. 4.4c we report the comparison between the measured fidelities and the predicted values of our model as a function of the ratio $|\alpha|^2/\mathrm{p}_{NV}$. The model includes the effect from leakage of the intensity modulator, resulting in a preparation error for the Z states of around 4%. More details on the simulated curves are included in Sec. 4.5. The small discrepancy between the X and Y data points and the simulation may be due to errors not captured by the model. On one side, the model does not consider imperfections in the microwave pulses that implement the NV quantum gates, which we estimate to cause an accumulated error below 1.5%. On the time-bin qubit side, our model does not take into account phase errors due to imperfections in the fast phase modulation, which affect the preparation of the X and Y states but not Z. Correcting for the input photonic qubit preparation errors yields a best estimate for the teleportation fidelity of $(78.3 \pm 0.9)\%$.

4.3. Discussion

We have benchmarked a photonic interface between 795nm converted time-bin qubits and an NV center-based quantum processor. The time-bin qubits are compatible with Thulium-doped crystals employed for quantum memories as well as Rubidium gas quantum memories. The interface exhibits a high photon indistinguishability, thanks to a low-noise two-step quantum frequency conversion setup, that leads to beating the classical bound for the quantum teleportation protocol, together with the capabilities of the NV center as quantum processor, which shows long coherence time and a reliable optical interface. Additionally, the implementation of control scripts made the setup to be operable at a distance and for long periods. Our results demonstrate the realization of interfaces between heterogeneous platforms that constitute the building-blocks of the future Quantum Internet. The interface presented in this work is versatile, as the methods and results presented can be transferred to platforms with similar functionalities. Hence, further improvements can be targeted at several aspects, like application field, experimental rate and bandwidth matching. Some examples might include the use of actual quantum memories that can be synchronized with the photon emission from the NV center, along with the integration of NV centers into optical cavities [37] for higher photon rate. Another possibility is the use of different color center defects in diamonds, like the group-IV, that promise higher photon emission rates and the possibility of integration into nanophotonic structures [38–41]. Other promising quantum processor platforms might include defect centers in SiC [42], Si [43] and optical quantum dots [4, 44]. Additionally, higher efficiency frequency conversion setups to telecom wavelengths [45, 46] can be employed to convert the photons emitted from both parties, leading towards the real-case scenario of quantum networks over long distances.

4.4. METHODS

4.4.1. FIDELITY CALCULATION

The fidelity for each teleported state is calculated as $F = (1 + \frac{R_{\langle i \rangle i} - R_{\langle j \rangle i}}{R_{\langle i \rangle i} + R_{\langle j \rangle i}})/2$, given that we prepared the time-bin qubit in the state $|i\rangle$ and we measure in the state $|i\rangle$ and in its orthogonal state $|j\rangle$. The quantity $R_{\langle\rangle}$ represents the tomography-related single-shot readout outcome, including the correction for known errors (see Supplementary of [10]).

4.5. SUPPLEMENTARY INFORMATION

4.5.1. FREQUENCY LOCKING

To ensure indistinguishability in frequency between the converted weak coherent states and the zero phonon line emission of the NV we apply an active frequency locking scheme outlined in Fig. 4.5. We use two additional frequency converters (QFC3 and QFC4) which convert a continuous wave tap-off from the 795nm laser to light at 637nm. We interfere this light on a balanced beamsplitter with light from the laser that excites the NV center. This excitation light passes through an AOM before reaching the NV center, which shifts the frequency by 200 MHz. Thus, we stabilize the frequency of the converted light to a fixed frequency offset of 200 MHz. We detect the light in the two output ports of the beamsplitter (BS) using a balanced photodiode, the output of which feeds, together with a 200 MHz reference, that comes from the driving frequency of the AOM, into a custom control box based on a HMC3716 Digital Phase Frequency Detector. This control box generates an error signal which we use to adapt the frequency of our telecom pump laser.

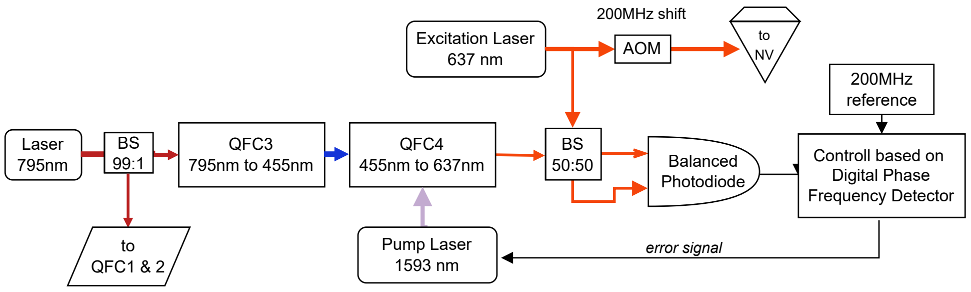


Figure 4.5: Outline of the frequency locking scheme. The converted light and the excitation laser light interfere on a balanced beam-splitter. The signal from the two output ports is detected by a balanced photodiode. A custom and homemade Digital Phase Frequency Detector box generates an error signal that is used by the frequency modulator of the pump laser to correct the second step of the frequency conversion.

4.5.2. TELEPORTATION PROTOCOL

In this section, we report step-by-step the general evolution of the total system.

- Initialization: on the time-bin qubit side, we prepare the general state $\gamma|E\rangle + e^{i\theta}\beta|L\rangle$. For the NV center, we initialize the electron spin in a superposition state: $1/\sqrt{2}(|0\rangle +$

$|1\rangle$). The state after the double optical excitation of the NV is $\frac{1}{\sqrt{2}}(|1\rangle|E_{NV}\rangle+|0\rangle|L_{NV}\rangle)$. Therefore, the total state of the system is the following: $(\gamma|E\rangle+e^{i\theta}\beta|L\rangle)\frac{1}{\sqrt{2}}(|1\rangle|E_{NV}\rangle+|0\rangle|L_{NV}\rangle)$

- Bell-state measurement: given that the photons are indistinguishable, when they impinge on the beam-splitter, we erase the which-path information and the Hong-Ou-Mandel effect holds. The beam-splitter entangles the photons coming from the two platforms. We post-select on events where the detectors clicked in both the early and late time-bin (same detector or different ones), namely we project into the state: $\frac{1}{\sqrt{2}}(|E\rangle|L\rangle \pm |L\rangle|E\rangle)$.

- Real-time feed-forward: To retrieve the correct state on the electron spin qubit's side, a feed-forward operation, based on the outcome of the Bell-state measurement, is necessary. The two different click pattern reflect a phase difference of π between the two output ports. Therefore, in case of a click pattern where early and late photons are detected in two different detectors, an extra π rotation is fed back in real-time to the electron spin qubit.

4.5.3. Model of Expected TPQI Visibility

In this section, we will give details on our model to predict the Two Photon Quantum Interference (TPQI) Visibility. We base this model on a similar one presented in [47]. We will derive the expected TPQI visibility as a function of p_{NV}, the NV emission probability and $|\alpha|^2$ the mean photon number of the weak coherent state.

We start by defining the visibility as

$$V = 1 - \frac{p_{ind}}{p_{dist}} \tag{4.2}$$

with p_{ind} (p_{dist}) the probability of a coincidence detection if the photons are indistinguishable (distinguishable). We will assume the probability of 3-photon events to be negligible and thus p_{ind} consists of four contributions

$$p_{ind} = p_{2nv} + p_{2wcs} + p_{nv,wcs} + p_{bg}, \tag{4.3}$$

where p_{2nv} is the probability of detecting 2 NV photons, p_{2wcs} is the probability of detecting 2 photons from the weak coherent state, $p_{nv,wcs}$ is the probability of detecting one NV photon and one weak coherent state photon and p_{bg} is the probability of detecting a coincidence where one click is originated from the background noise. Here, the probability of detecting 2 photons in the NV detection window is given by

$$p_{2nv} = \frac{1}{4} p_{NV}^2 g^{(2)} \tag{4.4}$$

The autocorrelation coefficient $g^{(2)}$ of the NV can be determined separately during the distinguishable sequence of the TPQI experiment by calculating the ratio of a coincidence event p_{coinc} between the two detectors $D1$ and $D2$ and the individual probabilities of a detector click $p_{D1(2)} = 0.5 p_{NV}$, $g^{(2)} = \frac{p_{coinc}}{p_{D1} p_{D2}}$.

Secondly, the probability of detecting two photons originating from the weak coherent state is given by $p_{2wcs} = \frac{1}{4}|\alpha|^4$. Furthermore, the probability of a coincidence originating from one photon entering on each side of the beam splitter depends on the indistinguishability η of the photons

$$p_{nv,wcs} = (1-\eta)\frac{p_{NV}|\alpha|^2}{2} \tag{4.5}$$

and finally the probability of a coincidence where a noise count is involved is given by

$$p_{bg} = p_{noise}(|\alpha|^2 + p_{NV} + p_{noise}) \tag{4.6}$$

with p_{noise}, the probability of a single background (noise) click in one detector per time window. Thus, p_{ind} becomes

$$p_{ind} = \frac{1}{4}p_{NV}^2 g^{(2)} + \frac{1}{4}|\alpha|^4 + (1-\eta)\frac{p_{NV}|\alpha|^2}{2} + p_{noise}(|\alpha|^2 + p_{NV} + p_{noise}) \tag{4.7}$$

For perfectly distinguishable photons ($\eta = 0$) we obtain

$$p_{dist} = \frac{1}{4}p_{NV}^2 g^{(2)} + \frac{1}{4}|\alpha|^4 + \frac{p_{NV}|\alpha|^2}{2} + p_{noise}(|\alpha|^2 + p_{NV} + p_{noise}) \tag{4.8}$$

This then leads to

$$V = 1 - \frac{p_{ind}}{p_{dist}} = 1 - \frac{\frac{1}{4}p_{NV}^2 g^{(2)} + \frac{1}{4}|\alpha|^4 + (1-\eta)\frac{p_{NV}|\alpha|^2}{2} + p_{noise}(|\alpha|^2 + p_{NV} + p_{noise})}{\frac{1}{4}p_{NV}^2 g^{(2)} + \frac{1}{4}|\alpha|^4 + \frac{p_{NV}|\alpha|^2}{2} + p_{noise}(|\alpha|^2 + p_{NV} + p_{noise})} \tag{4.9}$$

or

$$V = \frac{\eta p_{NV}|\alpha|^2}{\frac{1}{2}p_{NV}^2 g^{(2)} + \frac{1}{2}|\alpha|^4 + p_{NV}|\alpha|^2 + 2p_{noise}(|\alpha|^2 + p_{NV} + p_{noise})} \tag{4.10}$$

Finally, we can re-write the Visibility as a function of the ratio x between $|\alpha|^2$ and p_{NV}, with $x = |\alpha|^2/p_{NV}$ as follows

$$V = \frac{\eta x}{\frac{1}{2}g^{(2)} + \frac{1}{2}x^2 + x + \frac{2p_{noise}(1+x)}{p_{NV}} + \frac{2p_{noise}^2}{p_{NV}^2}} \tag{4.11}$$

which we use to fit the data shown in Fig. 4.3 of the main text.

4.5.4. MODEL OF EXPECTED TELEPORTATION FIDELITY

In this section, we will derive the model we used to predict the Fidelity of the teleported state. We start by defining the input states and measurements and then continue to discuss the different emission patterns that can lead to a valid heralding event and how they affect the final fidelity we can expect to observe.

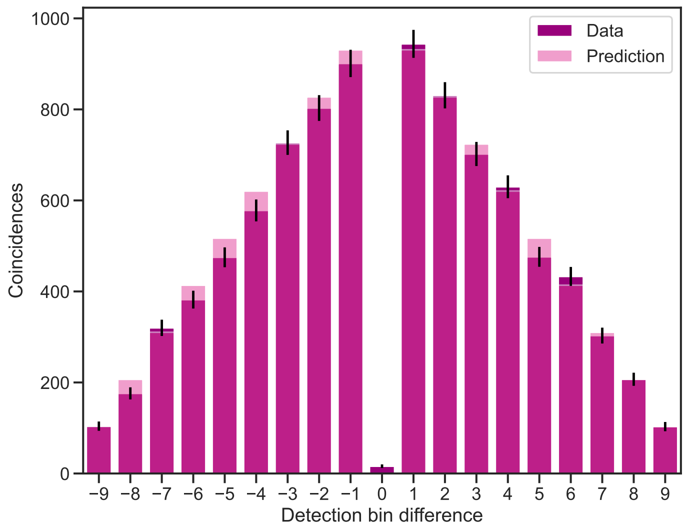


Figure 4.6: Autocorrelation function of the NV center. The magenta bins represent the data collected during the TPQI experiment for all the measured ratios. The pink bins, instead, represent the expected coincidences when assuming a perfect single-photon source. The resulting $g^{(2)}$ value corrected for noise is 0.011±0.004.

The state to be teleported will be denoted as $|\Psi_A\rangle$ and is prepared in either the early time bin $|E\rangle$, the late time bin $|L\rangle$ or an equal superposition of the two. The set of states prepared for teleportation are the two states on the poles of the Bloch-sphere

$$|+Z\rangle = |E\rangle \tag{4.12}$$
$$|-Z\rangle = |L\rangle \tag{4.13}$$
$$\tag{4.14}$$

as well as the four equatorial states

$$|\pm X\rangle = \frac{1}{\sqrt{2}}(|E\rangle \pm |L\rangle) \tag{4.15}$$
$$|\pm Y\rangle = \frac{1}{\sqrt{2}}(|E\rangle \pm i|L\rangle) \tag{4.16}$$
$$\tag{4.17}$$

. Due to imperfections in the preparation, the polar states cannot be prepared perfectly. Still, there is a probability of leakage light emission in the orthogonal time bin which we will consider in our model. The electron spin qubit of the NV center and the emitted photon are prepared in the joint state

$$|\Phi\rangle_B = \frac{1}{\sqrt{2}}(|1\rangle|E\rangle + |0\rangle|L\rangle) \tag{4.18}$$

where $|0\rangle$ ($|1\rangle$) denotes the bright (dark) state of the spin qubit (see main text).

The Bell-state measurement, required for teleportation consists of interfering the two photonic states on a balanced beam splitter and post-selecting events where a detection event happened in both the early and late time-bin (either in the same or in different detectors). This corresponds to projection onto the states:

$$|\Psi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|E\rangle|L\rangle \pm |L\rangle|E\rangle) \tag{4.19}$$

where the sign is determined by the detection pattern. As neither of the photon sources emits perfect single-photon states, we have to consider several distinct emission cases that can lead to an accepted heralding signal and how these affect the resulting quantum state. Due to the low emission probability, we neglect all terms in which more than two photons are emitted from any side as well as the case where both sides emit two photons. We will start by defining the different probabilities of occurrence for different photon numbers. For the weak coherent state we have:

$$P_0^w = e^{-\mu} \tag{4.20}$$
$$P_1^w = e^{-\mu}\mu \tag{4.21}$$
$$P_2^w = e^{-\mu}\frac{\mu^2}{2} \tag{4.22}$$
$$\tag{4.23}$$

with $P_i^w$the probability of emission of i photons from a weak coherent state with mean photon number μ.

The probability of collecting i photons from the NV-center is given by P_i^{NV}

$$P_0^{NV} = (1 - p_{NV}) \tag{4.24}$$
$$P_1^{NV} = p_{NV}(1 - p_{de}) \tag{4.25}$$
$$P_2^{NV} = p_{NV}p_{de} \tag{4.26}$$
$$\tag{4.27}$$

where p_{de} denotes the double excitation probability of the NV-center. We now write out these individual contributions as non-normalized density matrices ρ_{ij} with their respective probability of occurrence $P_i^{NV}P_j^w$ for the different numbers of photons emitted from the NV center (i) and from the weak coherent state (j) as well as contributions in which (at least) one click was triggered by a background (or noise) count. For the case where the weak coherent state is prepared in a pole state ($\pm|Z\rangle$) we also consider the probability of emitting k unwanted or leaked photons in the state orthogonal to the desired one which are denoted as $P_k^{w\perp}$. In the following we use the simplified notation P_{ijk} for $P_i^{NV}P_j^wP_k^{w\perp}$ or P_{ij} for $P_i^{NV}P_j^w$. The probability of one or two background or noise photons contributing to a valid trigger event is, for the pole states, given by

$$P_{bg}^{pole} = 2p_{noise}(2p_{noise}P_{000} + P_{010} + P_{100} + P_{001}) \tag{4.28}$$

and for the equatorial states by

$$P_{bg}^{eq} = 2p_{noise}(2p_{noise}P_{00} + P_{10} + P_{01}) \tag{4.29}$$

Where p_{noise} is the probability of a background or noise detection per detector and time bin and we have limited the background contributions we consider to a maximum of two emitted photons. For the teleportation experiment, the measured p_{noise} per detector is $(5.5\pm0.2)\mathrm{e}^{-6}$.

Now we will consider the non-normalized density matrix contributions corresponding to these probabilities of occurrence. They are non-normalized as not all detection patterns are considered valid trigger events and we will post-select on these valid heralding events. In writing down these contributions and their probabilities of yielding a valid heralding event, we will have to differentiate between teleporting the pole-states ($|\pm Z\rangle$) from the states in the equatorial plane of the Bloch-sphere ($|\pm X\rangle$ and $|\pm Y\rangle$), which we will mark as ρ^{pole} and ρ^{eq}. For the desired case of one emitted NV photon and one photon emitted from the weak coherent state we can write

$$\rho_{11}^{pole} = \frac{P_{110}}{2}|\Psi_A\rangle\langle\Psi_A| + \frac{P_{101}}{2}|\Psi_A^{\perp}\rangle\langle\Psi_A^{\perp}| \tag{4.30}$$

$$\rho_{11}^{eq} = P_{11}(\frac{\eta}{2}|\Psi_A\rangle\langle\Psi_A| + \frac{(1-\eta)}{2}\mathbb{I}) \tag{4.31}$$

The factor $\frac{1}{2}$ takes into account the fact that we could project on any of the four Bell-states but we can only unambiguously discern two of them and thus, only these two will lead to an accepted heralding pattern. The difference between the pole and the equatorial states is due to the fact that in the case of the pole states we profit from the classical correlations in the system, while for the equatorial states we will only obtain the desired result if the photons from the two sources interfere.

In the case of a double emission from the weak coherent state and no NV photon, there will only be a valid trigger event for a pole state if the double emission happened in the form of one photon from the desired time bin and one in the orthogonal one, and there can as well be a valid trigger in case of an equatorial state which will lead to

$$\rho_{02}^{pole} = P_{011}\mathbb{I} \tag{4.32}$$

$$\rho_{02}^{eq} = \frac{P_{02}}{2}\mathbb{I} \tag{4.33}$$

We will omit the case in which one photon was emitted from the NV center but two from the weak coherent state due to it's low probability of occurrence for the mean photon numbers used in the experiment.

Finally, for the case of two NV photons and one from the weak coherent state we get

$$\rho_{21}^{pole} = \frac{P_{210}}{4}|\Psi_A\rangle\langle\Psi_A| + \frac{P_{201}}{4}|\Psi_A^{\perp}\rangle\langle\Psi_A^{\perp}| \tag{4.34}$$

$$\rho_{21}^{eq} = \frac{P_{21}}{4}\mathbb{I} \tag{4.35}$$

When teleporting pole states the final density matrix becomes

$$\rho^{pole} = \frac{\frac{1}{2}P_{110} + \frac{1}{4}P_{210}}{N}|\Psi_A\rangle\langle\Psi_A| + \frac{\frac{1}{2}P_{101} + \frac{1}{4}P_{201}}{N}|\Psi_A^{\perp}\rangle\langle\Psi_A^{\perp}| + \frac{P_{011} + P_{bg}^{pole}}{N}\mathbb{1} \tag{4.36}$$

with

$$N = \frac{P_{110} + P_{101}}{2} + \frac{P_{210} + P_{201}}{4} + P_{011} + P_{bg}^{pole} \tag{4.37}$$

Thus, we can calculate the expected fidelity to be

$$F^{pole} = \frac{P_{110} + \frac{1}{2}P_{210} + P_{011} + P_{bg}^{pole}}{2N} \tag{4.38}$$

For the equatorial states we obtain

$$\rho^{eq} = \frac{\eta P_{11}}{2N}|\Psi_A\rangle\langle\Psi_A| + \frac{(1-\eta)P_{11} + P_{02} + P_{21} + 2P_{bg}^{eq}}{2N}\mathbb{1} \tag{4.39}$$

with

$$N = \frac{P_{11} + P_{02}}{2} + \frac{P_{21}}{4} + P_{bg}^{eq}, \tag{4.40}$$

thus, the Fidelity is given as

$$F^{eq} = \frac{1}{2N}(\frac{P_{11}(1+\eta) + P_{02} + P_{21}}{2} + P_{bg}^{eq}) \tag{4.41}$$

4.5.5. CLASSICAL BOUND WHEN TELEPORTING WITH WEAK COHERENT STATES

When comparing our experimentally achieved teleportation fidelity with the classical bound of $\frac{2}{3}$, this bound is derived using the optimal classical strategy when using single photons to encode qubits [48]. In the case of qubits encoded in weak coherent states, however, a classical strategy might use the higher photon number contributions of that state to achieve a higher probability of success. One can calculate the maximally achievable Fidelity for a classical strategy as shown in [49] as

$$F_{max}(|\alpha|^2) = \sum_{N\geq 1} F_{MP}(N)\frac{p(|\alpha|^2, N)}{1 - p(|\alpha|^2, 0)} \tag{4.42}$$

where N is the number of photons per pulse, p describes the poissonian distribution of photon numbers

$$p(|\alpha|^2, N) = \frac{|\alpha|^{2N}}{N!}e^{-|\alpha|^2} \tag{4.43}$$

and

$$F_{MP}(N) = \frac{N+1}{N+2} \tag{4.44}$$

is the maximum achievable fidelity for a state with a fixed amount of photons N. For the parameters of our teleportation experiment we can now calculate the maximum achievable fidelity for a classical strategy as follows: During the teleportation experiment we had an average NV emission probability of $p_{NV} = (4.50 \pm 0.9)e^{-4}$ and a ratio $\frac{|\alpha|^2}{p_{NV}} = 1.20 \pm 0.24$. It is noteworthy that, with respect to the TPQI experiment, a degradation of the p_{NV} parameter occurred, as well as lower CR check counts were encountered. The reason for the lowered photon emission of the NV might be due to a fault in our cryostat that led to ice formation inside the sample chamber. Despite that, the setup was stable in the new configuration and the experiment was executed remotely. At the same time, the intensity modulator employed for the generation of the time-bin qubits showed higher leakage, leading to a generally increased noise probability. Using the upper bound of this ratio between mean photon number and p_{NV} (and thus $|\alpha|^2 = 6.50e^{-4}$) the maximally achievable fidelity for a classical strategy would be $F_{max} = 0.666694$. As we can see, due to the low mean photon numbers used in our experiment, the correction is minimal but should be considered in implementations with higher mean photon numbers.

4.5.6. NOISE CHARACTERIZATION

In this section, we report on the characterization of the noise sources that are involved in the experiments. To obtain the dark count rates of the detector, we block the ZPL collection path of the NV and the output of the QFC2. This results in a mean noise rate of (11.7±5.6)Hz. The contribution of the pump lasers for the two-step frequency conversion is measured by keeping the ZPL path closed and blocking the 795nm input of the QFC1. The rate in this case is (12.3±5.6)Hz, showing that our conversion setup is low noise, if compared with the rate obtained when no conversion setup was involved. To characterize the noise contribution of the weak-coherent state in the NV center window (particularly relevant in the distinguishable sequence of the TPQI experiment), we block the ZPL path of the NV center and we sweep the voltage applied to the variable optical attenuator, namely we manipulate the mean-photon number of the weak-coherent state. We play the distinguishable sequence of the TPQI experiment, collecting the counts in the time window where the NV center pulse is supposed to be. The results are illustrated in Fig. 4.7.

Lastly, we check the noise contribution coming from the NV setup. We block the output of the QFC2 setup and we open the ZPL path. We repeat the distinguishable sequence, collecting counts in the weak-coherent state window. The rate is (12.8 ± 5.4)Hz, which is comparable with the rate measured above for the detector dark counts and the pump noise. We can therefore conclude that the main source of noise comes from the preparation of the weak coherent state, particularly the combination of the intensity modulator, whose bias voltage needs to be optimized throughout the measurement, and the variable optical attenuator. However, this noise source is relevant only in the distinguishable sequence of the TPQI experiment, while in the indistinguishable one, we can

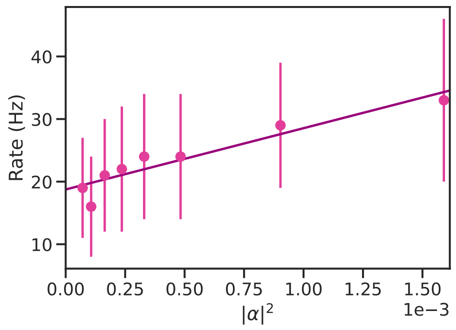


Figure 4.7: Noise characterization of the weak coherent state. By sweeping the voltage applied to the VOA, the mean-photon number changes. We assume a linear model for the noise count rate vs. the mean-photon number. The result of the linear fit is used in the calculation of the p_{noise} term in Eq. 4.9, thus affecting the visibility of the TPQI experiment.

consider as p_{noise} the constant background noise given by the detectors and the conversion setup. For the teleportation experiment, the noise rate increased to (275±10)Hz due to equipment degradation, as discussed above.

4.5.7. PHASE MODULATOR CHARACTERIZATION AND STABILITY

To characterize the phase modulator, namely to identify V_π and $V_{\pi/2}$, we build a Mach-Zehnder-type interferometer at 795nm. In particular, the input of the QFC1 from main text, the shaped pulses, is connected to a 50:50 in-fiber beam splitter. The two output arms of such a beam splitter are 2 m-long fibers, and on one of the two arms, we include the phase modulator device we want to characterize. The two arms impinge on a second in-fiber 50:50 beam splitter, whose output ports are connected to two APDs. The voltage source for the phase modulator is one of the wave channels of the AWG, whose output has an amplitude between ±5V.

We send a pulse to the interferometer, encountering a phase shift due to the phase modulator, and we register, through the time-tagger at the detectors, the counts per shot for the two pulses. We repeat this experiment while sweeping the voltage applied to the phase modulator, reconstructing the plot in Fig. 4.8a for the two detectors.

The data collected for the two detectors are jointly fit to the following curves:

$$A_1 \sin\left(\frac{s \cdot cps_{det_1}}{2} + o\right) + c_1$$

$$A_2 \cos\left(\frac{s \cdot cps_{det2}}{2} + o\right) + c_2$$

From the fit of the s parameter, it is possible to estimate V_π (and $V_{\pi/2}$) as $|\pi/s|$ (and $|\pi/(2s)|$).

At this point, we repeat the measurement and the data fit more times over a time span of 15h. In Fig. 4.8b we report the estimation of $V_{\pi/2}$ and V_π over time, resulting in an average of (2.601±0.002)V and (5.202±0.004)V. These values are then used to make the time bin qubits in several cardinal states. The results in Fig. 4.8b also show the stability of these values, confirming the reproducibility. Given that V_π exceeds the maximum amplitude that the HDAWG can provide, we use a phase modulator pulse per bin for this case.

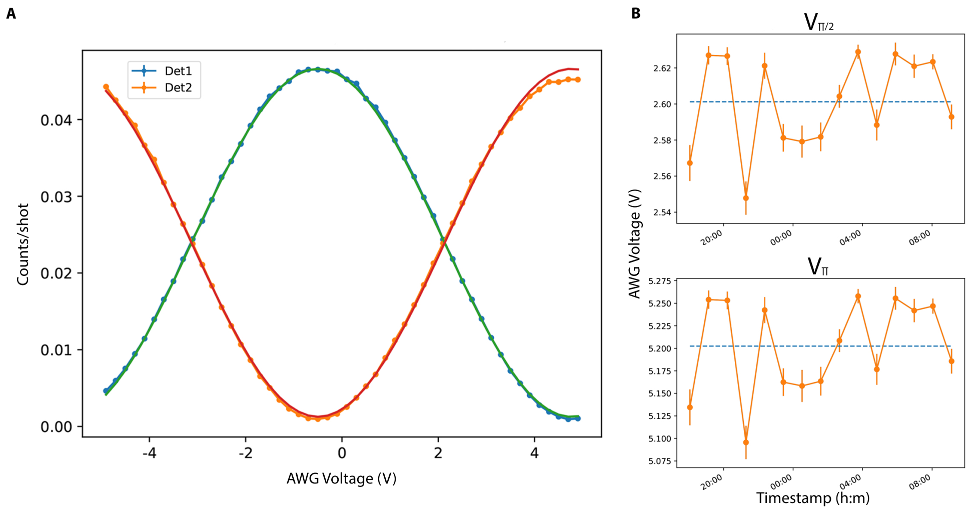


Figure 4.8: a) Data fit to extract the values of V_π and $V_{\pi/2}$. The solid lines represent the fitted curve per detector. b) Stability measurement for V_π and $V_{\pi/2}$. The dashed lines represent the average value.

4.5.8. Devices and experimental monitoring

As stated in the main text, the NV setup is “Alice" of [10, 11]. However, some devices have been replaced. In particular, in this work we employed the Zurich Instruments HDAWG as an arbitrary waveform generator, and the PicoQuant MultiHarp as timetagger. All the other devices remained unchanged.

The experiments, both the TPQI and the teleportation, were running remotely, at a distance of up to 7745km between the scientists and the setup. This shows that the setup is robust over time and automated experimental monitoring is crucial. Here we report the list of automatization routines that have been implemented using the software environment in Ref.[50].

- For the TPQI experiment only: auto-relocking system for the 795nm laser. During the TPQI experiment, the 795nm laser wavelength was locked to an external cavity. A homemade background program detects when the laser frequency drifts and about to go out of lock and adjusts the piezo voltage of the laser to bring the desired spectral mode back. For the teleportation experiment, the laser was locked to a wavemeter with a constant PID loop running to keep the desired wavelength. It is important to notice that the wavelength of the 795nm laser did not change between the two experiments, as the same wavemeter was monitoring the wavelength during the TPQI experiment.

- The system is automatically calibrated over time. In particular, the calibration is targeted at the laser power, the position of the NV with respect to the objective, in such a way as to maximize the fluorescence under the excitation using green light in the Phonon-Side Band (PSB). Small drifts in the optics of the NV setup are compensated by a Python-controlled deformable mirror that is included in the Zero-Phonon Line (ZPL) path. The calibration maximizes the fluorescence in the ZPL when the green light is on. A system of automated waveplates minimizes the leakage of pulse light in the ZPL. The bias voltage of the EOM for the NV optical π-pulses and of the intensity modulator for the 795nm pulses is also optimized during the experiments to minimize leakage.
- A set of control scripts checks for frequency shifts of converted light and all the lasers involved with "real-time" data filtering. In particular, when the control scripts detect an anomaly in one of the frequencies monitored via the wavemeter, a flag is raised and the ongoing measurement is tagged as failed and discarded.
- For teleportation experiment only: calibration of the ratio mean-photon number/counts per shot of the NV every 5 datasets taken. The variable optical attenuator is controlled via the Micro-Controller Unit (MCU). In this way, it is also possible to compensate for drifts in the mean-photon number as well as in the conversion setup that might cause lower efficiency during the experiment.

The experimental rate of teleportation, including the overhead time due to the experimental monitoring and calculated after time-filtering (as discussed in the main text) is ~0.6mHz.

References

[1] H. J. Kimble. "The quantum internet". In: *Nature* 453.7198 (June 2008), pp. 1023–1030. DOI: 10.1038/nature07127.

[2] S. Wehner, D. Elkouss, and R. Hanson. "Quantum internet: A vision for the road ahead". In: *Science* 362.6412 (Oct. 2018). DOI: 10.1126/science.aam9288.

[3] E. Togan et al. "Quantum entanglement between an optical photon and a solid-state spin qubit". In: *Nature* 466.7307 (Aug. 2010), pp. 730–734. DOI: 10.1038/nature09256.

[4] P. Lodahl. "Quantum-dot based photonic quantum networks". In: *Quantum Science and Technology* 3.1 (Oct. 2017), p. 013001. DOI: 10.1088/2058-9565/aa91bb.

[5] M. Bock, P. Eich, S. Kucera, M. Kreis, A. Lenhard, C. Becher, and J. Eschner. "High-fidelity entanglement between a trapped ion and a telecom photon via quantum frequency conversion". In: *Nature Communications* 9.1 (May 2018), p. 1998. DOI: 10.1038/s41467-018-04341-2.

[6] M. Ruf, N. H. Wan, H. Choi, D. Englund, and R. Hanson. "Quantum networks based on color centers in diamond". In: *Journal of Applied Physics* 130.7 (Aug. 2021), p. 070901. DOI: 10.1063/5.0056534.

[7] P.-J. Stas et al. "Robust multi-qubit quantum network node with integrated error detection". In: *Science* 378.6619 (Nov. 2022), pp. 557–560. DOI: 10.1126/science.add9771.

[8] D. Lago-Rivera, J. V. Rakonjac, S. Grandi, and H. d. Riedmatten. "Long distance multiplexed quantum teleportation from a telecom photon to a solid-state qubit". In: *Nature Communications* 14.1 (Apr. 2023), p. 1889. DOI: 10.1038/s41467-023-37518-5.

[9] J.-L. Liu et al. *A multinode quantum network over a metropolitan area.* Aug. 2023.

[10] M. Pompili et al. "Realization of a multinode quantum network of remote solid-state qubits". In: *Science* 372.6539 (Apr. 2021). Publisher: American Association for the Advancement of Science, pp. 259–264. DOI: 10.1126/science.abg1919.

[11] S. L. N. Hermans, M. Pompili, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson. "Qubit teleportation between non-neighbouring nodes in a quantum network". In: *Nature* 605.7911 (May 2022), pp. 663–668. DOI: 10.1038/s41586-022-04697-y.

[12] M. W. Doherty, N. B. Manson, P. Delaney, F. Jelezko, J. Wrachtrup, and L. C. Hollenberg. "The nitrogen-vacancy colour centre in diamond". In: *Physics Reports* 528.1 (July 2013), pp. 1–45. DOI: 10.1016/j.physrep.2013.02.001.

[13] L. Childress and R. Hanson. "Diamond NV centers for quantum computing and quantum networks". In: *MRS Bulletin* 38.2 (Feb. 2013), pp. 134–138. DOI: 10.1557/mrs.2013.20.

[14] C. W. Thiel, N. Sinclair, W. Tittel, and R. L. Cone. "Tm3+:Y3Ga5O12 Materials for Spectrally Multiplexed Quantum Memories". In: *Physical Review Letters* 113.16 (Oct. 2014), p. 160501. DOI: 10.1103/PhysRevLett.113.160501.

[15] N. Sinclair, K. Heshami, C. Deshmukh, D. Oblak, C. Simon, and W. Tittel. “Proposal and proof-of-principle demonstration of non-destructive detection of photonic qubits using a Tm:LiNbO3 waveguide”. In: *Nature Communications* 7.1 (Nov. 2016), p. 13454. DOI: 10.1038/ncomms13454.

[16] M. F. Askarani et al. “Long-Lived Solid-State Optical Memory for High-Rate Quantum Repeaters”. In: *Physical Review Letters* 127.22 (Nov. 2021), p. 220502. DOI: 10.1103/PhysRevLett.127.220502.

[17] Y.-W. Cho, G. T. Campbell, J. L. Everett, J. Bernu, D. B. Higginbottom, M. T. Cao, J. Geng, N. P. Robins, P. K. Lam, and B. C. Buchler. “Highly efficient optical quantum memory with long coherence time in cold atoms”. In: *Optica* 3.1 (Jan. 2016), pp. 100–107. DOI: 10.1364/OPTICA.3.000100.

[18] R. Zhao, Y. O. Dudin, S. D. Jenkins, C. J. Campbell, D. N. Matsukevich, T. a. B. Kennedy, and A. Kuzmich. “Long-lived quantum memory”. In: *Nature Physics* 5.2 (Feb. 2009). Number: 2 Publisher: Nature Publishing Group, pp. 100–104. DOI: 10.1038/nphys1152.

[19] W. Rosenfeld, S. Berner, J. Volz, M. Weber, and H. Weinfurter. “Remote Preparation of an Atomic Quantum Memory”. In: *Physical Review Letters* 98.5 (Feb. 2007), p. 050504. DOI: 10.1103/PhysRevLett.98.050504.

[20] L. Heller, P. Farrera, G. Heinze, and H. de Riedmatten. “Cold-Atom Temporally Multiplexed Quantum Memory with Cavity-Enhanced Noise Suppression”. In: *Physical Review Letters* 124.21 (May 2020), p. 210504. DOI: 10.1103/PhysRevLett.124.210504.

[21] K. Azuma, S. E. Economou, D. Elkouss, P. Hilaire, L. Jiang, H.-K. Lo, and I. Tzitrin. “Quantum repeaters: From quantum networks to the quantum internet”. In: *Reviews of Modern Physics* 95.4 (Dec. 2023), p. 045006. DOI: 10.1103/RevModPhys.95.045006.

[22] P. Drmota et al. *Verifiable blind quantum computing with trapped ions and single photons.* May 2023. DOI: 10.48550/arXiv.2305.02936.

[23] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. “Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels”. In: *Physical Review Letters* 70.13 (Mar. 1993), pp. 1895–1899. DOI: 10.1103/PhysRevLett.70.1895.

[24] D. Bouwmeester, J.-W. Pan, K. Mattle, M. Eibl, H. Weinfurter, and A. Zeilinger. “Experimental quantum teleportation”. In: *Nature* 390 (1997), pp. 575–579. DOI: https://doi.org/10.1038/37539.

[25] K. Heshami, D. G. England, P. C. Humphreys, P. J. Bustard, V. M. Acosta, J. Nunn, and B. J. Sussman. “Quantum memories: emerging applications and recent advances”. In: *Journal of Modern Optics* 63.20 (Nov. 2016), pp. 2005–2028. DOI: 10.1080/09500340.2016.1148212.

[26] Y. Lei, F. K. Asadi, T. Zhong, A. Kuzmich, C. Simon, and M. Hosseini. “Quantum optical memory for entanglement distribution”. In: *Optica* 10.11 (Nov. 2023), pp. 1511–1528. DOI: 10.1364/OPTICA.493732.

[27] M. Afzelius, C. Simon, H. de Riedmatten, and N. Gisin. "Multimode quantum memory based on atomic frequency combs". In: *Physical Review A* 79.5 (May 2009), p. 052329. DOI: 10.1103/PhysRevA.79.052329.

[28] J. P. Covey, H. Weinfurter, and H. Bernien. "Quantum networks with neutral atom processing nodes". In: *npj Quantum Information* 9.1 (Sept. 2023), pp. 1–12. DOI: 10.1038/s41534-023-00759-9.

[29] M. Lipka, M. Mazelanik, A. Leszczyński, W. Wasilewski, and M. Parniak. "Massively-multiplexed generation of Bell-type entanglement using a quantum memory". In: *Communications Physics* 4.1 (Mar. 2021), pp. 1–10. DOI: 10.1038/s42005-021-00551-1.

[30] F. Bussieres, N. Sangouard, M. Afzelius, H. de Riedmatten, C. Simon, and W. Tittel. "Prospective applications of optical quantum memories". In: *Journal of Modern Optics* 60.18 (Oct. 2013), pp. 1519–1537. DOI: 10.1080/09500340.2013.856482.

[31] N. Sangouard, C. Simon, H. de Riedmatten, and N. Gisin. "Quantum repeaters based on atomic ensembles and linear optics". In: *Reviews of Modern Physics* 83.1 (Mar. 2011), pp. 33–80. DOI: 10.1103/RevModPhys.83.33.

[32] P. Kumar. "Quantum frequency conversion". In: *Optics Letters* 15.24 (Dec. 1990), pp. 1476–1478. DOI: 10.1364/OL.15.001476.

[33] C. K. Hong, Z. Y. Ou, and L. Mandel. "Measurement of subpicosecond time intervals between two photons by interference". In: *Physical Review Letters* 59.18 (Nov. 1987), pp. 2044–2046. DOI: 10.1103/PhysRevLett.59.2044.

[34] R. Loudon. *The Quantum Theory of Light*. Clarendon Press, 1983.

[35] H. Bernien et al. "Heralded entanglement between solid-state qubits separated by three metres". In: *Nature* 497.7447 (May 2013), pp. 86–90. DOI: 10.1038/nature12016.

[36] Z.-H. Wang, G. de Lange, D. Ristè, R. Hanson, and V. V. Dobrovitski. "Comparison of dynamical decoupling protocols for a nitrogen-vacancy center in diamond". In: *Physical Review B* 85.15 (Apr. 2012), p. 155204. DOI: 10.1103/PhysRevB.85.155204.

[37] M. Ruf. "Cavity-enhanced quantum network nodes in diamond". PhD thesis. Delft University of Technology, 2021.

[38] C. Hepp et al. "Electronic Structure of the Silicon Vacancy Color Center in Diamond". In: *Physical Review Letters* 112.3 (Jan. 2014), p. 036405. DOI: 10.1103/PhysRevLett.112.036405.

[39] M. K. Bhaskar et al. "Quantum Nonlinear Optics with a Germanium-Vacancy Color Center in a Nanoscale Diamond Waveguide". In: *Physical Review Letters* 118.22 (May 2017), p. 223603. DOI: 10.1103/PhysRevLett.118.223603.

[40] C. T. Nguyen et al. "Quantum Network Nodes Based on Diamond Qubits with an Efficient Nanophotonic Interface". In: *Physical Review Letters* 123.18 (Oct. 2019), p. 183602. DOI: 10.1103/PhysRevLett.123.183602.

[41] A. E. Rugar, S. Aghaeimeibodi, D. Riedel, C. Dory, H. Lu, P. J. McQuade, Z.-X. Shen, N. A. Melosh, and J. Vučković. "Quantum Photonic Interface for Tin-Vacancy Centers in Diamond". In: *Physical Review X* 11.3 (July 2021), p. 031021. DOI: 10.1103/PhysRevX.11.031021.

[42] D. M. Lukin, M. A. Guidry, and J. Vučković. "Integrated Quantum Photonics with Silicon Carbide: Challenges and Prospects". In: *PRX Quantum* 1.2 (Dec. 2020), p. 020102. DOI: 10.1103/PRXQuantum.1.020102.

[43] D. B. Higginbottom et al. "Optical observation of single spins in silicon". In: *Nature* 607.7918 (July 2022), pp. 266–270. DOI: 10.1038/s41586-022-04821-y.

[44] S. Liu et al. *Violation of Bell inequality by photon scattering on a two-level emitter*. June 2023. DOI: 10.48550/arXiv.2306.12801.

[45] M. Schäfer, B. Kambs, D. Herrmann, T. Bauer, and C. Becher. "Two-stage, low noise quantum frequency conversion of single photons from silicon-vacancy centers in diamond to the telecom C-band". In: *Advanced Quantum Technologies* 8.2 (Feb. 2025). arXiv:2307.11389 [quant-ph], p. 2300228. DOI: 10.1002/qute.202300228.

[46] E. Bersin et al. "Telecom Networking with a Diamond Quantum Memory". In: *PRX Quantum* 5.1 (Jan. 2024), p. 010303. DOI: 10.1103/PRXQuantum.5.010303.

[47] A. Padrón-Brito, J. Lowinski, P. Farrera, K. Theophilo, and H. de Riedmatten. "Probing the indistinguishability of single photons generated by Rydberg atomic ensembles". In: *Physical Review Research* 3.3 (Sept. 2021), p. 033287. DOI: 10.1103/PhysRevResearch.3.033287.

[48] S. Massar and S. Popescu. "Optimal Extraction of Information from Finite Quantum Ensembles". In: *Physical Review Letters* 74.8 (Feb. 1995), pp. 1259–1263. DOI: 10.1103/PhysRevLett.74.1259.

[49] H. P. Specht, C. Nölleke, A. Reiserer, M. Uphoff, E. Figueroa, S. Ritter, and G. Rempe. "A single-atom quantum memory". In: *Nature* 473.7346 (May 2011), pp. 190–193. DOI: 10.1038/nature09997.

[50] I. T. Raa et al. *QMI - Quantum Measurement Infrastructure, a Python 3 framework for controlling laboratory equipment*. Oct. 2023.

5

DESIGN AND DEMONSTRATION OF AN OPERATING SYSTEM FOR EXECUTING APPLICATIONS ON QUANTUM NETWORK NODES

People in their right minds never take pride in their talents.

Harper Lee - To Kill a Mockingbird

C. Delle Donne*, M. Iuliano*, B. van der Vecht*, G.M. Ferreira, H. Jirovská, T.J.W. van der Steenhoven, A. Dahlberg, M. Skrzypczyk, D. Fioretto, M. Teller, P. Filippov, A. R.-P. Montblanch, J. Fischer, H.B. van Ommen, N .Demetriou, D. Leichtle, L. Music, H. Ollivier, I. te Raa, W. Kozlowski, T.H. Taminiau, P. Pawełczak, T.E. Northup, R. Hanson, S. Wehner

The goal of future quantum networks is to enable new internet applications that are impossible to achieve using solely classical communication[1–3]. Up to now, demonstrations of quantum network applications[4–6] and functionalities[7–12] on quantum processors have been performed in ad-hoc software that was specific to the experimental setup, programmed to perform one single task (the application experiment) directly into low-level control devices using expertise in experimental physics. Here, we report on the design and

The results of this chapter are published in Nature 639 (2025)
* These authors contributed equally

implementation of the first architecture capable of executing quantum network applications on quantum processors in platform-independent high-level software. We demonstrate the architecture's capability to execute applications in high-level software, by implementing it as a quantum network operating system – QNodeOS – and executing test programs including a delegated computation from a client to a server[13] on two quantum network nodes based on nitrogen-vacancy (NV) centers in diamond[14, 15]. We show how our architecture allows us to maximize the use of quantum network hardware, by multitasking different applications on a quantum network for the first time. Our architecture can be used to execute programs on any quantum processor platform corresponding to our system model, which we illustrate by demonstrating an additional driver for QNodeOS for a trapped-ion quantum network node based on a single $^{40}\mathrm{Ca}^{+}$ *atom[16]. Our architecture lays the groundwork for computer science research in the domain of quantum network programming, and paves the way for the development of software that can bring quantum network technology to society.*

Figure 5.1: **QNodeOS**. Microcontroller hosting the quantum network operating system. Photo credit: Studio Oostrum.

5.1. INTRODUCTION

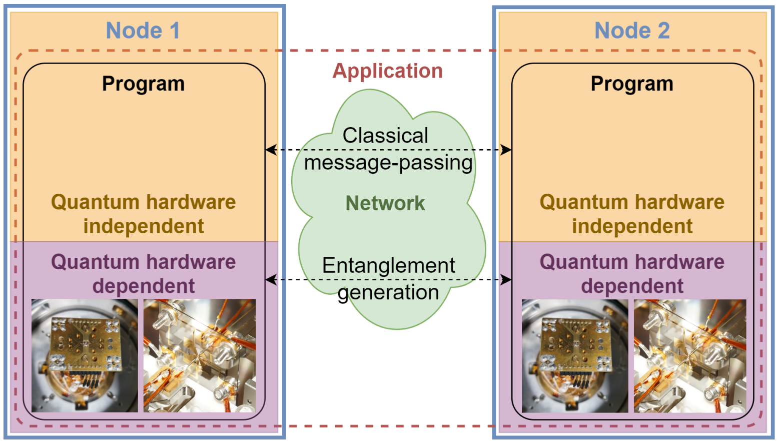
The first quantum networks linking multiple quantum processors as end nodes have recently been realized as physics experiments in laboratories [17–23] and fiber networks [24–26], opening the tantalizing possibility of realizing advanced quantum network applications [2] such as data consistency in the cloud [27], privacy-enhancing proofs of deletion [28], exponential savings in communication [29], or secure quantum computing in the cloud [13, 30]. Demonstrations relied either on ad-hoc software, or chose to establish that hardware parameters were in principle good enough to support a given quantum network application, although the application itself was not realized [6, 31, 32]. While quantum nodes have been linked at the hardware level [17–23, 25, 26, 33, 34], including the design [35–38] and realization [39, 40] of network stacks to manage entanglement generation, a critical innovation required to make quantum networks useful is lacking: an architecture enabling the execution of quantum applications.

It is a major challenge to design and implement an architecture that can enable the execution of arbitrary quantum network applications on quantum processor end nodes (Figure 5.2), while enabling programming in high-level software that neither depends on the underlying quantum hardware, nor requires the programmer to understand the physics of the underlying devices. In the domain of the conventional internet, the possibility of programming arbitrary internet applications in high-level software has led to the realization of radically new communication applications by diverse communities, which had a transformative impact on our society [41]. What's more, the advent of programmable hardware and new application areas sparked novel fields of computer science research and guided further hardware development (e.g. network programming and protocols, distributed systems, internet of things, and more). A similar development is underway in quantum computing, where the availability of high-level programming tools allows a broad participation in developing applications [42].

In realizing the first such system architecture we overcome all challenges below, including both fundamental challenges that are inherent to quantum network applications at any scale, as well as technological challenges that arise from the current state of the art of quantum network hardware.

5.2. DESIGN CONSIDERATIONS AND CHALLENGES

Interactive Classical-Quantum Execution. The execution of quantum network applications requires a continuing interaction between the quantum and classical parts of the execution, including interactions between different programs (Figure 5.2). For example, during secure quantum computing in the cloud [13, 46], the program on the server is waiting for classical messages from a remote client program before continuing the quantum execution at the server. This is in sharp contrast to quantum computing applications, where a quantum application is a single program that can be executed in one batch, without the need to keep quantum states live while waiting for input from other programs. In quantum computing, only relatively low-level and predictable interactions between classical and quantum processing are realized, such as in quantum error correction [47], or mid-circuit measurements [48]. Higher-level classical-quantum interactions in quantum computing [49] do not keep qubits live in memory.



Node 1
Node 2
Application
Program
Program
Classical message-passing
Network
Entanglement generation
Quantum hardware independent
Quantum hardware independent
Quantum hardware dependent
Quantum hardware dependent

Figure 5.2: **Application Paradigm.** A quantum networking application consists of multiple programs, each running on one of the end nodes [43]. An end node is a device in a quantum network that executes user applications. A network stack enables entanglement generation between end nodes over a quantum network (Figure 5.7). The distinct programs at each end node can only interact via (1) quantum communication (e.g. entanglement generation) and (2) classical communication. This allows a programmer to realize security-sensitive applications, but prohibits a global orchestration of the quantum execution, like one might do in (distributed) quantum computing [44] in which a single quantum program is executed on multiple nodes. Our architecture allows programs to be written in high-level quantum hardware independent software, and executed on a quantum hardware independent system that controls a hardware dependent system (QDevice, Figure 5.3) such as a nitrogen-vacancy (NV) center node with a diamond chip (photo taken by authors, left images) or a trapped-ion quantum node [45] (right images). These platforms constitute physically very different QDevice systems, but can both be programmed by our architecture.

We assume that the programs are divided into classical and quantum blocks of instructions (by a programmer or a compiler). Classical blocks consist of local classical operations executed on a conventional classical processor, as well as networked classical operations (i.e. sending messages to remote nodes) executed using network devices. Quantum blocks consist of local quantum operations (gates, measurements, classical control logic), as well as networked quantum operations (entanglement generation) executed on quantum hardware. A single quantum block, in essence, corresponds to a program in quantum computing, and may contain simple classical control logic, such as for the purpose of mid-circuit measurements [48].

Different Hardware Platforms. Interfacing with different hardware platforms presents technological challenges: currently, a clear line between software and hardware has not been defined, and the low-level control of present-day quantum processor hardware has been built to conduct physics experiments. Early microarchitectures [50, 51] and operating systems [52, 53] for quantum computing do not address the execution of quantum network applications. We thus have to define a hardware abstraction layer (HAL), capable of interfacing with quantum network processors, including present-day setups.

Timescales. It is a fundamental challenge that different parts of such a system operate at vastly different timescales. For nodes separated by hundreds of kilometers, the duration of network operations is in the millisecond (ms) regime, and some applications [2] need significant local classical processing (ms). In contrast, the time to execute quantum operations on processing nodes is in the regime of microseconds (μs), and the low-level control (including timing synchronization between neighboring nodes to generate entanglement [54]) requires nanosecond (ns) precision.

Memory Lifetimes. Present-day quantum network nodes have short coherence times, posing a technological challenge to ensure operations are executed within the timeframe allowed by the quantum memory.

Scheduling Local and Network Operations. Entanglement is a key resource for quantum network applications [2]. In contrast to classical networking, entanglement is a form of stateful connection already at the physical layer where both nodes hold one qubit. Our architecture should allow for the execution of applications at end nodes, and these may be separated by a large quantum network that facilitates entanglement generation between them. This can be achieved by the implementation of a network stack [35, 36]. A technological challenge arises in the integration of such a network stack with the application stack of QNodeOS, when employing heralded entanglement generation at the physical layer [35] (as done in all current demonstrations linking quantum processors [25, 39]). Heralded entanglement generation requires agreement between neighboring network nodes to trigger entanglement generation in precise time-bins [35], organized into a network schedule [55] that dictates when nodes make entanglement. Such a schedule could be set by a centralized controller [55], or by a distributed protocol (see e.g. [35]).

It is a technological challenge to manage the interdependencies between the schedule of local operations, and the networked operations, since in all current processing node implementations [22, 56], entanglement generation cannot be performed simultaneously with local operations [22, 57]. While interdependencies may be mitigated in the future [58], this implies that we cannot schedule (i.e. decide when to execute) the execution of local quantum operations independently of the network schedule.

Multitasking. When executing (quantum) network applications, one node is typically idle while waiting for the other node before it can continue execution. For example, a client program may need to wait for a server to finish processing and send a message. It is hence a fundamental challenge how we can increase the utility of the system by performing multitasking [59, 60], that is, allowing the concurrent execution of several programs at once to make use of idle times. Consequently, there is a need for managing state and resources for multiple independent programs, including processes, quantum memory management, and entanglement requests.

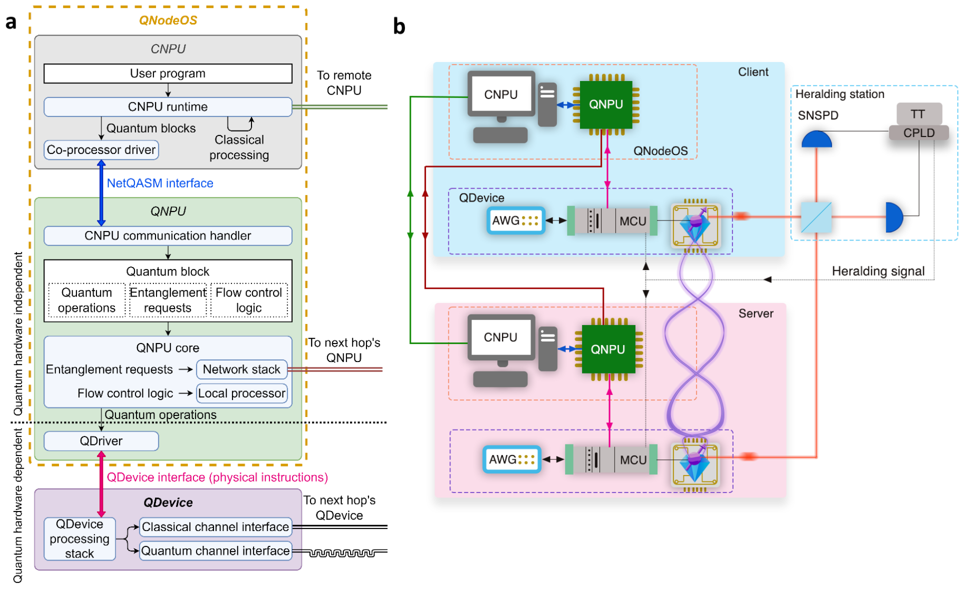


Figure 5.3: **QNodeOS architecture. (a)** QNodeOS consists of a Classical Network Processing Unit (CNPU) and a Quantum Network Processing Unit (QNPU, classical system). QNodeOS controls a QDevice (quantum hardware and low-level classical control). **(b)** Schematic of our implementation of QNodeOS on a two-node setup where both QDevices control a single qubit in a diamond nitrogen-vacancy (NV) center. The CNPU is implemented on a general-purpose PC, and the QNPU on an embedded system, connected via Gigabit Ethernet (blue). The QNPU connects to its QDevice via a serial peripheral interface (SPI, pink). The two QNPUs (brown), and the two CNPUs (green) connect to each other via Gigabit Ethernet. The setup is based on [39] with two QDevices (including arbitrary waveform generators (AWG) and microcontroller units (MCU); QDevices communicating over a classical DIO interface) and a heralding station composed by a balanced 50:50 beam-splitter (whose output ports are connected to superconducting nanowire single-photon detectors (SNSPD) via optical fibers (red)), a TimeTagger (TT), and a Complex Programmable Logic Device (CPLD) that heralds the entanglement generation between QDevices and sends a classical message to the MCU.

5.3. Architecture

We divide the architecture logically into three main components (Figure 5.3, Section 5.6): The Classical Network Processing Unit (CNPU) is the logical element responsible for starting the execution of programs, and the execution of classical code blocks; the Quantum Network Processing Unit (QNPU, realized on classical hardware) is the logical element responsible for governing the execution of the quantum code blocks; The CNPU and QNPU together form QNodeOS and control the QDevice, which is responsible for executing any quantum operations (gates, measurements, entanglement generation at the physical layer [35]) on the quantum hardware. Upon starting the execution, the CNPU creates a process (a well-known concept in classical operating systems [61, 62]) on the CNPU (a CNPU process), registers the program on the QNPU (via the QNPU's end node application programming interface (API), Section 5.8.2), which, in turn, creates its own associated QNPU process (including context such as process owner, ID, process state and priority). QNodeOS also defines kernel processes on the QNPU, which are similar to user processes, but are created when the system starts (on boot). The CNPU sends quantum blocks to the QNPU in the form of NetQASM subroutines [43]. Classical control logic in quantum blocks is executed by the QNPU processor. Quantum gates and measurements (from any QNPU process) and entanglement instructions (from the network process) are delegated to the QDevice by submitting physical instructions (Section 5.6), after which the QDevice responds back to the QNPU with the result of the instruction (Section 5.8.6).

To enable different hardware platforms, we introduce a QDriver realizing the HAL for any hardware corresponding to our minimal QDevice system model (Section 5.6). The QDriver is the only hardware-dependent element of the architecture, and is responsible for translating quantum operations, expressed in NetQASM [43], into platform dependent (streams of) physical instructions to the underlying QDevice. We realize a QDriver for the trapped-ion system of [45, 63], and one for NV centers in diamond based on the system of [7, 22, 39]. We validate the trapped-ion QDriver (Figure 5.6) by implementing and verifying a set of single-qubit gate operations (Section 5.6), and the QDriver on the NV system as part of the full stack system evaluation (see below). To allow for different timescales, we logically divide the architecture into CNPU, QNPU and QDevice which can thus be realized at different timing scale granularities. In our proof-of-concept implementation, we realize the CNPU and QNPU on different devices, reflecting the ms timescales of communication between distant nodes (Section 5.6).

Ensuring the necessary interactivity requires architectural as well as implementation choices: as programs may depend on messages from remote nodes, the architecture needs to be able to dynamically handle both classical and quantum blocks, even if not known at runtime. Consequently, it is not possible to preload all quantum blocks of the program into the low-level controller of the QDevice ahead of time as done in previous physics experiments. Instead, in our system model the QDevice is capable of executing individual physical instructions similar to a classical CPU. Consequently, the QNPU is continuously ready to receive new NetQASM subroutines from the CNPU, and the QDevice can continuously receive and respond to physical instructions from the QNPU (Section 5.6).

In our NV QDevice implementation, we address the challenge of interactivity by interleaving specific user-requested pulse sequences (realizing physical instructions sent from QNodeOS) and dynamical decoupling (DD) sequences (protecting quantum memory from decoherence) in an arbitrary waveform generator (AWG) [64]. The DD sequences extend qubit coherence times up to $T_{\mathrm{coh}} = 13(2)$ ms, while arbitrary physical instructions can be handled by triggering the corresponding pulse sequence, without knowing them in advance (Section 5.6).

To integrate local operations with the network schedule, our architecture first introduces a QNPU scheduler that can choose which of the ready processes is assigned to the local processor (Figure 5.3, Section 5.6) and QDevice. This allows interleaving the execution of different processes directly on the QNPU without incurring delays on the timescale of the CNPU (ms), addressing the challenge of short coherence times. In our implementation, we choose to schedule QNPU processes using a priority based non-preemptive scheduler [65], due to limited quantum memory lifetimes, which make it undesirable to pre-empt and temporarily store quantum states while halting the execution. Second, we realize a network process as a kernel process, which manages entanglement generation using the network stack [35, 36] (implemented in [39] without the ability to execute network applications). While our architecture can work with any way of setting a network schedule, in our implementation we choose to determine the schedule using a time-division multiple access (TDMA) controller [55], allowing the schedule to be centrally optimized to mitigate present-day memory decoherence. The network process handles entanglement requests submitted by user processes, coordinates entanglement generation with the rest of the network via the TDMA controller, interacts with the QDevice and eventually returns entangled qubits to user processes. User processes enter the waiting state when they need entanglement, and become ready again once entanglement was delivered. The network process has the highest scheduling priority, and is consequently given precedence over the execution of any local quantum operations. We remark local operations may still be executed during time-bins already occupied by the network schedule, if a running non-preemptable user process prevents the network process from running, as we indeed observe in our evaluation.

To increase utility, QNodeOS allows multiple programs to be run concurrently, using the QNPU scheduler from above to enable multitasking [59, 60] user processes on the QNPU itself. The QNPU hence needs to keep context for each process, including a virtual quantum memory space (as in classical operating systems [66]). Similar to classical memory management systems [67], a quantum memory management unit (QMMU) on the QNPU manages qubit allocations from processes, and translates virtual qubit addresses in NetQASM subroutines to physical addresses in the QDevice. This allows flexibility in translating a virtual qubit address to: (1) a different physical qubit address over time, allowing qubits to be rearranged transparently in the physical memory in the future, or (2) a logical qubit address, when QNodeOS is executed on top of a processor employing quantum error correction [47] (Section 5.6). Entanglement generation between different pairs of processes at remote nodes are distinguished by Entanglement Request (ER) sockets, inspired by classical sockets [68, 69], which are established once a user process requests entanglement from the network process. In our implementation, processes of the same priority are scheduled first-come-first-served [67], where the total schedule of the program in our implementation is dependent both on the schedule on the CNPU as well as the QNPU (Section 5.6).

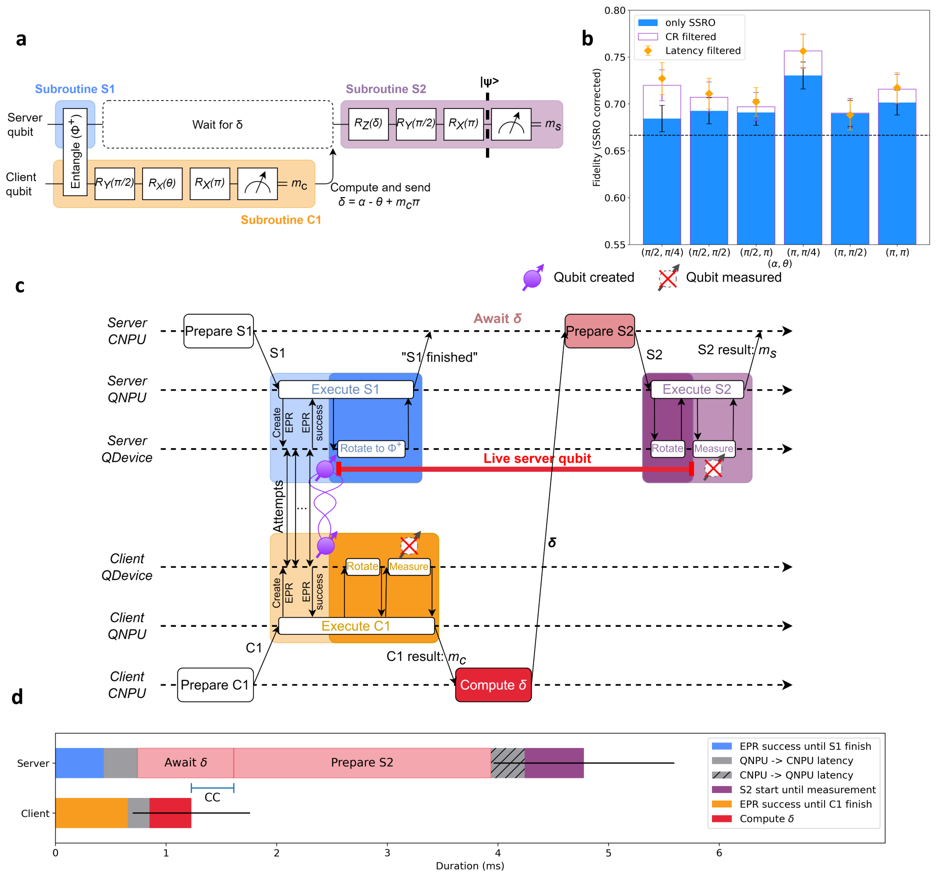


Figure 5.4: **Delegated computation between two NV center nodes using QNodeOS. (a)** Delegated Quantum Computation (DQC) circuit (effective computation: single-qubit rotation $R_Z(\alpha)$, Section 5.6). The DQC application consists of k repetitions of this circuit (varying measurement bases for tomography on $|\psi\rangle$) realized by two programs: the DQC-client program (client node, repeating the sequence "quantum block (C1, orange) – classical block (computing δ)" k times), and the DQC-server program (server node, repeating "quantum block (S1, blue) - classical block (receiving δ) – quantum block (S2, purple)" k times). Client and server produce an entangled pair $|\Phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ (S1 and first part of C1). The client performs local gates and a measurement ("destroying" qubit), resulting in outcome bit m_c (rest of C1). Client computes δ as function of m_c and DQC parameters $\alpha \in [0, 2\pi)$ and $\theta \in [0, 2\pi)$, and sends δ to server (classical message). Meanwhile the server keeps its qubit coherent (alive). Upon receiving δ, the server applies gates depending on δ, resulting in single-qubit state $|\psi\rangle$ (S2) depending only on α and θ. **(b)** Experimental results of executing DQC for 6 different sets of (α, θ) parameters ($k = 1200$, i.e. 7200 executions of the circuit of 5.4a). The fidelity of the resulting server state to the target state $|\psi\rangle$ is estimated using single-qubit tomography (1200 measurement results per data point), and corrected for known tomography errors (SSRO, blue), and post-selected for Charge-Resonance (CR) check validation (purple), and post-selected for latencies (orange) (Section 5.6). **(c)** Sequence diagram including the interaction CNPU-QNPU-QDevice for one execution of the DQC circuit of 5.4a on QNodeOS (repeated $k = 1200$ times in each experiment) (time flows to the right; not to scale). CNPUs prepare NetQASM subroutines (C1, S1, S2), and send them to their respective QNPUs. CNPUs also do classical computation (computing δ) and communication (message containing δ). QNPUs execute subroutines, sending physical instructions to their QDevices. Entanglement is generated by QDevices doing a batch of attempts, resulting in the heralding of a two-qubit entangled state (Bell pair) rotated to $|\Phi^+\rangle$ by the server. **(d)** Processing times and latencies while server qubit is live (time frame red line 3c, averaged over all 7200 circuit executions except executions with latency spikes, see Section 5.6), including CNPU-QNPU communication latencies, CNPU processing on both nodes and client-server communication latency (CC) (average total of ~ 4.8(±0.8) ms, error bars for the sum of individual segments (variance per segment in Section 5.10.6).

5.4. DEMONSTRATIONS

Delegated Computation. We first validate our architecture and implementation by the first successful execution of an arbitrary — i.e. not preloaded — execution of a quantum network application in high-level software on quantum processors. We implement QNodeOS on a two-node setup of NV centers using one qubit per node (Figure 5.3, Section 5.6). We choose to execute an elementary form of delegated quantum computation (DQC) [13] from a client to a server, because the client and server programs jointly realize repetitions of a circuit (Figure 5.4a) that triggers all parts of our system (Figure 5.4c). We first verify that the quantum result (fidelity) was found to be above the classical bound [70] $> 2/3$, which verifies that QNodeOS can successfully handle interactive applications consisting of entanglement generation, millisecond-scale memory lifetimes, and classical message passing. The non-perfect fidelity (Figure 5.4b) comes mainly from two sources: a noisy entangled state with fidelity 0.72(2) (quantum hardware limitation), and decoherence in the server qubit (depending on T_{coh}) due to waiting for several milliseconds (classical software latencies, Figure 5.4d). We proceed to characterize latencies. As expected, we find that the duration that the server qubit must remain alive is dominated ($> 50\%$) by processing in the CNPU, which could be improved by caching the preparation of S2, and implementing the CNPU and QNPU on one board (Outlook). We observe that CNPU processing time varies significantly (standard deviation 30%, Section 5.10.6), due to limited scheduling control over CNPU processes (Section 5.6). Using an a priori estimate of what delays lead to too low a quality of execution (i.e. delays that are too long for the server qubit to be stored with sufficiently high quality), we discard application iterations in which the CNPU latencies spiked by more than 8.95 ms. This lead to discarding of 2% of iterations in post-processing (Section 5.6).

Demonstration of Multitasking. We also validate QNodeOS's multitasking capability by the first concurrent execution of two quantum applications on a quantum network: the DQC application, and a single-node local gate tomography (LGT) application on the client (Figure 5.5a). The two programs for the client are started in the CNPU at the same time (two CNPU processes, subject to CNPU scheduler), which means that the QNPU continuously receives subroutines for both programs from the CNPU (two QNPU processes and corresponding subroutines, subject to QNPU scheduler). This leads to a multitasking challenge directly on the QNPU to schedule the different subroutines received (Figure 5.5b). Since the client has only one qubit, the multitasking of DQC and LGT never results in both programs having a quantum state alive on the client; therefore, multitasking should not affect the fidelity of LGT. We observe interleaved execution of DQC quantum blocks and LGT quantum blocks on the client node (Figure 5.5b). The LGT application produces a quantum result (fidelity, Figure 5.5c) equal to that in the scenario where we run LGT on its own (not interleaved by DQC circuit executions), as expected.

We further test multitasking by scaling up the number of programs executed concurrently on the client node, up to 5 DQC and 5 LGT programs running at the same time on the client. The interleaved execution of subroutines of different programs increases device utilization (fraction of time spent on executing physical instructions) on the client QDevice compared to the same scenario but with multitasking disabled (Figure 5.5d). As expected, we observe that LGT subroutines were scheduled to be executed in between DQC subroutines, resulting in lower client QDevice idle time. When multitasking 1 DQC and 1 LGT program, we observe 1 or 2 subroutines in between DQC iterations in most cases (LGT subroutine duration 2.4 ms, Section 5.11.3). We observe cases where both server and client QDevice remain idle, which could be improved in part by smarter CNPU-QNPU scheduling algorithms: (1) both the client and server wait until the start of the next network schedule time-bin (time-bin length 10 ms) (2) the client QNPU finishes a subroutine for user process P, but must wait until the CNPU sends the next subroutine for P (up to 150 ms for 1 DQC and 1 LGT program, but less (up to only 8 ms) when more applications are running, since there are more CNPU processes independently submitting subroutines), (3) the client is ready to perform entanglement generation for DQC, but the next time-bin starts only at some future time t, preventing activation of the network process. The scheduler activates a user process which runs a LGT circuit, which completes at some time $> t$, delaying the start of the DQC network process, even though the server node was ready at t.

5.5. OUTLOOK

We designed and implemented the first architecture allowing high-level programming and execution of quantum network applications. Our architecture does not depend on the distance or connectivity between the end nodes, as long as the network stack enables the use of a quantum network to generate entanglement between them. To deploy our system onto nodes separated by several kms, one possible improvement to the implementation of our architecture would be to realize the CNPU and the QNPU on two devices on a single system board, ideally with mutual access to a shared memory to avoid ms delays in their communication. Such a merge would also allow the definition of a joint classical-quantum executable and processes, opening further doors to reduce latencies by a better scheduling control. Our architecture could also be used to distribute a quantum computing program over multiple quantum processors by submitting jobs as NetQASM subroutines to the QNPU of each node.

Our work provides a framework for a new domain of computer science research into programming quantum network applications on quantum processors including: novel real-time [71] scheduling algorithms for classical-quantum processes, compile methods for quantum network applications [43], or novel programming language concepts including entanglement to make software development even easier, thus advancing the vision to make quantum network technology broadly available.

5.6. METHODS

QDevice Model. The QDevice includes a physical quantum device, which can initialize and store quantum bits (qubits) which are individually identified by a physical address, apply quantum gates, measure qubits, and create entanglement with QDevices on other nodes (either entangle-and-measure, or entangle-and-keep [35]), either another end node, or an intermediary node in the network. We remark that the ability for two end node QDevices that are not immediate neighbors in the quantum network (but that are separated by other network nodes) to generate entanglement between them relies on the architecture implementing a network layer protocol as part of a network stack [35].

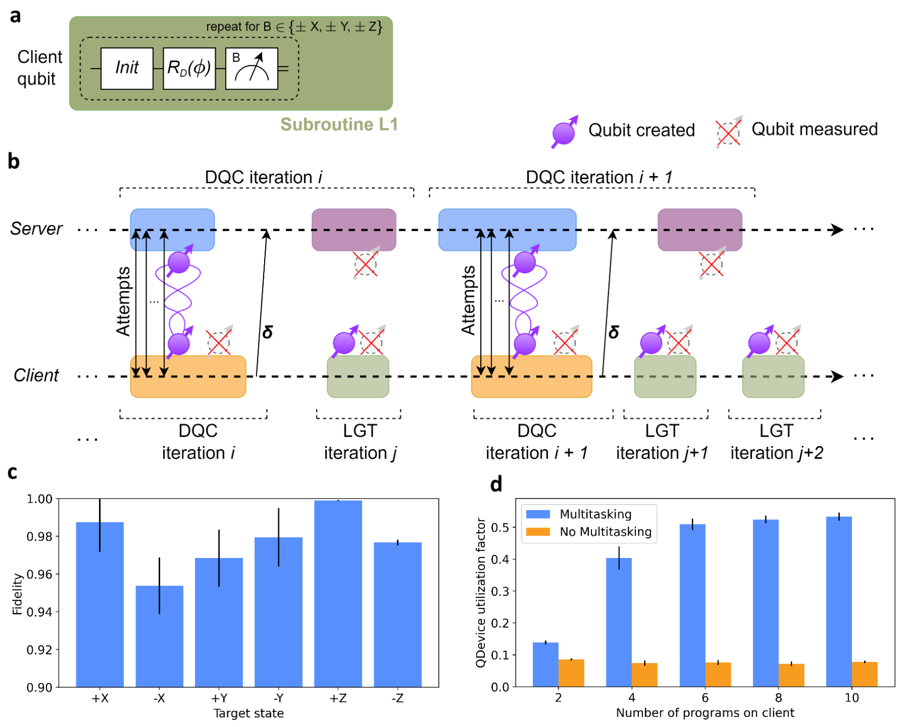


Figure 5.5: **Multitasking experiment on two NV centers with QNodeOS**. **(a)** Local Gate Tomography (LGT) Circuit. A single NetQASM subroutine (L1) executes the following 6 times for different bases $B \in \{\pm X, \pm Y, \pm Z\}$: initialize qubit to $|0\rangle$, rotate around fixed axis $D \in \{X, Y\}$ by angle $|\phi\rangle$, measure in B. The LGT application consists of a single LGT program, which submits subroutine L1 for execution to the QNPU (fixed D and ϕ) k times in succession. **(b)** Example sequence diagram illustrating concurrent execution (multitasking) of the DQC application (Figure 5.4) and the LGT program on the client: time slice in which two DQC circuit repetitions (Figure 5.4a) are realized (2 subroutines on the client (orange), 4 on the server (blue and purple)), and three LGT circuit repetitions (3 subroutines, green). The client QNPU receives subroutines for both the DQC program and the LGT program, which the QNPU scheduler can interleave: While the server executes S2 (purple), the client cannot yet execute the next S1 (orange) since it involves joint entanglement generation. In this idle time, the client can execute a number of LGT subroutines (number can vary). **(c)** Results of multitasking LGT (client) and DQC (on both server and client). For each input pair $(D, \phi) \in \{(X, 0), (X, \pi), (Y, pi/2), (Y, -\pi/2), (X, -\pi/2), (X, \pi/2)\}$ (6 cardinal states $\{\pm X, \pm Y, \pm Z\}$), the following experiment was performed: simultaneously (1) a single LGT program was initiated on the client ($k = 1000$), (2) a single DQC-client program was initiated on the client ($k = 200$ successive subroutines), and (3) a single DQC-server program was initiated at the server ($k = 200$, i.e. 400 successive subroutines). This resulted in a total of 6000 LGT subroutine executions and 36000 LGT measurement results, yielding plotted fidelity estimates for the LGT quantum state before measurement. Results are the same as running LGT on its own (no multitasking with DQC), as expected (Section 5.11.2). **(d)** Scaling number of programs on the client. For $N \in \{1, 2, 3, 4, 5\}$, we initiate at the same time: (1) N LGT programs (each using $k = 100$) on the client, (2) N DQC-client programs on the client (each using $k = 60$), and (3) N DQC-server programs on the server (each using $k = 60$). This results in $2N$ programs active at the same time on the client, each continuously submitting subroutines from the CNPU to the QNPU, where the QNPU scheduler chooses which process to execute when. Each experiment was repeated but with multitasking disabled on the client. Plot shows the utilization factor of the QDevice (fraction of time spent executing instructions), corrected for variable entanglement generation duration (Section 5.6), with (blue) and without (orange) multitasking, showing that multitasking can increase device utilization.

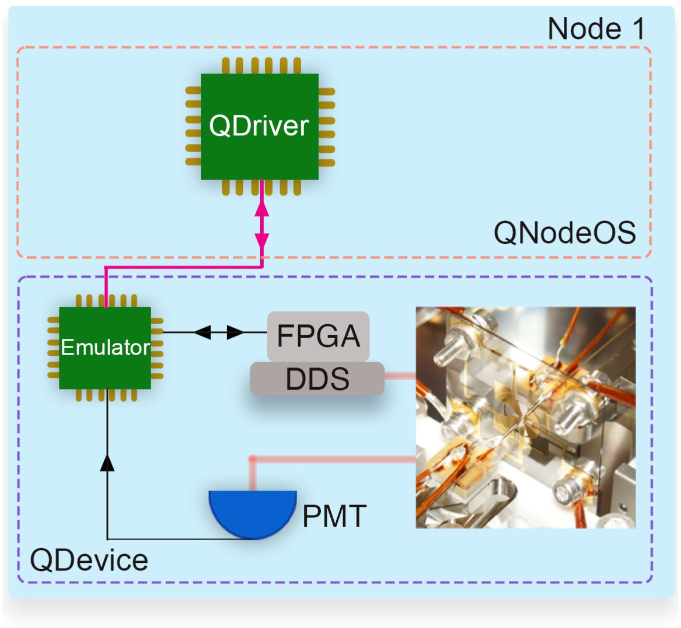


Figure 5.6: **Trapped-ion QDevice implementation.** Schematic of our implementation of QNodeOS on a single-node setup in which the QDevice contains a single trapped-ion qubit. The QNPU QDriver is implemented on a field-programmable gate array (FPGA) that connects to its QDevice via a serial peripheral interface (SPI) (Section 5.6). The setup consists of an emulator that translates between SPI messages and TTL signals, experimental control hardware that includes an FPGA and direct digital synthesis (DDS) modules, a trapped-ion qubit [45] under ultra-high vacuum (Figure 5.2), and a photomultiplier tube (PMT) that registers atomic fluorescence.

Qubits thereby refers to any possible realization of qubits, including logical qubits realized by error-correction. The QDevice exposes the following interface to QNodeOS (Section 5.8.6): number of qubits available, and the supported physical instructions that QNodeOS may send. Physical instructions include qubit initialization, single- and two-qubit gates, measurement, entanglement creation, and a 'no-op' for do nothing. Each instruction has a corresponding response (including entanglement success or failure, or a measurement outcome) that the QDevice sends back to QNodeOS.

QNodeOS and the QDevice interact by passing messages back and forth on clock ticks at a fixed rate (100 kHz in our NV implementation, 50 kHz in the trapped-ion implementation). During each tick, at the same time (1) QNodeOS sends physical instruction to QDevice, (2) QDevice can send a response (for a previous instruction). Upon receiving an instruction, the QDevice performs the appropriate (sequence of) operations (e.g. a particular pulse sequence in the AWG). An instruction may take multiple ticks to complete, where the QDevice returns the response (success, fail, outcome) during the first clock tick following completion. The QDevice handles an entanglement instruction by performing (a batch of) entanglement generation attempts [39] (synchronized by the QDevice with the neighboring node's QDevice).

QNodeOS Architecture. QNodeOS consists of two layers: CNPU and QNPU (Figure 5.3a, Section 5.3, Supplementary). Processes on the QNPU are managed by the Process Manager, and executed by the local processor. Executing a user process means executing NetQASM [43] subroutines (quantum blocks) or that process, which involves running classical instructions (including flow control logic) on the QNPU's local processor, sending entanglement requests to the network stack, and handling local quantum operations by sending physical instructions to the QDriver (Figure 5.3a). Executing the network process means asking the network stack which request (if any) to handle and sending the appropriate (entanglement generation) instructions to the QDevice.

A QNPU process can be in the following states (Figure 5.9 in Supplementary for state diagram): idle, ready, running and waiting. A QNPU process is running when the QNPU processor is assigned to it. The network process becomes ready when a network schedule time-bin starts; it becomes waiting when it finished executing and waits for the next time-bin; it is never idle. A user process is ready when there is at least one NetQASM subroutine pending to be executed; it is idle otherwise; it goes into the waiting state when it requests entanglement from the network stack (using NetQASM entanglement instructions [43]) and is made ready again when the requested entangled qubit(s) are delivered.

The QNPU scheduler oversees all processes (user and network) on the QNPU, and chooses which ready process is assigned to the QNPU processor. CNPU processes can run concurrently, and their execution (order) is handled by the CNPU scheduler. The QNPU scheduler operates independently and only acts on QNPU processes. CNPU processes can only communicate with their corresponding QNPU processes. Since multiple programs can run concurrently on QNodeOS, the QNPU may have multiple user processes that have subroutines waiting to be executed at the same time. This hence requires scheduling on the QNPU.

Processes allocate qubits through the Quantum Memory Management Unit (QMMU), which manages virtual qubit address spaces for each process, and translates virtual addresses to physical addresses in the QDevice. The QMMU can also transfer ownership of qubits between processes, for example from the network process (having just created an entangled qubit), to a user process that requested this entanglement. The Network Stack uses Entanglement Request (ER) sockets (opened by user programs through QNPU API once execution starts) to represent quantum connections with programs on other nodes. The Entanglement Management Unit (EMU) maintains all ER sockets and makes sure that entangled qubits are moved to the correct process.

NV QDevice Implementation. The two-node network employed in this work includes the nodes "Bob" (server) and "Charlie" (client) (separated by 3 meters) described in [7, 22, 39]. For the QDevice, we replicated the setup used by [39], which mainly consists of: an Adwin-Pro II [72] acting as the main orchestrator of the setup; a series of subordinate devices responsible for qubit control, including laser pulse generators, optical readout circuits and an arbitrary waveform generator (Zurich Instruments HDAWG [64]). The quantum physical device, based on NV centers, counts one qubit for each node. The two QDevices share a common 1 MHz clock for high-level communication and their AWGs are synchronized at sub-nanosecond level for entanglement attempts.

We address the challenge of limited memory lifetimes by employing dynamical decoupling (DD). While waiting for further physical instructions to be issued, DD sequences are used to preserve the coherence of the electron spin qubit [73]. DD sequences for NV-centers can prolong the coherence time (T_{coh}) up to hundreds of ms [7] or even seconds [74]. In our specific case, we measured T_{coh}=13(2) ms for the server node, corresponding to 1300 DD pulses. The discrepancy to the state-of-the-art for similar setups is due to several factors. To achieve such long T_{coh}, a thorough investigation of the nuclear spin environment is necessary to avoid unwanted interactions during long DD sequences, resulting in an even more accurate choice of interpulse delay. Other noise sources include unwanted laser fields, the quality of microwave pulses and electrical noise along the microwave line.

A specific challenge arises at the intersection of extending memory lifetimes using DD, and the need for interactivity: to realize individual physical instructions, many waveforms are uploaded to the Arbitrary Waveform Generator (AWG), where the QDevice decodes instructions sent by QNodeOS into specific preloaded pulse sequences. This results in a waveform table, containing 170 entries. The efficiency of the waveforms is limited by the AWG's waveform granularity that corresponds to steps that are multiples of 6.66 ns, having a direct impact on the T_{coh}. We are able to partially overcome this limitation through the methods described in [75]. Namely, each preloaded waveform, corresponding to one single instruction, has to be uploaded 16 times in order to be executed with sample precision. To not fill up the waveform memory of the device, we apply the methods in [75] only to the DD pulses that are played while the QDevice waits for an instruction from the QNPU, whereas the instructed waveforms (gate/operation + first block of XY8 DD sequence) are padded according to the granularity, if necessary. The physical instructions supported by our NV QDevice is given in Section 5.9.1.

NV QNPU Implementation. The QNPUs for both nodes are implemented in C++ on top of FreeRTOS [76], a real-time operating system for microcontrollers. The stack runs on a dedicated MicroZed [77]—an off-the-shelf platform based on the Zynq-7000 SoC, which hosts two ARM Cortex-A9 processing cores, of which only one is used, clocked at 667 MHz. The QNPU was implemented on top of FreeRTOS to avoid re-implementing standard OS primitives like threads and network communication. FreeRTOS provides basic OS abstractions like tasks, inter-task message passing, and the TCP/IP stack. The FreeRTOS kernel—like any other standard OS—cannot however directly manage the quantum resources (qubits, entanglement requests and entangled pairs), and hence its task scheduler cannot take decisions based on such resources. The QNPU scheduler adds these capabilities (Section 5.8.5).

The QNPU connects to peer QNPU devices via TCP/IP over a Gigabit Ethernet interface (IEEE 802.3 over full-duplex Cat 5e). The communication goes via two network switches (Netgear JGS524PE, one per node). The two QNPUs are time-synchronized through their respective QDevices (granularity 10 μs), since these already are synchronized at the μs-level (common 1Mhz clock).

The QNPU device interfaces with the QDevice's ADwin-Pro II through a 12.5 MHz SPI interface, used to exchange 4-byte control messages at a rate of 100 kHz.

NV CNPU Implementation. The CNPUs for both nodes are a Python runtime executing on a general-purpose desktop machine (4 Intel 3.20 GHz cores, 32 GB RAM, Ubuntu 18.04). The choice of using a high-level system was made as the communication between distant nodes would ultimately be in the ms-timescales, and this allows for ease of programming the application. The CNPU machine connects to the QNPU device via TCP over a Gigabit Ethernet interface (IEEE 802.3 over full-duplex Cat 8, average ping RTT of 0.1 ms), via the same single network switch as mentioned above (one per node), and sends application registration requests and NetQASM subroutines over this interface (10 to 1000 bytes, depending on the length of the subroutine). CNPUs communicate with each other through the same two network switches.

Scheduler Implementation. We use a single Linux process (Python) for executing programs on the CNPU. CNPU 'processes' are realized as threads created within this single Python process. Python was chosen since the NetQASM SDK is implemented in Python. When running multiple programs concurrently, a pool of such threads is used. Scheduling of the Python process and its threads is handled by the Linux OS. Each thread establishes a TCP connection with the QNPU in order to use the QNPU API (including sending subroutines and receiving their results) and executes the classical blocks for its corresponding program. Both the CNPU and QNPU maintain processes for running programs. The CNPU scheduler (standard Linux scheduler, see above) schedules CNPU processes, which indirectly controls in which order subroutines from different programs arrive at the QNPU. The QNPU scheduler handles subroutines of the same process priority on a first-come-first-served (FCFS) basis, leading however to executions of QNPU processes not in the order submitted by the CNPU (Section 5.11.3).

Using only the CNPU scheduler is not sufficient since (1) we want to avoid millisecond delays needed to communicate scheduling instructions across CNPU and QNPU, (2) user processes need to be scheduled in conjunction with the network process (meeting the challenge of scheduling both local and network operations), which is only running on the QNPU, and (3) QNPU user processes need to be scheduled with respect to each other, (e.g. a user process is waiting after having requested entanglement, allowing another user process to be run; as observed in the multitasking demonstration).

Sockets and the Network Schedule. In an ER Socket, one node is a 'creator' and the other a 'receiver'. As long as an ER socket is open between the nodes, an entanglement request from only the creator suffices for the network stack to handle it in the next corresponding time-bin, i.e. the 'receiver' can comply with entanglement generation even if no request has (yet) been made to its network stack.

Trapped-ion Implementation. The experimental system used for the trapped-ion implementation is discussed in [45, 63] and is described in detail in [78]. The implementation itself is described in [16]. We confine a single $^{40}\mathrm{Ca}^{+}$ ion in a linear Paul trap; the trap is based on a 300 µm thick diamond wafer on which gold electrodes have been sputtered. The ion trap is integrated with an optical microcavity composed of two fiber-based mirrors, but the microcavity is not used here. The physical-layer control infrastructure consists of C++ software; Python scripts; a pulse sequencer that translates Python commands to a hardware description language for a field-programmable gate array (FPGA); and hardware that includes the FPGA, input triggers, direct digital synthesis (DDS) modules, and output logic.

QNodeOS provides physical instructions through a development FPGA board (Texas Instruments, LAUNCHXL2-RM57L75) that uses a serial peripheral interface (SPI). We programmed an additional board (Cypress, CY8CKIT-14376) that translates SPI messages into TTL signals compatible with the input triggers of our experimental hardware. The implementation consisted of sequences composed of seven physical instructions: initialization, $R_x(\pi)$, $R_y(\pi)$, $R_x(\pi/2)$, $R_y(\pi/2)$, $R_y(-\pi/2)$, and measurement. First, we confirmed that message exchange occurred at the rate of 50 kHz as designed. Next, we confirmed that we could trigger the physical-layer hardware. Finally, we implemented seven different sequences. Each sequence was repeated 10^4 times, which allowed us to acquire sufficient statistics to confirm that our QDriver results are consistent with operation in the absence of the higher layers of QNodeOS.

Metrics. Both classical and quantum metrics are relevant in the performance evaluation: The quantum performance of our test programs is measured by the fidelity $F(\rho,|\tau\rangle)$ of an experimentally obtained quantum state ρ to a target state $|\tau\rangle$ where $F(\rho,|\tau\rangle) = \langle\tau|\rho|\tau\rangle$, estimated by quantum tomography [79]. Classical performance metrics include device utilization $T_{\text{util}} = 1 - T_{\text{idle}}/T_{\text{total}}$ where T_{idle} is the total time that the QDevice is not executing any physical instruction, and T_{total} is the duration of the whole experiment excluding time spent on entanglement attempts (see below).

Experiment Procedure NV Demonstration. Applications are written in Python using the NetQASM SDK [43], with a compiler targeting the NV flavour [43], as it includes quantum instructions that can be easily mapped to the physical instructions supported by the NV QDevice. The client and server nodes independently start execution of their programs by invoking a Python script on their own CNPU, which then spawns the threads for each program. During application execution, the CNPUs have background processes running, including QDevice monitoring software.

A fixed network schedule is installed in the two QNPUs, with consecutive time-bins (all assigned to the client-server node pair) with a length of 10 ms (chosen to be equal to 1000 communication cycles between QNodeOS and QDevice as in Ref. [39]) to assess the performance without introducing a dependence on a changing network schedule. During execution, the CNPUs and QNPUs record events including their timestamps. After execution, corrections are applied to the results (see below) and event traces are used to compute latencies.

Delegated Quantum Computation. Our demonstration of DQC (Figure 5.4) implements the effective single-qubit computation $|\psi\rangle = H \circ R_z(\alpha) \circ |+\rangle$ on the server, as a simple form of blind quantum computing (BQC) that hides the rotation angle α from the server, when executed with randomly chosen θ, and not performing tomography. The remote entanglement protocol utilized is the single-photon protocol [80–82] (Section 5.9.1).

Filtering. Results, with no post-selection, are presented including known errors that occur during the tomography single-shot readout (SSRO) process (Figure 5.4b, blue) (details on the correction Supplementary of [22]). We also report the post-selected results in which data are filtered based on the outcome of the Charge-Resonance check [83] after one application iteration (Figure 5.4b, purple). This filter enables the elimination of false events, specifically when the emitter of one of the two nodes is not in the right charge state (ionization) or the optical resonances are not correctly addressed by the laser fields after the execution of one iteration of DQC.

Additional filtering (Figure 5.4b latency filter) is done on those iterations that showed latency not compatible with the combination of T_{coh} of the server and the average entangled state fidelity. For this filter, a simulation (using a depolarizing model, based on the measured value T_{coh}, Section 5.10.4) was used to estimate the single qubit fidelity (given the entanglement fidelity measured above) as a function of the duration the server qubit stays live in memory in a single execution of the DQC circuit (Figure 5.4a). This gives a conservative upper bound of the duration as 8.95 ms, to obtain a fidelity of at least 0.667. All measurement results corresponding to circuit executions exceeding 8.95 ms duration were discarded (146 out of 7200 data points).

Other main sources of infidelity, that are not considered in this analysis of the outcome, include, for instance, the non-zero probability of double excitation for the NV center [82]. During entanglement generation, the NV center can be re-excited, leading to the emission of two photons that lower the heralded entanglement fidelity. The error can be corrected by discarding those events that registered, in the entanglement time-window, a photon at the heralding station (resonant Zero-Phonon Line photon) and another one locally at the node (off-resonant Phonon-Side Band photon).

Finally, the dataset presented in Figure 5.4b (not shown chronologically) was taken in "one shot" to prove the robustness of the physical layer, therefore no calibration of relevant experimental parameters was performed in between, leading to possible degradation of the overall performance of the NV-based setup.

The single qubit fidelity is calculated with the same methods as in [8], measuring in the state $|i\rangle$ and in its orthogonal state $|-i\rangle$, provided that we expect the outcome $|i\rangle$, whereas the two-qubit state fidelity is computed taking into account only the same positive-basis correlators (XX, YY, ZZ).

Multitasking: Delegated Computation and Local Gate Tomography. In the first multitasking evaluation, we concurrently execute two programs on the client: a DQC-client program (interacting with a DQC-server program on the server) and a Local Gate Tomography (LGT) program (on the client only) (Figure 5.5). The client CNPU runtime executes the threads executing the two different programs concurrently. The client QNPU has two active user processes, each continuously receiving new subroutines from the CNPU, which are scheduled with respect to each other and the network process.

Estimates of the fidelity (Figure 5.5b) include same corrections as in the Supplementary of [22] To assess the quantum performance of the LGT application, we used a mocked entanglement generation process on the QDevices (executing entanglement actions without entanglement) to simplify the test: weak-coherent pulses on resonance with the NV transitions, that follow the regular optical path, are employed to trigger the CPLD in the entanglement heralding time-window. This results in comparable application behavior for DQC (comparable rates and latencies, Section 5.11.1) with respect to multitasking on QNodeOS.

Multitasking: QDevice Utilization when scaling number of programs. We scale the number of programs being multitasked (Figure 5.5d): We observe how the client QNPU scheduler chooses the execution order of the subroutines submitted by the CNPU. DQC subroutines each have an entanglement instruction, causing the corresponding user process to go into the waiting state when executed (waiting for entanglement from the network process). The QNPU scheduler schedules another process [(56%, 81%, 99%) for (N=1, N=2, N>2)] of the times that a DQC process is put into the waiting state (demonstrating that the QNPU schedules independently from the order in which the CNPU submits subroutines). The number of consecutive LGT subroutines (of any LGT process; LGT block execution time 2.4 ms) that is executed in between DQC subroutines is 0.83 for N=1, increasing for each higher N until 1.65 for N = 5, showing that indeed idle times during DQC are partially filled by LGT blocks (Section 5.11.3).

Device utilization (see Metrics above) quantifies only the utilization factor in between entanglement generation time windows to fairly compare the multitasking and the non-multitasking scenario. In both scenarios, the same entanglement generation processes are performed, which hence have the same probabilistic durations in both cases. To avoid inaccurate results due to this probabilistic nature, we exclude the entanglement generation time windows in both cases.

5.7. BACKGROUND AND FURTHER DETAILS ON DESIGN CONSIDERATIONS

This section provides background information to bridge the gap between physics and computer science for convenience of the reader. A reader, who is an expert in quantum networking in both disciplines may wish to skip parts of this Section. Having provided additional background information then helps us elucidate further on the design considerations and challenges in designing an operating system for execution applications on quantum network nodes.

5.7.1. QUANTUM NETWORKS

We refer the reader with a background in computer science to [3] for a gentle introduction to quantum networks.

Node Types. Within a quantum network, one can distinguish between two main types of nodes: First, there are *end nodes* [2], on which users execute quantum network applications. Classically, such end nodes are laptops, phones or other devices connected to the network. In the quantum domain, end nodes may be simple photonic devices that can only create or measure quantum states, or they may be quantum processors capable of arbitrary qubit operations and storage of information within a quantum memory. The type of end node dictates what applications are possible [2].

Here, we have chosen to focus on the most general form of an end node, namely, a quantum processor. More precisely, our goal is to enable programming and execution of arbitrary quantum network applications on end nodes that are quantum processors. For the remainder of this text, we will thus always take end nodes to be quantum-processor end nodes.

Second, a quantum network can include *intermediate nodes*, such as quantum repeaters, that perform routines necessary to establish long-distance entanglement between two or more end nodes (Figure 5.7). These intermediate nodes may employ protocols such as entanglement swapping and entanglement distillation in order to realize end-to-end links (i.e. entanglement) with sufficiently high fidelity for network applications. These protocols are handled by a network stack (see, e.g. [35]) that exists at each node. The network stack includes a link layer, a network layer, a control plane, and other networking functions; it is responsible for entanglement generation.

Intermediate nodes do not execute user applications, which is done only by end nodes. Therefore, only end nodes need to have an additional stack implementing the application layer in a network, which is referred to as an *application stack* (see Figure 5.7). The application stack is responsible for the execution of arbitrary user applications, and integrates with the network stack for entanglement generation over the network. We remark that it is the purpose of a network layer in the network stack [35, 36] to provide a service to the application layer that allows entanglement generation with remote end nodes. Importantly, this service should not require the application layer to have any knowledge about the connectivity of the network.

While QNodeOS can in principle also be run on intermediate nodes as it already implements a network stack, we remark that it is designed primarily to enable the execution of applications on end nodes, which is the focus of this paper.

End Node Quantum Processors. A quantum-processor end node includes a quantum memory in the form of qubits, and allows gates and measurements to be executed on such qubits (see the Methods section in the main text, 'QDevice Model'), which can be used to realize user applications. These QDevice qubits may be physical qubits, but may also be logical qubits (multiple physical qubits together representing one more robust qubit) in case the end node employs quantum error correction [47]. For our architecture it does not matter how qubits are realized in the QDevice (which may internally employ quantum error correction), as long as the QDevice follows our system model.

Entanglement. Entanglement is a phenomenon in quantum physics where two or more qubits are correlated in a way that is not possible for classical bits. In a quantum network, establishing entanglement means we create entanglement between qubits at different nodes. Entanglement can be used by quantum network applications as a resource in order to realize applications [2] that are impossible with classical networks, including applications such as data consistency in the cloud [27], privacy-enhancing proofs of deletion [28], exponential savings in communication [29], or secure quantum computing in the cloud [13, 30]. While our architecture could readily be extended to networks that inherently produce multi-partite entanglement (i.e. entanglement between several qubits at once), we focus on the case of bipartite entanglement (i.e. entanglement between two qubits) as the basic unit of entanglement.

5.7.2. Application Paradigm

Our architecture is primarily meant to *enable the execution of quantum network applications in the quantum memory stage [2] and above*. That is, applications that require the use of a quantum processor that can manipulate and store quantum bits (qubits). For simpler applications in the prepare-and-measure and entanglement generation stages [2], e.g. quantum key distribution [85, 86], where the quantum states are immediately measured by the nodes, our system can also be used, but it would be sufficient to realize a system implementing a quantum network stack (including a measure-directly functionality [35]) and classical processing only.

Separated Programs. Recall that a quantum networking application consists of multiple programs, each running on one of the end nodes. For ease of explanation we will assume we are executing an application between two nodes, e.g. a client and a server, although our system can also be used to execute applications among many end nodes. Each node in the network runs its own independent *Quantum Network Operating System (QNodeOS)*, on which the node's program is executed. The two programs may interact with each other via message passing and entanglement generation, where both types of interactions are managed by the node's QNodeOS. Next to interaction via the programs, the nodes may exchange additional classical messages which are not part of the program itself, for example, in order to enable the realization of a network stack [35] managing entanglement generation between the nodes.

Classical blocks of code consist of instructions for local classical operations and classical message passing. Quantum *blocks of code* consists of (1) quantum operations (initialization, quantum gates, measurement), (2) low-level classical control logic (branching on classical variables and loops), as well as (3) instructions to make entanglement between remote nodes. We remark that classical and quantum instructions may require many actions by QNodeOS (and quantum system controlled by it) in order to be fulfilled: it is the goal of such instructions to abstract away aspects of the underlying system.

Classical blocks of code may depend on quantum ones via classical variables generated during the quantum execution (such as measurement results, notification of entanglement generation, and information on the state of the quantum system such as the availability of qubits). Similarly, quantum blocks may depend on variables set by the classical blocks (such as messages received from remote network nodes). Finally, quantum blocks may themselves depend on other quantum blocks via qubits in the quantum memory.

Performance Metrics. Next to classical metrics, such as utility (see the Methods section in the main text, 'Metrics'), throughput or latency [87], the successful execution of quantum network applications is governed by quantum performance metrics, which are unique to quantum networks and not present in classical networks. Such quantum network-specific metrics include fidelity (see the Methods section in the main text, 'Metrics'), or the probability of success in executing an application, where the latter depends directly on the fidelity of the quantum states prepared.

Mode of Execution. There exist quantum applications and functionalities, where one pair of programs is executed only once. A simple example of such a functionality is quantum teleportation [88]. As in quantum computing, however, some quantum network applications [2] are expected to succeed only with a specific *probability of success* p_{succ} when executed once. The application is then typically executed many times in succession in order to gather statistics (for example to amplify p_{succ}). A common use case for executing the same application repeatedly also occurs when evaluating the performance of a system (as we do here), where the goal is to estimate quantum performance metrics, such as the probability of success or the fidelity (see the Methods section in the main text, 'Metrics'). When executing the same application multiple times, the programmer can choose to launch many instances of the same program at once if multitasking is possible (see below), or to write one wrapper program which repeatedly executes the original program in a loop, asking for a successive execution of the application.

5.7.3. Interactive Classical-Quantum Execution

Let us elaborate further on the relation, and differences, between the execution of quantum network applications, and the execution of quantum computing applications: One could envision building a system for executing quantum network applications on top of a simpler system for the execution of quantum computing programs, as long as the latter can be augmented with networking instructions to generate entanglement: in essence one quantum block can be seen as one quantum computing program. Such a block may realize mid-circuit measurements by the classical control logic allowed within one quantum block, or error correction. Error correction could in this paradigm be realized both by classical control logic allowed within one quantum block, or by considering the error correction itself as part of the Quantum Device (QDevice) (see Section 5.8.1) which then only exposes logical qubits and operations to QNodeOS, instead of physical qubits and operations.

In that sense, one may think of the interactivity required between classical and quantum operations in quantum network applications as taking place not only at a higher level, but also stemming from the fact that classical messages are used to create a new interaction between *separate* quantum programs, while in quantum computing we have only one single program.

5.7.4. Different Hardware Platforms

Platform Independence

We provide further background on the concept of platform, i.e. hardware, independence. It is the goal of our architecture to be platform-independent, including a standard interface to a driver for different hardware platforms. The driver is thus the only part that is platform-dependent in order to steer the underlying hardware platform. Such an interface is known as a Hardware Abstraction Layer (HAL) that allows interfacing with different (quantum) platforms. To restate, in the context of "classical" operating systems, a HAL is a core component that existed in many operating systems (like Windows 7 [89, Section 19.3.1]) and continues to be used extensively to this day in operating systems for a broad set of computing platforms, including mobile ones [90]. A HAL allows the operating system kernel to interact with the device hardware (drivers) through standardized programming interfaces, instead of relying on interfaces written specifically for each available hardware. Therefore, a HAL allows for an ultimate portability of the operating system, making it platform-independent above the HAL, and simplifies the architecture of the operating system.

QDevice

We consider as a quantum processor system the QDevice model (see the Methods section in the main text, 'QDevice Model'), exposing a set of physical instructions addressing specific qubits (see Section 5.8.6). These physical instructions may be dependent on the type of quantum hardware, e.g., Nitrogen Vacancy (NV) in diamond, or trapped ions, and (1) include instructions for initializing and measuring qubits on the chip, (2) moving the state of a qubit to another location in the quantum memory (3) performing quantum gates, as well as (4) to make attempts at entanglement generation at the physical layer [39].

Quantum processors in general offer two types of qubits (see e.g. [35]): *communication qubits* which can be used to generate entanglement with remote nodes next to performing other quantum operations, as well as *storage qubits* which cannot be used to generate entanglement and only allow local quantum operations. We remark that on near-term quantum processors, the types of operations that can be performed on specific sets of qubits also depends on the connectivity of the qubits. That is, not all (pairs of) qubits may allow the same set of quantum operations to be performed on them.

To later enable compile time optimization, it is desirable that the QDevice furthermore exposes the capabilities of the quantum chip: (1) the number of qubits, (2) the type of each qubit, (3) the memory lifetime of the qubits, (4) the physical instructions that can be performed on on the qubit(s) and (5) the average quality of these instructions.

5.7.5. TIMESCALES

Quantum network programs are meant to be executed between distant nodes, meaning the communication times between them are in the *millisecond* regime. We remark that the same is *not true for networked or distributed quantum computing*: if the goal is to combine several less powerful quantum processors via a network into one more powerful quantum computing cluster, then it is advisable to place the individual processors as close to each other as possible, in order to minimize the time needed to (1) exchange messages, and (2) generate entanglement between processors. Thus, apart from the execution of applications following a different paradigm (see Figure 1 in the main text), the case of distributed quantum computing also has different timescales than quantum networking. Of course, it is conceivable that in the future, one may also link distant quantum computers into more powerful quantum computing clusters via quantum internet infrastructure.

5.7.6. SCHEDULING NETWORK OPERATIONS

In order for two neighboring quantum network nodes to produce heralded entanglement between them, they need to simultaneously perform an action to trigger entanglement generation (at the physical layer, synchronized to nanosecond precision). This means neighboring quantum network nodes need to perform a network operation (entanglement generation) in a *very specific* time slot in which they make an attempt to generate entanglement. Such time slots are generally aggregated into larger time bins, corresponding to making batches of attempts in time slots synchronized at the physical layer. We refer to e.g. Ref. [39] for background information on the physical layer of entanglement generation in quantum networks, and the readers with a background in computer science to e.g. Ref. [35] for a detailed explanation.

In short, network operations in quantum networks need to be executed by the node at very specific time bins. These time bins cannot be determined by the quantum node on its own. Instead selection of time bins for a specific quantum operation require agreement with the neighboring node [35] (and more generally with the quantum network when the end-to-end entanglement is made via intermediary network nodes) by means of a network schedule, e.g. determined by a (logically) centralized controller, see Ref. [55].

5.7.7. Scheduling Local Operations versus Scheduling Network Operations

For computer scientists, we provide further information on the inability to simultaneously execute both local as well as networked quantum operations on present-day quantum processors. At a high-level, present-day quantum processor can be seen as both a quantum Central Processing Unit (CPU)/memory, as well as network device. Physical properties of the device and its control at the level of experimental physics, prohibits the usage of the quantum processors for both network and CPU/memory functions at the same time. A good example is given by the system of NV centers in diamond [54, 91]: the communication qubit, i.e. the network device, of the NV quantum processor system is given by its electron spin. Further storage qubits may be available by the surrounding nuclear spins in the diamond material. However, such nuclear spins cannot easily be addressed without involving the electron spin, prohibiting their use as a separate processor that is independent from the use as a network device.

It is conceivable that in the future, two devices could be used [58]: one *quantum device as a network device* for entanglement generation with other network nodes, and a separate *quantum processor as a quantum CPU, performing only local quantum operations*. The network device could produce entanglement with distant quantum nodes (which may be taking many *milliseconds* to conclude successfully), and only once such entanglement is ready inject it into the quantum CPU. Such an injection may still involve short-distance entanglement generation between the network device and the quantum CPU, which however is very fast at short distances. This way the time that the quantum CPU would be blocked by networking operations would diminish significantly.

5.7.8. Multitasking

When executing quantum network applications, multitasking is well motivated in order to increase the utility of the system. Multitasking (or time sharing) is a well-established concept in classical operating systems (see e.g. [89, Section 1.4]) that allows the concurrent execution of multiple programs. For the reader from physics, we summarize some of these concepts in order to give context, and then reflect on what these imply in our setting.

In order to allow for multitasking, operating systems typically employ a notion of processes (or threads [89, Chapter 4], or tasks [89, Section 3.1]), where a process is created whenever a program starts, and the process forms an instance of the program being executed on the system. Multitasking (time sharing) thus refers to the concurrent execution of multiple processes at once, where it is possible to have multiple processes for the same program, corresponding to the execution of several instances of the program in parallel. We remark that the term concurrent thereby refers to the fact that the processes are existing in the system at the same time, while—due to the fact that they need to share limited resources (e.g. a CPU or other devices)—not all of them may be running at the same time.

Allowing multitasking requires the system to include a number of additional features.

Managing Processes

At a high-level, multitasking requires the system to keep track of the currently running processes, which means that when program starts executing, a process must be registered in the system. Since the system needs to decide which process can be executed at what time, i.e., which process can be given access to the necessary resources to allow its execution, the system needs to keep track of the state of the process, which typically includes (1) whether it is ready for execution, (2) currently running, or (3) whether it cannot currently be executed since it is waiting e.g. for other processes.

In the case of executing quantum network applications, different parts of the application require different resources in order to run: classical blocks need the classical processor (CPU) and potentially the network device present in a Classical Network Processing Unit (CNPU), while quantum blocks require the quantum processor (QDevice). It is desirable for our system that both resources can be used concurrently. That is, two different processes should be able to execute a classical block (on the CNPU), and a quantum block (on the Quantum Network Processing Unit (QNPU)) at the same time.

Memory Management Unit

A program typically relies on the ability to store classical variables, as well as quantum variables. Such variables are stored in a classical and quantum memory device (here, the quantum processor), respectively. In order to allow multiple concurrent processes at the same time, the system needs to keep track of which part of the classical and quantum memory is assigned to which process. This concept is known broadly as memory management [89, Section 1.7] in classical operating systems.

In order to allow multitasking of quantum network applications, we thus require a Quantum Memory Management Unit (QMMU) (next to standard ways of performing classical memory management). The QMMU is responsible for the following tasks:

Qubit information handling. A QMMU has knowledge of the qubits available on the underlying QDevice, and may keep any other information about said qubits, such as the qubit type (communication or storage qubit) and qubit lifetime. Qubits, or QDevice qubits, thereby refer to both qubits realized at the device level, e.g. in the electron spin states of the NV center in diamond, or at a logical level where quantum error correction [47] is used to protect the quantum memory, i.e. one logical qubit is created by performing error-correcting using many device level qubits. A QMMU should allow such qubits to be assigned to different owners, i.e. different processes, or the operating system itself.

Transfer of qubit ownership. The QMMU may also allow a transfer of ownership of the qubits from one owner to the other, such as for example from a network process to a user process.

Quantum memory virtualization. A QMMU may also provide abstractions familiar from classical computing such as a virtual address space, where the applications refer to virtual qubit addresses that are then translated to QDevice qubit addresses. This virtual address space avoids the situation in which QDevice qubit addresses must be bound at compile time, particularly limiting when allowing multiple applications to concurrently run on the same node. This would allow the transparent moving of qubits in a quantum memory in the future (for example moving them from a processor to a memory-only device while the process is waiting, e.g., for a message from a remote node). We remark however that the noise in present-day quantum devices means that any such move introduces a significant amount of additional noise to the quantum state that may prevent the successful execution of the application.

Qubit memory lifetime management. Advanced forms of a QMMU may also cater to the limitations of near term quantum devices, by matching memory lifetime requirements specified by the application code to the capabilities of the underlying qubits, as well their topology, i.e., taking into account which two qubits allow two-qubit gates to be performed on them directly. While one cannot measure the decoherence of a qubit during program execution on the quantum level, the QMMU could also take into account additional information from the classical control system to signal to the application that a qubit has become invalid.

PROCESS SCHEDULER

When multitasking, we need to decide which process should be executed at what time. This concept is referred to as scheduling in classical operating systems [62, Section 2.4], [89, Section 3.2]. We first discuss design considerations for scheduling when executing quantum network applications, and then reflect on how scheduling may be realized at different levels of an operating system for quantum network nodes.

General considerations. We first provide three general considerations for completeness, which are not specific to the execution of quantum network applications but apply to all systems in which several resources (such as the QDevice and a classical CPU) can be used (largely) independently of each other:

1. *Local quantum computation*: in addition to quantum networking, a node's resources must also be reserved for local quantum gates, which are integral parts of quantum network applications.

2. *Multitasking*: for a node to be shared by multiple users, the scheduler should not allocate all the available resources to a single application indefinitely, and instead it should be aware of the presence of multiple applications.

3. *Inter-block dependencies*: quantum and classical processing blocks of an application may depend on results originating from other blocks, and thus cannot be scheduled independently.

Quantum network considerations. Two specific considerations stand out in the domain of quantum networking:

1. *Synchronized network schedule*: due to the bilateral nature of entanglement, each node needs to have its quantum networking activity synchronized with its immediate neighbors. This means that while the process schedulers at different QNodeOS nodes run independently of each other, nodes must take into account the network schedule which defines when the node needs to perform networking actions with its neighboring node.

2. *Limited memory lifetimes*: the performance of quantum networking applications depends on both classical as well as quantum metrics. Once qubits are initialized, or entanglement has been created, the limited lifetime of present-day quantum memories implies that execution must be completed by a specific time in order to achieve a desired level of quantum performance.

Quantum/classical performance metrics trade-off. The best quantum performance is reached when the entire quantum network system (all nodes) are reserved for the execution of one single quantum network application. That is, programs are executed in a serial fashion and no multitasking is performed that could introduce delays. Due to limited quantum memory lifetimes, delays can negatively impact the quantum performance. However, this approach does not in general achieve the best utilization of the system.

While our implementation makes use of a simple priority based scheduler, we remark that our work opens the door to apply more advanced forms of schedulers in the future. In particular, the fact that execution quality degrades over time suggests using forms of real-time schedulers for quantum network applications (taking inspiration from the extensive work on this topic in classical systems, see e.g. Ref. [65]). We remark that a programmer (or compiler) could provide advise on such (soft) deadlines, for example in the form of a lookup table that includes suggestions for deadlines for a desired level of quantum performance, based on the capabilities provided by the underlying hardware systems (e.g. memory lifetimes, expected execution time of quantum blocks), and the network (e.g. rate and quality (fidelity) of the entanglement that can be delivered). This advise could then be used by the scheduler to inform its scheduling decisions.

We remark that determining precise deadlines (e.g. when too much time has elapsed for the qubits to yield a specific probability of success) is in general a computationally expensive procedure, sometimes estimated in practice by a repeated simulation of the execution. It is an interesting open question to find (heuristic) efficient methods to approximate a performance prediction. We remark that there is no way in quantum mechanics to measure the current quality of a qubit or operation during the ongoing execution, and such qualities are determined by performing estimates independently of the program execution itself. Of course, QNodeOS could itself engage in such estimates when idling in order to update its knowledge of the capabilities of the quantum hardware.

To allow for potentially time-consuming classical pre- and post-processing, it is natural to apply such deadlines not for the entirety of the application, but for the period between initializing the qubits and terminating the quantum part of the execution. While outside the scope of this work, we remark that this type of scheduling offers to inspire new work in a form of "quantum soft-real time" scheduling, where deadlines may occasionally be missed at the expense of reduced application performance (success probability), to maximize the overall (averaged) performance of the system in which applications are typically executed repeatedly.

Scheduling at different system levels. Above we discussed scheduling at the level of processes, corresponding to executions of program instances. A system may realize scheduling at different levels, including:

1. *Classical versus quantum processes*: The system may sub-divide processes into classical processes (executing classical code blocks), and quantum user processes (executing quantum code blocks). In this case, these can be scheduled independently (provided inter-dependencies are taken into account).

2. *Scheduling of quantum blocks*: The system may further sub-divide quantum processes into smaller units to allow different quantum code blocks of the same process to be scheduled independently.

3. *Scheduling of individual operations*: The level of operating systems is not typically concerned with the scheduling of individual operations, which is instead taken care of by the underlying CPU. We remark that while we do not envision this type of scheduling to be part of such a system in the future, but rather be relegated to control hardware in a microarchitecture for quantum nodes as e.g. in Fu et al. [92], our current realization of QNodeOS achieves a simple form of instruction schedule by populating an instruction queue in software due to the absence of a suitable low-level microarchitecture.

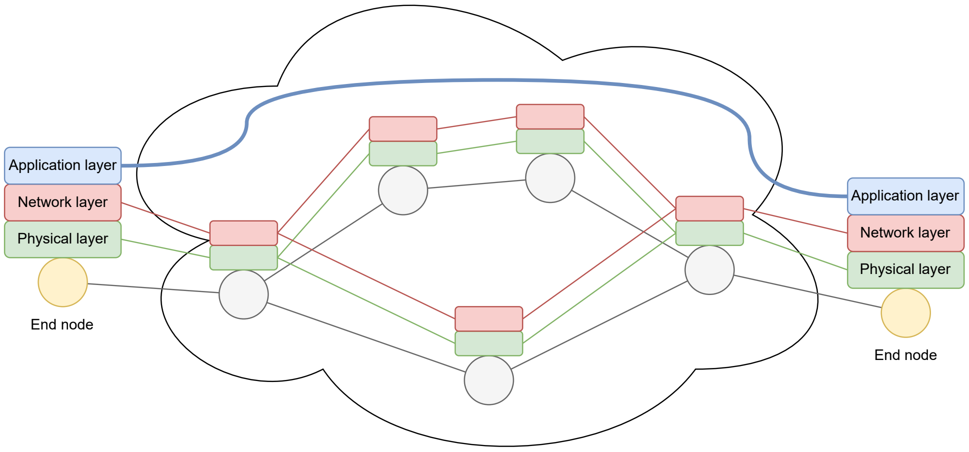


Figure 5.7: **High-level overview of a quantum network.** A quantum network consists of nodes (yellow and grey circles) that are connected by classical and quantum communication channels (grey lines). Each node implements a physical layer (green boxes and lines) that enables entanglement generation with neighboring nodes. The physical layer is the domain of the QDevice. Each node also implements a network stack, including a network layer (red boxes and lines, which may be subdivided into a separate link layer and a network layer [35, 84]). This layer realizes long-distance entanglement creation between nodes and may make use of protocols such as entanglement swapping and distillation. As QNodeOS implements a network stack, it could also be deployed on intermediate nodes (gray circles) in a network, where e.g. entanglement distillation could be added to the protocol realizing the network layer service implemented by QNodeOS.
We emphasize that the focus of this work is to program and execute applications on the end nodes, i.e. an application stack enabling the application layer in networking terms. Only *end nodes* (yellow circles) implement an additional application layer (blue boxes and line), which executes arbitrary user applications. From the perspective of this layer, end nodes are logically directly connected (blue line), and this layer is hence independent from implementations and protocols in the network layer and only depends on the service provided by the network layer. Logically directly connected means that the application layer relies on the service of the network layer to enable end-to-end entanglement generation between end nodes and does not concern itself with how the entanglement is generated. This abstraction is a key element enabled by a quantum network stack such as [35] and exactly analogous to abstractions used in classical networking, where e.g. a web browser can be executed on a laptop independently of how the internet connection between the laptop and a web server is realized. In the same way, QNodeOS could operate on end nodes separated by a large quantum network of the future, in which many intermediary nodes may lie on the path connecting the end nodes.

5.8. QNodeOS Design and Implementation

We proceed with a more detailed description of the QNodeOS architecture and implementation, where (for the reader's convenience) we include some information already found in the main part of the paper. Recall, that a quantum network application is realized by running separate programs, one on each end node of the quantum network that takes part in the quantum application. The individual programs interact with each other only via classical message passing and entanglement generation. Each program itself consists of classical and quantum blocks of code (see Section 5.7.2) which require execution on the quantum processor for the application to succeed.

5.8.1. QNodeOS Architecture

Quantum Network Node System

We remark that QNodeOS—a real-time system for quantum network nodes—is designed to be deployed on end and intermediary nodes (Section 5.7.1), where QNodeOS use on intermediary nodes can be restricted to facilitate entanglement generation over the network via a (series) of intermediate nodes. As the focus of this paper is the execution of quantum network applications, we focus here on running QNodeOS on end nodes.

In our model, as depicted in paperFigure 5.3a Figure 2a in the main text, we divide the functions of a node into three high-level components:

- a *CNPU*, on which classical blocks of code are executed. The *CNPU* requires classical computing hardware (including a classical CPU), as well as a classical network device to allow the exchange of messages with the CNPU of remote nodes. While quantum networking programs can in principle be developed and compiled outside of the CNPU, the *CNPU* may also realize a user environment where quantum networking programs (Section 5.8.1) are developed and compiled, and where program results are stored;
- a *QNPU*, which receives quantum blocks from the CNPU and entanglement generation requests from peer nodes, and manages execution on the quantum physical device;
- a *QDevice*—the quantum physical device—consisting of a quantum processor, a quantum network device, and a quantum memory, where actual quantum computations and communications take place. In present-day quantum hardware implementations, the same device acts as a quantum processor, a network device and a memory.

In summary, in our design a quantum network program starts on the CNPU, which runs classical code blocks internally, and offloads quantum code blocks to the QNPU. The QNPU runs the quantum code blocks, relying on the underlying quantum device, i.e., QDevice, to execute the actual quantum operations.

CNPU and QNPU—while both being capable of performing non-quantum operations—are conceptually separate components, with the main difference being that the QNPU is expected to meet real-time requirements (to enable entanglement generation) and perform its arbitration tasks within set deadlines, whereas the CNPU does not need to provide such guarantees. This is because the QNPU should adhere to a network schedule which imposes real-time requirements. CNPU, QNPU and QDevice have a classical connection to their counterpart at the remote node, where the QDevice also has an additional optical fiber connection to the quantum network to perform quantum operations.

An implementation of the quantum network node could have these three top-level components (CNPU, QNPU and QDevice) deployed on three physically distinct environments, or group some of them on the same chip or board. That is, CNPU and QNPU could be realized on a single system, provided that this system has a connection to the quantum device to execute the actual quantum instructions. However, in the interest of a simpler implementation, where each system has a scoped responsibility, we here opted to map classical and quantum blocks onto two distinct environments. Classical blocks are run on a system that features a fully-fledged Operating System (OS) (here, Linux), with access to high level programming languages (like C++ and Python) and libraries. Quantum blocks are delegated to the *QNPU*, which is a system capable of interpreting quantum code blocks and managing the resources of a quantum device.

We note that the QNPU itself is an entirely classical system that interacts with the quantum hardware (the QDevice). At the moment, our implementation of the QNPU is fully software, including the instruction processor. In general, the system may be implemented entirely in software running on a classical CPU, or parts of its functionality may be implemented in classical hardware, e.g. Field Programmable Gate Array (FPGA) (see the description of the trapped-ion platform implementation in **??**) or Application-Specific Integrated Circuit (ASIC).

Quantum Network Programs

A quantum networking user program is what a programmer writes on the CNPU, in a high-level language, through the use of some Software Development Kit (SDK). Classical code blocks can in principle be programmed in any language yielding an executable suitable to run on the CNPU. Fully-classical code blocks—which include local processing and communication with other end nodes—often produce input data for the next quantum code blocks. That is, a classical code block typically precedes a quantum code block whose instructions depend on external data coming from a remote end node. In the future, quantum blocks could include real-time execution constraints, for example, a deadline by which execution should be completed in order to reach a specific application performance while the quantum memory has a limited memory lifetime.

NetQASM. To express quantum code blocks, we make use of *Quantum Network Assembly Language (NetQASM)* [43] as an instruction set for quantum network programs, which is described in detail in [43]. Before this work, NetQASM has only ever been used to execute quantum network programs on simulated quantum network nodes, and has never been realized on hardware to execute quantum network applications.

The instruction set used in NetQASM for the quantum code blocks is similar to other Quantum Assembly Language (QASM) languages (see e.g. Refs. [51, 93, 94]), but it is extended to include instructions for quantum networking. We emphasize that NetQASM is not a strict requirement of QNodeOS, and other ways to express quantum code blocks could be used in other implementations. The instruction set of this language should support both computational and networking quantum instructions, as well as simple classical arithmetic and branching instructions to be used for real-time processing on the QNPU. It is the compiler's task to transform high-level blocks for the QNPU into NetQASM blocks.

NetQASM defines a notion of NetQASM subroutines, where each subroutine corresponds to a quantum block of code, specified by the compiler or programmer. We therefore use the term *quantum block* to refer to a *NetQASM subroutine* in the remainder of this text. A full list of operands that can appear in a NetQASM subroutine is given in [43, Appendix B]. NetQASM assumed subroutines would be executed on a form of QNPU (without specifying an architecture for the QNPU), potentially using a form of shared memory with CNPU. In the absence of a shared memory, NetQASM allowed classical variables inside subroutines to be kept on the QNPU, and accessed read-only by the CNPU via the NetQASM interface (see below). The CNPU can also specify classical constants for the use inside subroutines, as part of submitting a subroutine to the QNPU.

We use here the NetQASM SDK [95] to write programs, where the SDK compiles a quantum network program, written in Python, into a series of classical and quantum code blocks. This SDK was previously used to express programs on a simulated quantum network [96].

NetQASM Interface. Our interface between the CNPU and the QNPU (Section 5.8.2) includes the NetQASM interface defined in [43, Appendix A]. This interface in particular allows the CNPU to register a program on the QNPU, submit NetQASM subroutines, and access the results of said subroutines.

Program Processing Pipeline

CNPU Processing. When a program starts execution on the CNPU, a new CNPU process is created. As we separate the CNPU from the QNPU in our implementation, it is natural to rely on the properties of an existing classical operating system to take care of this function. In our implementation, we start a single program on the CNPU which then creates a thread (using standard Linux thread library [97]) for each CNPU process. The classical blocks belonging to the CNPU program are executed locally on the CNPU. These may involve some form of coordination with the remote CNPU of the user program, as well as pre- or post-processing of the results coming from NetQASM subroutines. While this can also be done later, when the program starts it will typically also establish a TCP/IP connection with the program running on the remote CNPU leading to the establishment of a TCP/IP socket that will be used for classical application level communication between the CNPUs.

The CNPU then registers the program on the QNPU. Later, NetQASM subroutines of these programs are sent from the CNPU to the QNPU through the *NetQASM interface*.

QNPU Processing. When a program is registered with the QNPU by the CNPU, the QNPU creates a user process to store program data and execution state. The QNPU also keeps track of NetQASM subroutines belonging to the user process, which may be submitted only later, as well as other run-time data analogous to what a typical process control block contains, useful for the execution of the program. As depicted in paperFigure 5.3a Figure 2a in the main text, a subroutine can, in general, be composed of three classes of instructions:

- *Quantum operations*: quantum physical operations, to be performed on the underlying quantum device;
- *Classical logic*: arithmetic and branch instructions, to be executed in-between quantum operations, useful to store results of quantum operations and to perform responsive decision-making;
- *Entanglement requests*: requests to generate an entangled qubit pair with a remote node in the network.

Classical logic is processed locally on the QNPU, and potentially results in the update of a process's data. This data includes NetQASM variables capturing measurement results, for example, that may latter be conveyed to the CNPU.

When the user process starts on the QNPU an Entanglement Request (ER) socket (see Section 5.8.3) is established with the remote QNPU that is used to associate later entanglement requests with the specific user process. Entanglement requests contained in the NetQASM subroutines are forwarded to the quantum network stack, which stores them together with other requests coming from network peers. Entanglement generation requests coming from other nodes in the network are received on the quantum network stack through the *Quantum Network Stack (QNetStack) interface*.

Quantum instructions are sent to the QDevice through the QDevice Driver (QDriver), which provides an abstraction of the QDevice interface. The QDriver translates NetQASM instructions into physical instructions suitable to the underlying physical platform.

QDevice Processing. Physical instructions are executed on the QDevice, the quantum processing and networking unit. The QDevice processing stack heavily depends on the underlying physical platform—for instance, NV centers in diamond, or Trapped Ions.

As we remarked in Section 5.8.1, a QDevice has two communication channels with its direct neighbors: a *classical channel*, used for low-level synchronization of the entanglement generation procedure and other configuration routines, and a *quantum channel*, typically an optical fiber, through which qubits can travel.

5.8.2. QNPU STACK

QNodeOS is an architecture consisting of multiple abstraction layers, as depicted in Figure 5.8. It is designed to be platform-independent, i.e., independent of the underlying quantum physical platform (quantum hardware) controlled by QNodeOS, where connections to different realizations of QDevice are captured by a platform-dependent QDriver. The implementation of QNodeOS itself is of course dependent on the classical physical platform(s) on which QNodeOS is implemented, including the physical interface between the CNPU and QNPU.

QNPU API

At the center of the stack lie the *QNPU API handler* layer and the *QNPU core* layer. The API handler is responsible for listening to system calls made to the QNPU API, and to relay these calls to the appropriate component inside of the core layer. Such system calls may originate from the CNPU via the *CNPU communication handler*, see again Figure 5.8.

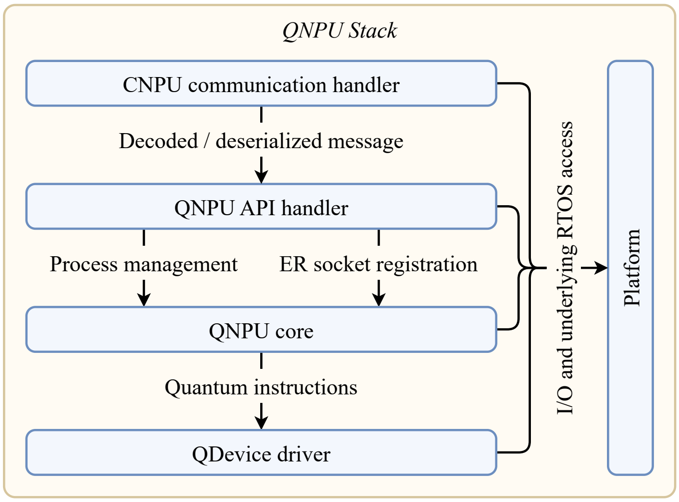


Figure 5.8: QNPU stack. The *QNPU API handler* and *the QNPU core* form the central processing layers, and are independent of the underlying quantum physical platform and of the device where QNodeOS runs. The *CNPU communication handler* translates protocol-specific messages from the CNPU into API calls. The *QDevice driver* (or QDriver) abstracts the QDevice hardware. The *Platform* layer abstracts the hardware where QNodeOS runs, and is accessible to all other layers. Note that three other Application Programming Interface (API) types are implemented, i.e. *control*, *management*, and *operations*. Control API is used for the network schedule, while management and operations API are for operational purposes.

The QNPU API is the central engine for managing the execution of local quantum operations and entanglement requests, and manages the hardware resources of the QDevice. The QNPU API exposes services to:

- *Register and deregister a program on the QNPU*; This is part of the NetQASM interface (see Section 5.8.1),
- *Add a quantum block (subroutine) for a user process*; This is again part of the NetQASM interface,
- *Open an ER socket with a remote node.* (NetQASM interface),
- *Control to configure the quantum network stack*, i.e., to configure the network schedule; This can be used for the interaction with a network controller that sets network-wide entanglement schedules, as presented in Ref. [55],
- *Perform management and operations functions.*

The topmost horizontal layer is the *CNPU communication handler*, which implements a *protocol wrapper* around NetQASM. We implement this wrapper protocol using EmbeddedRPC [98] for the on-the-wire definition of the messages (including (de-)serialization)). The communication handler translates protocol-specific messages into API calls for the QNPU. EmbeddedRPC allows to decouple the interface definition and (de-)serialization from the underlying transport layer. We note that only the transport layer is implementation-specific, which depends on the devices where CNPU and QNPU are implemented and on what the physical interface between them looks like.[1]

The *QDevice driver* (QDriver) layer, at the bottom of the stack, provides an abstraction of the QDevice hardware, and its implementation depends on the nature of the QDevice itself, and on the physical communication interface between QNPU and QDevice. Two QDevice implementations may differ in a variety of factors, including what quantum physical platform they feature and what digital controller interfaces with the QNPU.

Lastly, the vertical *Platform* layer provides System on a Chip (SoC)-specific abstractions for the QNPU to access the physical resources of the platform it is implemented on, including I/O peripherals, interrupts controllers and timers. Additionally, if the QNPU is implemented on top of a lower-level operating system, this layer gives access to system calls to the underlying OS. The Platform layer is vertical to indicate that it can be accessed by all other QNPU layers.

Porting the QNPU to a different SoC (or similar hardware) boils down to implementing a new platform layer. Deploying the QNPU on a different QDevice, instead, requires a new QDriver.

5.8.3. Processes

A quantum network program is divided into classical and quantum code blocks, i.e. NetQASM subroutines, where we will use the term subroutine and block interchangeably for quantum code blocks. The program starts on the CNPU, and creates a new process associated with the program. In the future, an optimized compilation could produce an

[1] TCP/IP for now, possibly a shared memory in the future.

executable that includes further information (such as execution deadlines depending on the device's memory lifetimes, as mentioned at the beginning of Section 5.8.1). The CNPU then registers the program with the QNPU (through the QNPU's end node API, see Section 5.8.2), which, in turn, creates its own process associated with the registered program. The process on the CNPU is a standard OS process, which executes the classical code blocks and interacts, (that is: communicating NetQASM subroutines and their results between CNPU and QNPU), with the counterpart process on the QNPU. This interaction can be done by means of a shared memory (and when no shared memory is physically realized: by an exchange of messages [43]). On the QNPU, a process encapsulates the execution of quantum code blocks of a program with associated context information, such as process owner, process number (ID), process state, and process priority.

In the near-term test applications we execute, the execution time of a program is typically dominated by that of quantum blocks, as entanglement generation is a time-consuming operation. Without advanced quantum repeaters [99], its duration grows exponentially with the distance between the nodes. For this reason, we focus on the scheduling of quantum blocks only, and thus we only discuss QNPU processes (also referred to as *user processes*) from this point onward. Again, this does not exclude that in a future iteration of the design CNPU and QNPU could be merged into one system, and therefore classical and quantum blocks would be scheduled jointly.

Process Types and Their Interaction

QNPU user processes. The QNPU allocates a new *user process* to each quantum network program registered by the CNPU. A user process is the program's execution context, and consists of NetQASM blocks and other context information—the process control block—including process number (ID), process owner, process state, process scheduling priority, program counter, and pointers to process data structures. Process state and priority determine how processes are scheduled on the QNPU. A user process becomes active (ready to be scheduled) as soon as the QNPU receives a quantum code block from the CNPU. Multiple user processes—relative to different CNPU programs—can be concurrently active on the QNPU, but only one can be running at any time. A running user process executes its quantum code block directly, except for entanglement requests, which are instead submitted to the quantum network stack and executed asynchronously.

QNodeOS network process. The QNPU also defines *kernel processes*, which are similar to user processes, but are created when the system starts (on boot) and have different priority values. Currently, the only existing kernel process is the *network process*. The network process, owned by the QNetStack, handles entanglement requests submitted by user processes, coordinates entanglement generation with the rest of the network, and eventually returns entangled qubits to user processes. The activation of the network process is dictated by a network-wide entanglement generation schedule. Such a schedule defines when a particular entanglement generation request can be processed, and therefore it has intersecting entries on adjacent nodes (given that entanglement is a two-party process). The schedule can be computed by a centralized network controller [55] or by a distributed protocol [35]. In our design, the network process follows a *time division multiple access schedule*, computed by a centralized network controller (as orig-

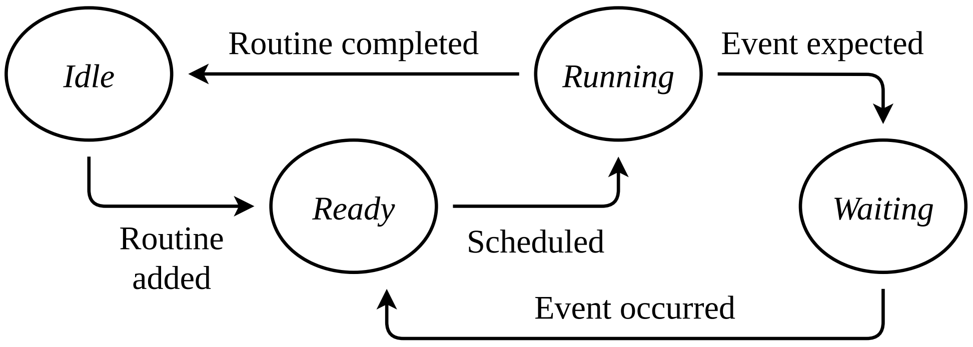


Figure 5.9: Process state diagram. An *idle* process becomes *ready* when a block for that process is loaded onto the QNPU (from the CNPU). A ready process becomes *running* when it is scheduled. A running process goes back to idle if all blocks are completed, or transitions to *waiting* if it expects an event to occur before it can proceed. A waiting process becomes ready again when the expected event occurs.

inally proposed by Skrzypczyk and Wehner [55]) and installed on each QNodeOS node (see Section 5.8.3).

QNPU process states. A QNPU process can be in any of the following states: (1) *Idle*: when the CNPU has registered a program and the QNPU has spawned a process, but it has not received a block yet; (2) *Ready*: when it has (at least) one block, sent from the CNPU, and can be scheduled and run; (3) *Running*: when it is running on the QNPU and has the quantum processor and the quantum network device assigned to it; (4) *Waiting*: when it is waiting for some event to occur. Figure 5.9 shows the possible process states and the valid state transitions. A process transitions from idle to ready when one block gets added. A ready process transitions to running when the QNPU scheduler assigns it to the processor. A running process transitions to waiting when it has to wait for an event to occur, and transitions from waiting to ready when the event occurs—for instance, a process could be waiting for an Entanglement Pair Request (EPR) pair to be generated, and become ready again when the pair is established. Finally, a process goes back to the idle state when all its blocks have been completed.

Inter-process communication. At the moment, the QNPU does not allow for any explicit inter-process communication. The only indirect primitive available to processes to interact with one another is *qubit ownership transfer*, used when a process produces a qubit state which is to be consumed by another process. Most notably, the quantum network stack kernel process transfers ownership of the entangled qubits that it produces to the process which requested the EPR pairs.

Process concurrency. The strict separation between local quantum processing and quantum networking is a key design decision in QNodeOS, as it helps us address the scheduling challenge, see Section 5.7. A user process can continue executing local instructions even after it has requested entanglement. Conversely, networking instructions can execute asynchronously of local quantum instructions. This is important in a quantum network, since entanglement generation must be synchronized with the neighboring node

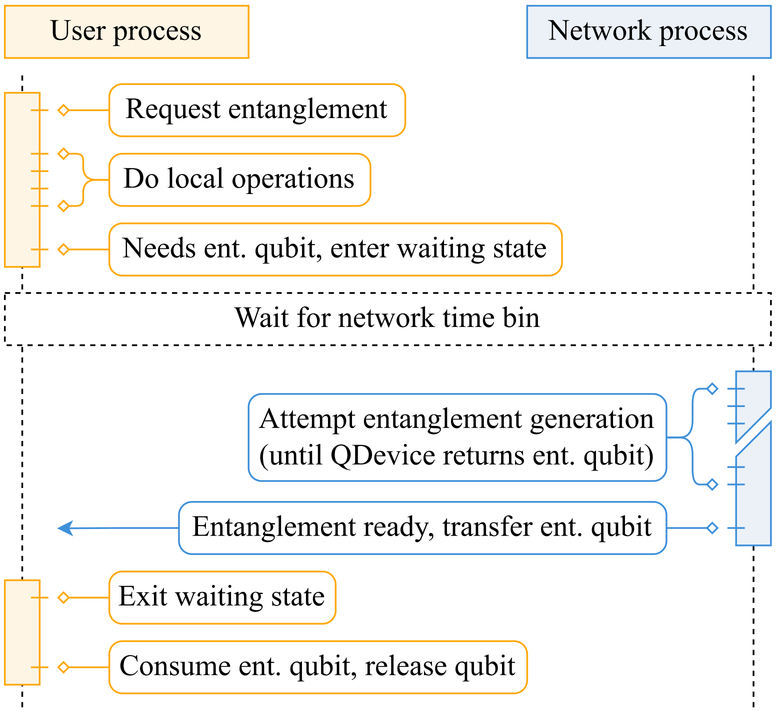

User process
Network process
Request entanglement
Do local operations
Needs ent. qubit, enter waiting state
Wait for network time bin
Attempt entanglement generation (until QDevice returns ent. qubit)
Entanglement ready, transfer ent. qubit
Exit waiting state
Consume ent. qubit, release qubit

Figure 5.10: Flow of execution between a user process requesting entanglement and the network process responsible for generating entanglement. The user process starts by asynchronously issuing an entanglement request. Once issued, it is free to continue with other local operations or classical processing. Once it reaches a point in its execution where entanglement is required the process enters the waiting state. The network process is scheduled once the appropriate time bin (as determined by the network schedule) starts. Once running, it attempts entanglement generation until entanglement success (or until a set timeout, such as the end of a time bin allocated to networking). The entangled qubit is then transferred to the user process. This unblocks the process which consumes the entanglement and releases the qubit. In our experiments, the process always immediately waits after requesting entanglement (no local operations are done in between).

(and possibly the rest of the network [55]). Additionally, separating user programs into user processes also allows QNodeOS to schedule several programs *concurrently*.

Process flow. Figure 5.10 illustrates the typical control flow between a user process and the network process. User processes are free to execute any non-networked instructions independently of the network process and other user processes. Once the program reaches a point in its execution where an entangled qubit is required, the process enters the waiting state and is flagged as waiting for entanglement. When the network process is scheduled, it issues network instructions and generates entanglement as requested by the user process. Once an entangled pair is generated by the network process, the qubit is handed over to the waiting user process. When all the entangled pairs that the user process was waiting for are delivered, the user process becomes ready and can start running again.

Process Scheduling

At present, the QNPU scheduler does not give any guarantees on when a process is scheduled—for that, one would need to define concrete real-time constraints to feed to the scheduler. Instead, the current version of the QNPU implements a best-effort scheduler, which selects processes on the basis of their *priority*, and does not allow preemption. In particular, the network process is assigned the highest priority, and is activated whenever the network schedule specifies entanglement should be made in the next time bin [55].

As already mentioned, QNodeOS defines the concept of *user* processes and *kernel* processes, with the QNetStack process being the only kernel process at the moment. User processes are released (i.e., they become ready) asynchronously—when a process block is loaded, or when they leave the waiting state—while the QNetStack process is released periodically—at the beginning of each time bin of the network schedule (although the period of time bins can vary). Given that generating an EPR pair on a link requires that both nodes attempt entanglement simultaneously, the QNPU assigns the QNetStack process a priority higher than any user process. This ensures that, at the beginning of each time bin of the network schedule, the *priority-based* process scheduler can assign the QNetStack process as soon as the processor is available, and thus a node can start attempting entanglement with its neighbor as soon as possible and minimize wasted attempts on the neighbor node.

Figure 5.11 exemplifies a snapshot of a hypothetical execution of a user process and the QNetStack process. The latter is activated at the beginning of a time bin assigned to networking, and is scheduled as soon as the processor is available—for instance, at times 0 and 4 it is scheduled immediately, while at time 8 it is scheduled after one time unit, as soon as the running process yields. The user process becomes ready at time 0—at which point the QNetStack process is ready as well and has highest priority, meaning the network process is scheduled; then it is scheduled at time 2, as soon as the QNetStack process completes; then it goes into waiting state at time 3 because the user process requested entanglement and it waits for the entanglement to be established; finally it becomes ready again at time 7—and it is scheduled immediately given that no other processes are running.

To avoid context switching overhead, potentially leading to degraded fidelity, the QNPU scheduler is *cooperative*. That is, once a process is scheduled, it gets to run until it either completes all of its instructions or it blocks waiting for entanglement. Allowing process preemption would need a definition of critical section and could potentially impact the quality of the affected qubit states. Moreover, the lack of a preemption mechanism could potentially result in low-priority user processes hogging the processor at the expense of high-priority entanglement generation attempts. On the other hand, if entanglement instructions always consume the entirety of the time bin, the QNetStack process would be immediately assigned the processor each time it relinquishes it, causing low-priority user processes to starve. To at least mitigate the second issue, we made sure that the number of consecutive entanglement attempts performed by the QDevice within one single entanglement instruction is always less than how many would fit in a time bin, so as to leave some slack for low-priority user processes to run.

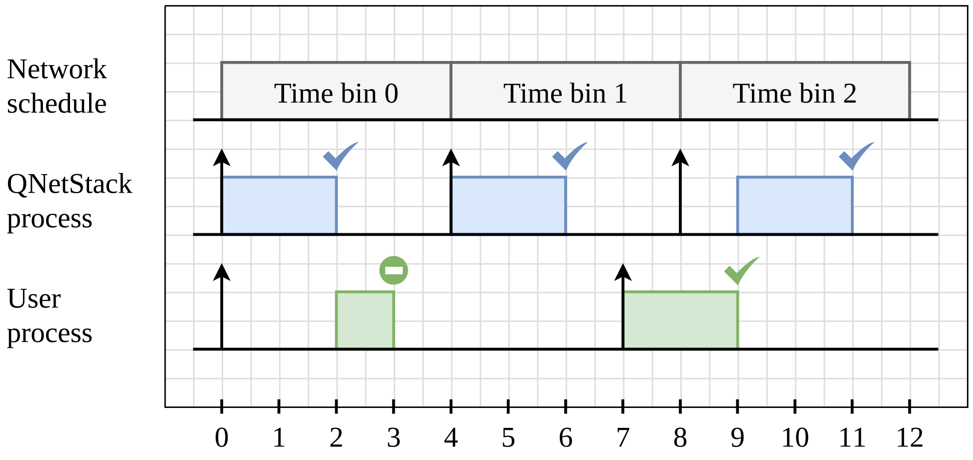


Figure 5.11: Snapshot of a hypothetical execution of a user process and the QNetStack process. The higher-priority QNetStack process is activated at the start of each time bin of the network schedule, and it is assigned to the processor as soon as it is available. The lower-priority user process gives precedence to the QNetStack process when they become ready at the same time, but, when it is running on the processor, it is not preempted if the QNetStack process becomes ready while the user process is running. Black arrows represents a moment where the process goes into the ready state and the green stop sign (at time 3) represents a process going into the waiting state.

Networking

The network stack QNetStack is based on the existing stack [35], including the link layer Quantum Entanglement Generation Protocol (QEGP) [35]. However the main difference between the QNetStack implemented on the QNPU and the original design of the protocols lies in how the QEGP processes the outstanding entanglement requests. QEGP [35] employed the concept of a distributed queue to sort and schedule entanglement requests on one node by coordinating with the counterpart node on the other end of the link, to ensure that both nodes would be servicing the same entanglement request at any given time. This synchronization is necessary because different entanglement requests may require different EPR pair fidelities, in which case QEGP would issue different QDevice entanglement instructions. However, link-local request scheduling becomes more complicated if nodes have more than just one link. In that case, entanglement requests would be better scheduled at a level where network-wide request schedules are known.

Network Schedule. The QEGP protocol implemented on the QNPU transitioned [39] from scheduling entanglement requests via a pairwise agreed upon distributed queue, to deferring this task to a *logically centralized control plane*, whereby a node's schedule can be computed on the basis of the whole network's needs by a (logically) centralized controller (see e.g. [55]). This means that the network stack of the nodes convey their demands for end-to-end entanglement generation to the central controller, who then makes a *network schedule*, which is communicated back to the nodes. We remark that our architecture does not depend on using a central controller, but could also use a dis-

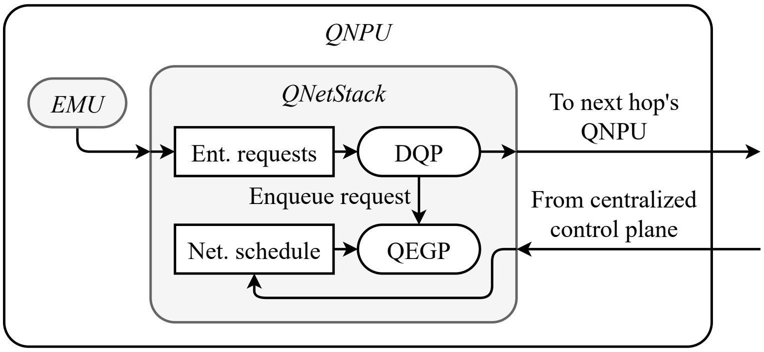


Figure 5.12: Internal components and data structures of the Quantum Network Stack (QNetStack). Entanglement requests are received through the Entanglement Management Unit (EMU), while the network schedule is installed by a centralized control plane. Quantum Entanglement Generation Protocol (QEGP) maps such requests onto the network schedule to produce the correct entanglement instructions. While not needed on our 2 node implementation, a Distributed Queue Protocol (DQP) (which is a simplified version of the DQP presented in [35, Section 5.2.1]) could forward entanglement requests to the next hop's Quantum Network Processing Unit (QNPU) to realize a network layer protocol such as [36].

tributed protocol to establish a network schedule.

All nodes divide time into time bins, where the central controller employs a scheduling algorithm to assign either network actions (or no actions) to time bins. That is, the term network schedule refers to a schedule, i.e. allocation of resources over time, of time bins at the nodes, where a time bin may be marked for networking activities (entanglement generation) or be left empty (to be used arbtirarily to execute local operations). Given that entanglement generation requires a non-deterministic amount of attempts and time, time bins are computed to be large enough to accommodate the (average) run time of an entanglement generation instruction. We remark that the node functions internally as a higher timing granularity than a time bin allocated by the network scheduler, that is, it can execute other operations (such as for example local quantum operations) also within a time bin allocated by the network schedule, provided entanglement is made early.

Once the node received the network schedule from the controller, the network schedule is used to satisfy all outstanding end-to-end entanglement requests, and is used by QEGP to produce the correct QDevice instructions at any point in time. Whenever a time bin is assigned to networking to two neighboring nodes, the nodes attempt entanglement generation over their shared link in order to realize the QEGP link layer protocol. Figure 5.12 shows internal components and data structures of the QNetStack as it is implemented on the QNPU. Entanglement requests received by the Entanglement Management Unit (EMU) are forwarded by Quantum Network Protocol (QNP) to the next hop's QNPU system. Entanglement requests and network schedule—the latter installed by a logically centralized control plane—are used by QEGP to produce the correct entanglement instructions to populate the QNetStack process's block at each activation of the process.

ER Socket. The concept of an ER socket is inspired by that of a classical network socket, in that it defines the endpoint of an entanglement generation request, and is used by the QNPU's quantum network stack to set up network tables and to establish connections with its peers. We remark that the current realization of the ER Socket (see below) is a proof of concept implementation opening future computer science research, and does not aim to prevent misuse if different users had access to the same node. A program can request from QNodeOS the opening of an ER socket with a program on a remote node. An ER socket is identified by the tuple (`node_id, er_socket_id, remote_node_id, remote_er_socket_id`). The other program (on the other node) must open its own corresponding ER socket (i.e with values (`remote_node_id, remote_er_socket_id, node_id, er_socket_id`)) on its own QNodeOS. A request for opening an ER socket is executed by the CNPU, by asking the QNPU (through the QNPU API) to open the socket. The QNPU then registers the ER socket with the quantum network stack (provided it did not yet exist), and the CNPU also keeps a reference using the tuple as an identifier. The program can then use this socket for requests. The network stack only handles requests for entanglement between two nodes if the corresponding ER sockets are opened on both nodes.

Programs are themselves responsible for coordinating the ER socket IDs. Using these IDs allow the same node pair to open multiple pairs of ER sockets, which may be used by different applications or inside the same application. Socket IDs must be unique within the node. ER sockets are typically opened at the start of a program, and live (and may be used multiple times) until the program finishes.

Programs use the ER socket to submit entanglement requests to the network stack. This is done through NetQASM instructions (`create_epr` and `recv_epr`) that refer to the ER socket in their operands. One program must execute a `create_epr` instruction and the other a `recv_epr` instruction (to be coordinated by the programs themselves). The program executing the `create_epr` instruction is treated by the network stack as the *initiator* and the program executing `recv_epr` the *receiver*. Upon receiving an entanglement request, the network stacks of the two nodes communicate between each other in order to coordinate entanglement generation. The initator node always initiates this communication. The receiver node always accepts the entanglement initiative as long as the corresponding ER socket is open. This means that the receiver node agrees with entanglement generation as soon as the initiator node has submitted an entanglement request (through its `create_epr`), even if the receiver node itself has not yet submitted its corresponding request (through its `recv_epr`). On the receiver node, the generated entangled qubit will remain in memory until it gets asked for by a user process executing this `recv_epr`.

MULTITASKING

Multitasking forms an essential element of our architecture already at the level of scheduling the network process in relation to any user process, to address the challenges inherent in the way entanglement is produced at the physical layer, requiring agreement on a network schedule (see Section 5.8.3, and main paper). For this important reason, the QNPU is designed to arbitrate between these two processes (see Figure 5.10), and to manage the resources being used by each of them. *Multitasking*, hence, is a funda-

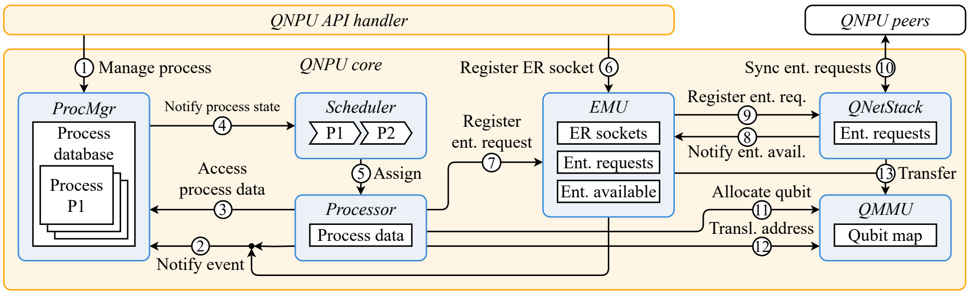


Figure 5.13: Quantum Network Processing Unit (QNPU) core components and internal interfaces. The core layer includes: (1) a *process manager* (ProcMgr), which owns and manages access to QNPU processes; (2) a *scheduler*, responsible for selecting the next process to be run; (3) a *processor*, which processes blocks' instructions; (4) an *EMU*, which keeps a list of entanglement requests and available entangled qubits; (5) a *QNetStack*, whose responsibility is to coordinate with peer nodes to schedule quantum networking instructions; (6) a *QMMU*, which keeps a record of allocated qubits.

mental requirement for our system managing the hardware of a quantum network node, especially while such hardware has only limited resources available.

To further increase the utility of the system, we also allow the multitasking of user processes (main paper): Like in most operating systems, these tasks, which on the QNPU are encapsulated into processes, can sometimes necessitate a resource which is not immediately available—for instance, a free qubit, or a qubit entangled with a remote one. Maximizing the utilization of the quantum device is one of the goals of QNodeOS, whose design allows multiple processes, user and kernel, to be active concurrently, so that whenever one is in a waiting state, another one can potentially be scheduled to use the quantum device. This design aspect is relevant for quantum networking nodes, as the execution of the local program is often waiting, both for classical messages from remote nodes, as well as the generation of entanglement.

Lastly, multitasking is an important feature for systems that are to be shared by multiple users, and that offer each user the possibility to run multiple programs concurrently.

5.8.4. QNodeOS Components and Interfaces

We provide here additional details on the components of the QNPU architecture and their interfaces. Figure 5.13 gives an overview of all the components of the QNPU. The *process manager* marshals accesses to all user and kernel processes. The *scheduler* assigns ready processes to the *processor*, which runs quantum instructions through the underlying QDevice, processes classical NetQASM instructions locally, and registers entanglement requests with the *EMU*. The EMU maintains a list of ER sockets and entanglement requests, forwards the latter to the *quantum network stack*, which, in turn, registers available entangled qubits with the EMU. Finally, the *QMMU* keeps track of used qubits, and transfers qubit ownership across processes when requested.

PROCESS MANAGER

The process manager owns QNPU processes and marshals accesses to those. Creating a process, adding a block to it and accessing the process's data must be done through the process manager. Additionally, the process manager is used by other components to notify *events* that occur inside the QNPU, upon which the state of one of more processes is updated. Process state updates result in a notification to the scheduler.

Interfaces. The process manager exposes interfaces for three services:

- *Process management* (interface 1 in Figure 5.13): to create and remove processes, and to add quantum blocks to them. When the user registers a program, the QNPU API Handler uses the process manager to create a QNPU user process. The returned process ID can be later used to add a block to that process, or to remove the process once all its blocks are fully processed.

- *Event notification* (interface 2 in Figure 5.13): to notify an event occurred inside the QNPU, including the addition of a block, the completion of a block, the scheduling of the process, the hitting of a *Waiting* condition (see Figure 5.9), and the generation of an entangled qubit destined to the process. Some events trigger follow-up actions—for instance, when a process that was waiting for an event becomes ready, it gets added to the queue of ready processes maintained by the scheduler.

- *Process data access* (interface 3 in Figure 5.13): to access a process's blocks and its classical memory space, mostly used while running the process (through the processor).

SCHEDULER

The QNPU scheduler registers processes that are ready to be scheduled, and assigns them to the QNPU processor when the latter is available. Ready processes are stored in a *prioritized ready queue*, and processes of the same priority are scheduled with a first-come-first-served policy.

Interfaces. The scheduler only exposes one interface for process state notifications (interface 4 in Figure 5.13), used by the process manager to signal when a process transitions to a new state. When a QNPU process transitions to the ready state, it is directly added to the scheduler's prioritized ready queue. When a process becomes idle, or is waiting for an event to happen, the scheduler simply registers that the processor has become available.

PROCESSOR

The QNPU processor handles the execution of QNPU user and kernel processes, by running classical instructions locally and issuing quantum instructions to the QDriver. It is also responsible for multitasking by means of process manager. While executing a process, the processor reads its blocks and accesses (reads and writes) its classical memory. The processor implements a specific instruction set architecture dictated by the NetQASM language of choice.

Interfaces. The processor exposes one interface for processor assignment (interface 5 in Figure 5.13), used by the QNPU scheduler to activate the processor, when it is idling, and assign it to a QNPU process.

Entanglement Management Unit

The Entanglement Management Unit (EMU) contains a list of open *ER sockets* and a list of *entanglement requests*, and keeps track of the *available entangled qubits* produced by the quantum network stack. Received entanglement requests are considered valid only if an ER socket associated to such requests exists. Valid requests are forwarded to the quantum network stack. Entangled qubit generations are notified as events to the process manager.

Interfaces. The EMU exposes interfaces for three services:

- *ER socket registration* (interface 6 in Figure 5.13): to register and open ER sockets belonging to a program, and to set up internal classical network tables and to establish classical network connection.
- *ER registration* (interface 7 in Figure 5.13): to add entanglement requests to the list of existing ones, to be used when matching produced entangled qubits with a process that requested them.
- *Entanglement notification* (interface 8 in Figure 5.13): to register the availability of an entangled qubit, produced by the quantum network stack, and to link it to an existing entanglement request.

Quantum Network Stack

The quantum network stack on the QNPU closely follows the model presented by Dahlberg et al. [35] which is based on the classical Open Systems Interconnect (OSI) network stack model for the purpose of the separation of responsibilities. In particular, *data link layer* is part of the quantum network stack on the QNPU. The *physical layer* is implemented on the QDevice, the *application layer* is part of the CNPU and QNPU.

The quantum network stack component has an associated *QNPU kernel process*, created statically on the QNPU. However, this process's block is dynamic: the instructions to be executed on the processor depend on the outstanding entanglement generation requests received from EMU and network peers.

Interfaces. The quantum network stack exposes interfaces for two services:

- *Entanglement request registration* (interface 9 in Figure 5.13): to add entanglement requests coming from the EMU to the list of existing ones, which are used to fill in the quantum network stack process's block with the correct quantum instructions to execute.
- *Entanglement request synchronization* (interface 10 in Figure 5.13): similar to the entanglement request registration interface, but to be used to synchronize (send and receive) requests with QNodeOS network peers.

QUANTUM MEMORY MANAGEMENT UNIT

Quantum Memory Management Unit (QMMU) receives requests for *qubit allocations* from QNPU processes, and manages the subsequent usage of those. It also translates NetQASM *virtual qubit addresses* into QDevice qubit addresses for the QDevice, and keeps track of which process is using which qubit at a given time. In general, a QMMU should take into account that the topology of a quantum memory determines what operations can be performed on which qubits, and thus allow processes to allocate qubits of a specific type upon request. An advanced QMMU could also feature algorithms to move qubits in the background—that is, without an explicit instruction from a process's block—to accommodate a program's topology requirements while not trashing the qubits being used by other QNPU processes. Such a feature could prove crucial to increase the number of processes that can be using the quantum memory at the same time, and to enhance multitasking performances.

Interfaces. The QMMU exposes interfaces for three services:

- *Qubit allocation and de-allocation* (interface 11 in Figure 5.13): a running process can ask for one or more qubits, which, if available, are allocated by the QMMU, and the QDevice qubit addresses of those are mapped to the virtual addresses provided by the requesting process.
- *Virtual address translation* (interface 12 in Figure 5.13): before sending quantum instructions to the QDriver, the processor uses virtual qubit addresses specified in NetQASM to retrieve QDevice qubit addresses from the QMMU, and then replaces virtual addresses with QDevice qubit addresses in the instructions for the QDriver.
- *Qubit ownership transfer* (interface 13 in Figure 5.13): qubits are only visible to the process that allocates them. However, in some cases, a process may wish to transfer some if its qubits to another one. A notable example is the quantum network process transferring an entangled qubit to the process that will use it.

5.8.5. QNPU IMPLEMENTATION: SCHEDULER

The QNPU scheduler is an important component of our QNodeOS architecture, and deals with scheduling of QNPU processes. The QNPU is implemented on FreeRTOS [76], which itself includes a scheduler. On FreeRTOS, code is organized into tasks, which can be seen as separate threads or processes. These tasks are scheduled concurrently by FreeRTOS based on priority. In our implementation, we realize QNPU components and interfaces (hence including the QNPU scheduler, which is distinct from the FreeRTOS scheduler) as FreeRTOS tasks. We configured task priorities such that the components with tight interaction with the QDevice (QDriver, quantum network stack, QNPU processor) have highest priority. We stress the difference between the FreeRTOS scheduler and our QNPU process scheduler. The QNPU scheduler schedules QNPU processes based on their status and priorities, which are independent of the priorities assigned by the FreeRTOS scheduler. The FreeRTOS hence runs on a different layer: it makes sure the QNPU components (including QNPU scheduler, processor, QDriver) run concurrently. The QPNU scheduler runs on the level of QNPU processes. Whenever the FreeRTOS scheduler activates the FreeRTOS task realizing the QNPU scheduler, the QNPU scheduler then schedules the process with the highest priority on a first come first serve basis, by adding it to the processing queue of the relevant resource (e.g. QNPU processor) and generating an interrupt leading to the execution of the QNPU processor task on FreeRTOS (and consequently the execution of the process).

5.8.6. QDevice Interface

The implementation of a QDevice depends on a number of factors. Most importantly, the physical signals that are fed to the quantum processing and networking device (and those that are output from the device) are specific to the nature of the device itself. Different qubit realizations require different digital and analog control. For instance, manipulating the state of a spin-based qubit (e.g., in a NV center processor) and that of an atom qubit (e.g., in a trapped ion processor) are two physical processes that vastly differ in a number of complex ways.

For QNodeOS to be portable to a diverse set of quantum physical platforms, there needs to be a common *QDevice interface* that QNodeOS can rely on, and that each QDevice instance can implement as it is most convenient for the underlying quantum device. This interface (1) needs to be *general*, (2) to be able to *express all quantum operations* that different quantum devices might be capable of performing, and (3) *abstract*, so that two different implementations of a well-defined qubit manipulation operation can be expressed with the same instruction on QNodeOS. Nevertheless, an interface that is too general could result in a high implementation complexity on the QDevice, as it might have to transform high-level instructions in a series of native operations on the fly. Other than complexity of implementation, a very high-level set of QDevice instructions might compromise the compiler's ability to optimize a program for a certain physical platform, as reported by Murali et al. [100].

Design Choices

Defining a set of instructions to express abstract quantum operations as close as possible to what different quantum physical platforms can natively perform is—to some extent—an open problem. Nonetheless, we have made an effort to specify an interface which is a good compromise between generality and expressiveness. The QDevice interface is essentially a set of instructions that QNodeOS expects a QDevice to implement. To be precise, a QDevice might implement a subset of the interface, according to what native physical operations it can perform. The CNPU compiler must then have knowledge about the set of instructions implemented by the underlying QDevice, so that it can decompose instructions that are not natively supported.

Even though this interface does not impose any formal timing constraints, it is important to note that a QDevice implementation that tries to guarantee more or less deterministic instruction processing latencies can prove more beneficial to the real-time requirements of the QNPU. Particularly, it would be advisable to time-bound the processing time of operations whose duration is by nature probabilistic—most notably, those involving entanglement generation. Creating an EPR pair may involve a varying number of attempts. Sometimes, if the remote node becomes unresponsive for some time, the number of necessary attempts can increase by a large amount. Capping the num-

Instruction	Description
`INI`	Initialize a qubit to default state
`SQG`	Perform a single-qubit gate
`TQG`	Perform a two-qubit gate
`AQG`	Perform a gate on all qubits
`MSR`	Measure a qubit in a specified basis
`ENT`	Attempt entanglement generation
`ENM`	Attempt entanglement and measure qubit
`MOV`	Move qubit state to another qubit
`SWP`	Swap the state of two qubits
`ESW`	Swap qubits belonging to two EPR pairs
`PMG`	Set pre-measurement gates

Table 5.1: Summary of QDevice instructions defined in the QDevice interface. A specific QDevice might implement a subset of these, depending on the underlying quantum physical device and on other design constraints.

ber of attempts could, for instance, provide a more deterministic maximum processing latency for entanglement instructions, which in turn might help QNodeOS react more timely to temporary failures or downtime periods of remote nodes. Not to mention that unbounded entanglement attempts affect the state of other qubits in memory, because of both passive decoherence and cross-qubit noise.

In the following sections we describe the interactions between QNPU and QDevice.

QDevice Synchronization

The QDevice receives physical instructions from QNodeOS, acts on them, and returns a response. For entanglement instructions, the QDevice must first synchronize with the QDevice on the other node (using classical communication). If the other QDevice is busy, (e.g. it is still trying to pass the CR check, see Section 5.9.1 and [39]), synchronization fails, and an `ENT_SYNC_FAIL` response is returned (see Table 5.2).

Instructions and Operands

Table 5.1 lists the complete set of instructions defined in the QDevice interface. Instructions can have operands, whose range of valid values depends on the underlying QDevice. For instance, an operand that specifies which qubit to apply an operation to can only have as many valid values as there are qubits in memory. Details for each instruction and its operands are given below.

Qubit Initialization (`INI`). The `INI` instruction brings a qubit to the $|0\rangle$ state. On some physical platforms, single-qubit initialization is not possible, thus this instruction initializes all qubits to the $|0\rangle$ state.

Operand	Description
`qubit`	Physical address of the qubit to initialize, ignored on platforms where single-qubit initialization is not possible

Single-Qubit Gate (`SQG`). The `SQG` instruction manipulates the state of one qubit. The gate is expressed as a rotation in the Bloch sphere.

Operand	Description
`qubit`	Physical address of the qubit to manipulate
`axis`	Rotation axis, can be X, Y, Z or H (support is QDevice-dependent)
`angle`	Rotation angle (granularity and range are QDevice-dependent)

Two-Qubit Gate (`TQG`). The `TQG` instruction manipulates the state of two qubits. The gate is expressed as a controlled rotation, with one qubit being the control and the other one being the target.

Operand	Description
`qub_c`	Physical address of the control qubit
`qub_t`	Physical address of the target qubit
`axis`	Rotation axis, can be X, Y, Z or H (support is QDevice-dependent)
`angle`	Rotation angle (granularity and range are QDevice-dependent)

All-Qubit Gate (`AQG`). The `AQG` instruction manipulates the state of all available qubits. The gate is expressed as a rotation in the Bloch sphere.

Operand	Description
`axis`	Rotation axis, can be X, Y, Z or H (support is QDevice-dependent)
`angle`	Rotation angle (granularity and range are QDevice-dependent)

Qubit Measurement (`MSR`). The `MSR` instruction measures the state of one qubit in a specified basis. A qubit measurement is destructive—that is—the qubit has to be reinitialized before it can be used again.

Operand	Description
`qubit`	Physical address of the qubit to measure
`basis`	Measurement basis, can be X, Y, Z, H (support is QDevice-dependent)

Entanglement Generation (`ENT`). The `ENT` instruction performs a series of entanglement generation attempts, until one succeeds, or until a maximum number of attempts is reached (the behavior is QDevice-dependent).

Operand	Description
`nghbr`	Neighboring node to attempt entanglement with, if the local QDevice has multiple quantum links
`fid`	Target entanglement fidelity (granularity and range are QDevice-dependent)

Entanglement Generation With Qubit Measurement (`ENM`). The `ENM` instruction performs a series of entanglement generation attempts followed by an immediate measurement of the local qubit, until one succeeds, or until a maximum number of attempts is reached (the behavior is QDevice-dependent).

Operand	Description
`nghbr`	Neighboring node to attempt entanglement with, if the local QDevice has multiple quantum links
`fid`	Target entanglement fidelity (granularity and range are QDevice-dependent)
`basis`	Measurement basis, can be X, Y, Z, H (support is QDevice-dependent)

Qubit Move (`MOV`). The `MOV` instruction moves the state of one qubit to another qubit. A qubit move renders the state of the source qubit undefined, and the qubit has to be reinitialized before it can be used again.

Operand	Description
`qub_s`	Physical address of the source qubit
`qub_d`	Physical address of the destination qubit

Qubit Swap (`SWP`). The `SWP` instruction swaps the state of two qubits.

Operand	Description
`qub_1`	Physical address of the first qubit
`qub_2`	Physical address of the second qubit

Entanglement Swap (`ESW`). The `ESW` instruction results in two qubits belonging to two EPR pairs to have their roles swapped.

Operand	Description
`qub_1`	Physical address of the first qubit
`qub_2`	Physical address of the second qubit

Pre-Measurement Gates Setting (`PMG`). The `PMG` instruction allows for a set of (up to) 3 rotations to be performed before a qubit measurement (`MSR` or `ENM`). If the axis of the second rotation is orthogonal to the axis of the first and the third rotation, these gates can be used to perform a qubit measurement in an arbitrary basis, given that most likely a QDevice can natively measure in a limited set of bases.

Operand	Description
`axes`	Combination of orthogonal axes to use for the three successive rotations, can be X–Y–X, Y–Z–Y and Z–X–Z (support is QDevice-dependent)
`ang_1`	Rotation angle of the first gate, relative to the first axis in `axes` (granularity and range are QDevice-dependent)
`ang_2`	Rotation angle of the second gate, relative to the second axis in `axes` (granularity and range are QDevice-dependent)
`ang_3`	Rotation angle of the third gate, relative to the third axis in `axes` (granularity and range are QDevice-dependent)

No operation (`NOP`). The `NOP` instruction does not result in any operation on the QDevice.

Return values

Table 5.2 lists the possible return values that the QDevice sends back to QNodeOS as a response to a physical instruction.

Physical Instruction	Return values	Description
`INI`, `SQG`, `TQG`, `AQG`, `PMG`	`SUCCESS`	Always successful
`MSR`	`SUCCESS_0` or `SUCCESS_1`	Measurement outcome is 0 or 1*
`ENT`	`SUCCESS_<state>`	Entanglement generation was successful; the state is <state>†
`ENM`	`SUCCESS_<state>_<outcome>`	Entanglement generation was successful; state was <state>† and outcome is <outcome> (0 or 1)
`ENT, ENM`	`ENT_FAILURE`	Entanglement generation was attempted and failed
`ENT, ENM`	`ENT_SYNC_FAILURE`	Entanglement generation was not attempted since synchronization failed (other node is busy)

Table 5.2: List of physical instructions (sent from QNodeOS to QDevice) and their possible return values (sent from QDevice to QNodeOS). *Measurements are always in the Z basis, where outcome 0 corresponds to $|0\rangle$ and outcome 1 to $|1\rangle$. † Possible states depend on the implementation. For NV these are `PSI_PLUS` and `PSI_MINUS`, see Section 5.9.1.

5.9. QDevice Implementations

5.9.1. NV Center Platform

The QDevice employed for the full stack benchmark experiments is constituted by an NV center processor. We use the NV center in its negatively charged state (called NV$^-$) for quantum information processing. NV$^-$ is a spin-1 system, whose ground states are non-degenerate in the presence of an external magnetic field, see Figure 5.14 [14]. We employ the $m_s = 0$ as our $|0\rangle$ state for the qubit, while for the $|1\rangle$ we can choose one of $m_s = \pm 1$. Details on how the choice is made will follow in the next section. The NV can be optically excited resonantly (637 nm) and off-resonantly (typically 532 nm), and it emits in 3% of the cases single photons (Zero-Phonon Line (ZPL) photons), while the remaining part is constituted by the emission of a photon and a phonon (Phonon-Side Band (PSB)). The optical transitions are spin-selective, as shown in Figure 5.14. In the presence of lateral strain and external DC field (Stark effect), the excited states of the NV split apart, maintaining their spin-selective properties. In this work, we use a natural lateral strain between 2 GHz and 5 GHz. The cycling transition denoted as Readout (RO) in Figure 5.14 is used to emit single photons (ZPL) for entanglement generation and to read out the state of the qubit (fluorescence in the PSB). From the excited states, the NV can also decay through metastable states (not shown in Figure 5.14). The preferable decay from such metastable states is the $m_s = 0$ state. In this way, it is possible to optically initialize the qubit state to $|0\rangle$ (dashed line in Figure 5.14), with fidelity above 99%, when on-resonantly exciting the Spinpump (SP) transition and averaging for long enough to ensure a spin-flip. In our experiments, we apply a laser field on resonance with the SP transition at 500 nW for 1.5 μs for fast initialization during entanglement attempts, whereas a slow initialization (10 nW for 100 μs) is used for single-qubit gates experiments (like local tomography). On the other hand, while exciting the RO transition, decays to $m_s = \pm 1$ are also possible, but they present longer cyclicity. This feature is relevant for the optical read-out of the qubit state, which can be done in a single shot and is discussed in the following sections.

In our demonstration, the server has an external magnetic field of $B_z = 189$ mT aligned along the symmetry axis z of the NV, while the client experiences $B_z = 23$ mT. The magnetic field is applied via permanent magnets placed both inside and outside the high-vacuum chamber of our closed-cycle cryostats. Fluctuations of the magnetic field are observed on the order of nT on a timescale of hours, therefore they do not constitute a limitation for our demonstration. We also measured a perpendicular component of the permanent magnetic field for both setups of ~1 mT. Such misalignment becomes crucial for the coherence time of the electron spin qubit, as in the interaction with the surrounding nitrogen nuclear spins, the off-axis hyperfine interaction terms become non-negligible and the decoupling of the electron spin is harder [14]. Notably, the server node is in the regime of “high magnetic field”. In the level structure depicted in Figure 5.14, this means that the $m_s = -1$ ground state crossed the $m_s = 0$ state (at ~ 100mT), and the optical transitions for the SP are well separated, such that a double laser field with proper detuning is necessary to correctly address both of them.

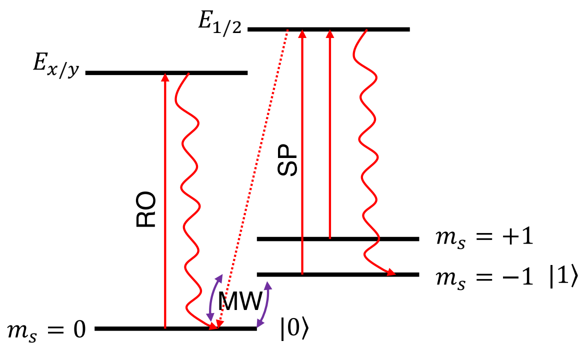


Figure 5.14: Energy structure of NV$^-$ at 4 K. The ground state of the NV splits into three distinct levels (Zeeman splitting). The optical transitions are spin-selective. The excited states are represented as one, but they are non-degenerate when lateral strain is applied. We denote as Readout (RO) the transition $|0\rangle \rightarrow E_{x/y}$ and as Spinpump (SP) the transition $|1\rangle \rightarrow E_{1/2}$. The wiggly lines represent the photoluminescence when such transitions are excited, whereas the dashed lines represent the decay via metastable states that is used for initialization of the qubit state into $|0\rangle$. Microwave (MW) pulses enables the transfer of population between the two states of the qubit, allowing for quantum information processing.

Single Node Operations

In this section, details on how to operate a single node for quantum information processing are given. The physical setup is the one employed for the demonstration in Ref. [39].

Charge-resonance check. To use the NV as a processing node, it is necessary to guarantee that it is in the correct charge state and the laser fields are on resonance with the transitions. Before executing any instructions coming from the QNPU, both nodes go through the so-called Charge-Resonance (CR) check. We apply resonant fields for 100 μs on both the RO (1 nW) and SP (10 nW) transitions and we monitor the fluorescence. If the number of photons exceeds the threshold (25 for the client and 60 for the server for our experiments), the node is considered ready to accept instructions from the QNPU and can proceed with synchronization with the other node (for multinode instructions). The threshold is set considering the brightness of each NV. The success is considered valid for 100 ms. After this time, if no instructions arrive, the CR check is repeated. In case the number of photons is below the threshold, we distinguish two cases: (1) the counts are between the success threshold and a second threshold called Repump: we repeat the CR check and tune the frequency of the red lasers, as they might not address the transitions correctly; (2) the number of counts is below the Repump threshold (set at 15 for the client and 25 for the server): this means that the NV might be in the dark charge state (NV^0) due to ionization. To restore the charge state, in the next round of CR check we first illuminate with off-resonant green laser (20 μW for 50 μs), or, for the client node only, with yellow light (575 nm, 35 nW for 300 μs) on resonance with the ZPL transition of NV^0 [101]. This is necessary because we additionally apply an external DC field to the NV on the client node. We, indeed, exploit the Stark effect to tune the RO transition to be the same as the server's one [102]. In this way, we can ensure photon indistinguishability in frequency that is crucial for entanglement generation. The typical DC field used for this work is ~2V, modulated via an error signal that is computed on the CR check counts,

	Client	Server
Duration π rotation	200 ns	190 ns
Amplitude π rotation	0.78	0.89
Skewness π rotation	-1.5e^{-9}	-3.5e^{-9}
Duration $\pi/2$ rotation	150 ns	100 ns
Amplitude $\pi/2$ rotation	0.38	0.56
Skewness $\pi/2$ rotation	-1.2e^{-8}	-7.1e^{-9}
Power	42 W	42 W

Table 5.3: Characterizing values for the MW pulses. Other rotation angles have the same duration and skewness as the π pulse, and the amplitudes scale accordingly. The rotation axes are obtained by changing the phase of the pulse. With the current setup configuration, only rotations along $\hat{x}$ and $\hat{y}$ axes are feasible, so $\hat{z}$ rotations are compiled as combinations of gates along $\hat{x}$ and $\hat{y}$.

acting as a Proportional–Integral–Derivative (PID) loop.

The CR check is repeated after an experiment iteration. This round is utilized to validate the experiment and post-select the results based on success or failure of this procedure, as discussed in paperSection 5.6 the Methods section of the main text.

Single qubit gates. To manipulate the state of the electron spin qubit, microwave pulses are on resonance with the transition $|0\rangle \rightarrow |1\rangle$ are employed. For the server node, the $m_s = -1$ state is used as $|1\rangle$ and the resonance frequency is 2.4 GHz. The client node utilizes the $m_s = +1$, with a resonance frequency of 3.5 GHz. The choice of the $|1\rangle$ is made based on the gate fidelity.

We use skewed-Hermite Microwave (MW) pulses [103, 104] with high Rabi frequency (~10 MHz), which generates an alternating magnetic field capable of manipulating the state of the qubit. The characterizing values for the two nodes are reported in Table 5.3. The measured infidelity on a single MW pulse is below 1%. Instructed by the QNPU, we performed local quantum tomography on both the server and the client, showing high fidelity. One example is reported in Figure 5.19.

Dynamical decoupling. Once MW pulses are set up with high fidelity, it is possible to implement Dynamical Decoupling (DD) sequences that increase the coherence time of the electron spin qubit. DD sequences are especially crucial in our experiments when the latency of the QNPU is long (milliseconds timescale), like in the Delegated Quantum Computation (DQC) demonstration. The characterizing parameter for a DD sequence is the time delay between the X and Y pulses. To optimize it, we swept the interpulse delay, at the sample precision of our Arbitrary Waveform Generator (0.42 ns, Zurich Instruments HDAWG), while playing the effective single-qubit computation of the DQC protocol instructed by the QNPU, as explained in paperSection 5.6 the Methods section in the main text, on both the client and the server. In doing so, we added an extra waiting time of 5 ms between the initialization of the qubit into the superposition state and the subsequent gates to mimic the real-case scenario of the DQC. In this way, we are able to set the optimal interpulse delay, obtaining a single-qubit fidelity of 0.96(2) for the server and 0.88(2) for the client.

Single-shot readout. When a measurement instruction arrives from the QNPU, this is translated by the physical layer as a Single-Shot Readout measurement. To assign a state to the qubit, we can use the RO optical transition. The RO laser field is on for $\sim 10\ \mu$s at 1 nW. This will produce fluorescence only if the NV is in the $|0\rangle$ state. If no photons are detected while the laser is on, the outcome is assigned to the $|1\rangle$ state. The fidelity of the measurement process is defined as $F = 1/2(F_{0|0} + F_{1|1})$, where $F_{0|0}$ ($F_{1|1}$) represents the fidelity of measuring $|0\rangle$ ($|1\rangle$) when the qubit is prepared in $|0\rangle$ ($|1\rangle$). For our experiments, we obtain 0.841(4) and 0.997(1) respectively for the client, and 0.912(3) and 0.995(1) for the server, achieving above 0.90 of process fidelity.

ENTANGLEMENT GENERATION

The entanglement request from the QNPU is translated into executing a single-photon protocol. The communication qubit on each node is initialized in the state $\sqrt{\eta}|0\rangle + \sqrt{1-\eta}|1\rangle$, where η represents the bright state population. For maximum state fidelity, the condition $\eta_C p_C \approx \eta_S p_S$ applies, where $\eta_{C(S)}$ is the bright state population of the client (server) and $p_{C(S)}$ is the photon detection probability of the client (server). In this work, $\eta_C = 0.07$ and $\eta_S = \eta_C \frac{p_C}{p_S} = 0.04$. The choice of η_C is a trade-off between entangled state fidelity and entanglement generation rate. The produced entangled state is (non-deterministically) one of two Bell states $|\Psi^\pm\rangle = \frac{1}{\sqrt{2}}(|01\rangle \pm e^{i\Delta\theta}|10\rangle)$, based on which detector clicked at the heralding station. The phase $\Delta\theta$ is actively stabilized [22] before the execution of the entanglement request, via a combination of homodyne interference, for the global phase of the network, and a heterodyne interference, to stabilize the local phase at each node. Pauli-correction gates, based on the state prepared, are issued from the server QNPU to its QDevice to obtain $|\Phi^+\rangle$: an X_π gate if the generated Bell-state is $|\Psi^+\rangle$ and an X_π gate followed by a Z_π gate (decomposed into X and Y gates) for $|\Psi^-\rangle$. As preparation for the experiment, we verified the entanglement generation, instructed by the QNPUs and using the same method as in Ref. [39], achieving a fidelity of 0.72(±0.02) for the $|\Phi^+\rangle$ state. The Bell corrections done through the server QNPU take up to 0.16 ms for $|\Psi^+\rangle$ and up to 0.49 ms for $|\Psi^-\rangle$. On the other hand, generating entanglement without the QNPU and with no Pauli correction, we achieve an average fidelity of 0.74(±0.03), with $\eta_C = 0.1$ and $\eta_S = 0.06$. The choice of different η values is due, in the first place, to speed up the rate of such a measurement. It shows, however, better performance with respect to the instructed version, which is due to the fact that the Pauli-correction instruction comes with a latency.

5.10. Delegated Quantum Computation (DQC) experiment on NV

5.10.1. Procedure

We execute the application in a tomography way to establish the quantum performance metrics (paperFigure 5.4b Figure 3b of the main text, where we use P_c to refer to the client program, and P_s to the server program): The client CNPU initiates P_c with fixed (α, θ). This results in a single CNPU process, a single QNPU process, and opening of an ER socket (see Section 5.8.3) with the server node. At the same time, the server CNPU initiates P_s resulting in a single CNPU process, a single QNPU process, and opening of an ER socket with the client node. The client and the server programs execute the subroutines in paperFigure 5.4c Figure 3c of the main text, looping 1200 times: both immediately start the second iteration once the first is completed. After the 1200th iteration, both client and server stop their respective CNPU and QNPU processes. We repeat 6 times for $(\alpha, \theta) \in \{\pi/2, \pi\} \times \{\pi/4, \pi/2, \pi\}$ for a total of 7200 executions of the circuit depicted in paperFigure 5.4a Figure 3a of the main text. We expect $|\psi\rangle$ to be either $|-Y\rangle$ (for $\alpha = \pi/2$) or $|-Z\rangle$ (for $\alpha = \pi$). To estimate the resulting $|\psi\rangle$ per (α, θ), the contents of S2 (containing the server qubit measurement) in the server loop was varied such that we obtained 600 measurement outcomes in basis $|+Y\rangle$ ($|+Z\rangle$) and 600 measurement outcomes in the corresponding orthogonal basis $|-Y\rangle$ ($|-Z\rangle$) for $\alpha = \pi/2$ (π).

Since our experiments are conducted on two NV nodes that are directly connected, we install a constant network schedule with time bins of 10 ms in which all time bins are assigned to networking. This allows us to assess the performance of executing quantum network applications without introducing a dependence on changing network schedules. This means the network process is made ready at the start of each such time bin, although may not instruct the QDevice to make entanglement if no requests for entanglement have been made.

5.10.2. Definitions

The result of a single DQC circuit execution (paperFigure 5.4a Figure 3a of the main text) is a single-qubit state ρ_{DQC} on the server. The success of running DQC can be expressed as the fidelity of ρ_{DQC} compared to the expected state (in case of no noise) $|\psi\rangle$ (paperFigure 5.4a Figure 3a of the main text). In the following we will call this fidelity the DQC fidelity, or F_{DQC}.

The value of F_{DQC} is affected the most by (1) the fidelity F_{EPR} of the entangled pair created between the client and server, and (2) the *qubit memory time* t_{mem}, which is the time that the server qubit must remain in memory (from entanglement success until measurement). The latter depends on the time at which the client sends a message to the server (paperFigure 5.4 Figure 3 of the main text). We refer to the two-qubit maximally entangled Bell states as $|\Phi^+\rangle = (|00\rangle + |11\rangle)$, and $|\Psi^\pm\rangle = (|01\rangle \pm |10\rangle)$, where $\Phi^+ = |\Phi^+\rangle\langle\Phi^+|$ and $\Psi^\pm = |\Psi^\pm\rangle\langle\Psi^\pm|$.

5.10.3. Post-Selection Based on Latency

In our experiments, the server qubit memory time t_{mem} has a significant variance across executions of the DQC circuit. In some iterations, there were huge spikes in latencies, which skew the results significantly. An upper bound t_{max} (see Section 5.10.4) was used to filter out results from iterations in which t_{mem} was larger than t_{max}. This resulted in filtering 146 out of 7200 data points. We note that for computing F_{DQC}, we applied the latency filter on top of the Single-Shot Readout (SSRO) and CR filters (see Methods). For the processing time analysis (below), however, we applied only the latency filter directly to all 7200 original data points.

5.10.4. SIMULATION

A simulation (using NetSquid [105]) of the DQC application was performed in order to estimate the expected F_{DQC} on our NV setup, and to establish a suitable value for t_{max} (used in latency post-selection).

We emphasize that this simulation is a heuristic to find t_{max}, and does not aim to predict the performance to full accuracy. All runs for which latencies were less than t_{max} were ultimately used to assess the performance from data, not using this simulation.

The simulation contains the following steps, where we used the model explained in Ref. [22]:

1. Start with a density matrix ρ_{EPR} describing the approximate state of the EPR pair just after entanglement success.

2. Apply operations representing the local gates on both the client and server, including the measurement on the client qubit. These operations are assumed to be perfect (no noise).

3. Apply depolarizing noise to the server qubit for a duration of t_{mem}, using the decoherence formula $e^{-(t_{\text{mem}}/T_{\text{coh}})^n}$ where T_{coh} was set to 13 ms and $n = 1.67$. These values are obtained via fitting experimental data from prior tests.

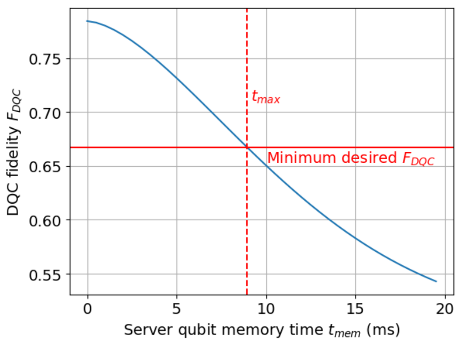
4. Calculate the fidelity between the final server qubit state and the expected state $|\psi\rangle$.

Based on the parameters of the setup when the DQC experiment was performed, ρ_{EPR} is set to

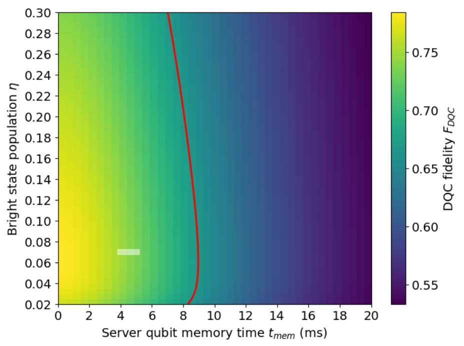
$$\begin{bmatrix} 0.049 & 0 & 0 & 0 \\ 0 & 0.437 & 0.284 & 0 \\ 0 & 0.284 & 0.454 & 0 \\ 0 & 0 & 0 & 0.061 \end{bmatrix}$$

which has fidelity 0.729 to the perfect Ψ^+ state. The setup can also produce Ψ^- states but for simplicity we use only the Ψ^+ case here.

The simulation computes an estimate of F_{DQC} for a given server qubit memory time t_{mem}. Since the desired minimum value for F_{DQC} was 0.667, the latency threshold t_{max} was set to 8.95 ms (Figure 5.15a).



(a) Expected values (based on simulation, Section 5.10.4) of DQC fidelity F_{DQC} for different duration values that the server qubit must remain in memory (t_{mem}). The maximum allowed qubit memory time $t_{\max}$ is chosen such that application iterations that are expected to result in too low F_{DQC} (< 0.667) are filtered out.



(b) Expected values (based on simulation) of DQC fidelity F_{DQC} for different values of the bright state population (η) in the single click protocol, and for different duration values that the server qubit must remain in memory (t_{mem}). The red line indicates the threshold of 0.667 for the target fidelity. The white box represents the experimentally obtained results (we fixed $\eta = 0.07$ and observed t_{mem} 4.8(8) ms, see paperFigure 5.4d Figure 3d of the main text).

Figure 5.15: Estimated fidelities based on simulation for executing Delegated Quantum Computation (DQC) on our NV setup.

5.10.5. Sweep of Qubit Memory Time and Bright State Population

As explained in Section 5.9.1, entanglement is created using the single-photon protocol using bright state population parameter η.[2] Using the simulation, we can estimate how F_{DQC} would change for different values of η and t_{mem}. Figure 5.15b shows the estimated F_{DQC} for different values of η and t_{mem}. It indicates that for the particular setup used, increasing η has little effect, while reducing qubit memory time does. For the DQC experiment $\eta = 0.07$ was used.

5.10.6. Processing Time and Latencies

Here we provide a detailed breakdown of the duration of execution phases of the DQC application, in order to gain insights into the processing times and latencies of the system for the different components.

Server Qubit Memory Time

paperFigure 5.4c Figure 3c of the main text shows the duration that the server qubit must remain in memory t_{mem} while waiting, averaged over all DQC circuit iterations that passed the latency filter. paperFigure 5.4d Figure 3d of the main text shows the breakdown of t_{mem} into individual segments of processing on both client and server. In Figure 5.16 we show the average duration and the standard deviation of each of these segments. The largest time is spent on preparing S2, which involves running Python code on the CNPU and converting this (using Python) into a NetQASM subroutine. Caching of the preparation of the NetQASM subroutine could significantly speed up this process.

[2]In most literature, the variable α is used for this parameter; here we use η to avoid confusion with the α parameter of the DQC application.

In the future, further improvements could include an optimized ahead-of-time compilation step. The large standard deviation is due to the fact that on the CNPU, other (background) processes run simultaneously with the DQC application process, and there is no precise control over the scheduling of these processes.

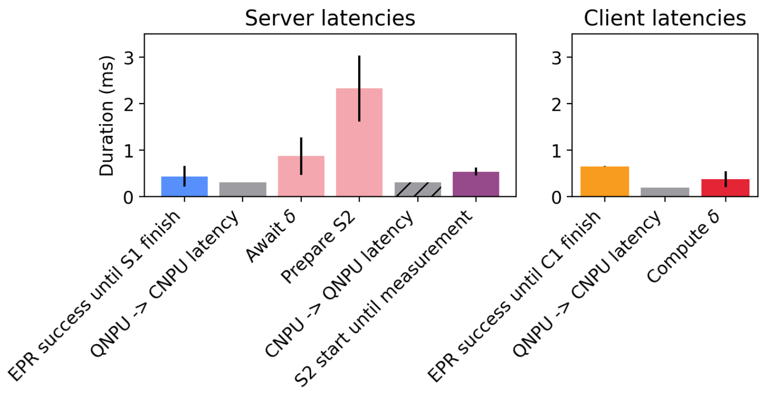


Figure 5.16: Average latency (duration) of each of the processes happening while the server qubit remains in memory in the DQC application. The QNPU to CNPU latency and CNPU to QNPU latency are estimated as explained in Section 5.10.6, and fixed to 0.305 ms (server) and 0.197 ms (client). The other latencies are the mean and standard deviation of the corresponding processes averaged over all DQC circuit iterations that passed the latency filter.

Tracing

The CNPU, QNPU, and QDevice all keep track of events happening in their system, by storing a tuple (t, e) where t is a timestamp and e the name of the event.

The QNPU timestamp granularity is 10 μs, since that is the duration of a single QNPU clock cycle. This clock cycle is synchronized with the clock of the QDevice, which in turn is synchronized with the QDevice of the other node (see paperSection 5.6 the Methods section in the main text and all paragraphs therein related to NV implementation). This results in the two QNPUs (of the two nodes in the experiment) having synchronized clocks with 10 μs precision. This means that the event indicating to the QNPUs that EPR generation has succeeded happens at the same clock cycle on both QNPUs.

The CNPU is not a real-time system (instead, it runs on a general purpose Linux OS) and records timestamps by consulting the system clock at μs precision. These timestamps are not synchronized to the QNPU timestamps. Furthermore, the CNPU timestamps obtained in this way are not as consistent as the real-time clock ticks on the QNPU. Therefore, the relative CNPU time compared to the QNPU time (on the same node) may fluctuate.

CNPU-QNPU Communication Latency

The latency of communication between the CNPU and QNPU can be calculated by looking at the time between CNPU events and QNPU events. However, since the CNPU

Derived latency (fit)	Description	Value (ms)
$\Delta_{cS1} - \Delta_{qS1}$	Send S1 + receive S1 result	0.384
$\Delta_{qS12} - \Delta_{cS12}$	Receive S1 result + Send S2	0.609
$\Delta_{cS2} - \Delta_{qS2}$	Send S2 + receive S2 result	0.467
$\Delta_{cC1} - \Delta_{qC1}$	Send C1 + receive C1 result	0.394

Table 5.4: Derived values for CNPU-QNPU communication latencies. The Δ variables are observed timestamp differences on the CNPU or QNPU, per execution of the DQC circuit, as shown in Figure 5.17. Subtracting pairs of variables from each other produces sums of two CNPU-QNPU communication latencies. These sums of latencies highly fluctuate per execution of the DQC circuit, due to the inaccuracy of the CNPU timestamps. However, the data fits a constant value, which is shown in the table and used in further analysis.

timestamps are fluctuating compared to the QNPU timestamps, we cannot use a direct comparison between CNPU and QNPU timestamps. Instead, we look at time differences on the CNPU and compare them to time differences on the QNPU, given that we know the order in which events occur during the DQC application execution. Figure 5.17 shows a schematic overview of events happening on the CNPU and the QNPU during a single execution of the DQC circuit. By comparing, e.g., (1) the time difference on the CNPU between sending subroutine S1 and receiving its result with (2) the time difference on the QNPU between receiving subroutine S1 and finishing it, we can estimate the total latency of sending S1 from CNPU to QNPU and receiving its result. Using this technique, we can estimate the latencies for each communication between CNPU and QNPU, as listed in Table 5.4. Again, since the CNPU timestamps fluctuate compared to the QNPU timestamps, the derived latencies fluctuate and can even be negative. However, for all derived latencies, we found that a constant function best fit the data. This verifies that the actual latency is constant as expected, and that the variance is due to the inaccuracy of CNPU timestamps.

Using the result from Table 5.4, we can compute bounds on the four individual latency variables of the server (we have a system of three linear equations, and we know that all latencies must be strictly non-negative):

- Sending S1 from CNPU to QNPU: < 0.242 ms.
- Receiving S1 result on CNPU from QNPU: between 0.142 and 0.384 ms.
- Sending S2 from CNPU to QNPU: between 0.225 and 0.467 ms.
- Receiving S2 result on CNPU from QNPU: < 0.242 ms.

In the latency breakdown of the server qubit memory time (see Section 5.10.6) we are only interested in the latencies that happen during the time that the server qubit is in memory. For the server these are the latencies for receiving the S1 result and sending S2. The sum of these two latencies is $\Delta_{qS12} - \Delta_{cS12} = 0.609$ ms (see Table 5.4). For simplicity, we say that both latencies constitute half of this time, as mentioned in the caption of Figure 5.16. Similarly, for the client we are only interested in the latency of receiving the C1 result. For simplicity we take this latency to be the same as that of sending C1, i.e. we use half of $\Delta_{cC1} - \Delta_{qC1}$.

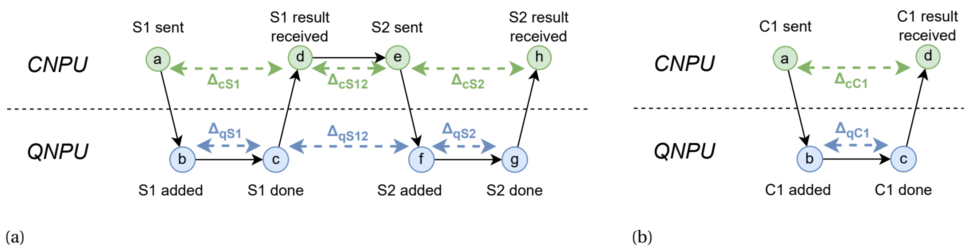


Figure 5.17: Schematic of events happening on the CNPU and QNPU during a single execution of DQC on the server (a) and the client (b). Time flows to the right. The Δ variables are the time differences between events, and are used to estimate CNPU-QNPU communication latencies ($a \to b$, $c \to d$, $e \to f$, $g \to h$ on the server and $a \to b$, $c \to d$ on the client).

Entanglement Generation

An overview of all values discussed in this section is given in Table 5.5.

EPR generation happens by attempting entanglement repeatedly until success. The QNPU sends an `ENT` physical instruction (Table 5.1) to the QDevice, which starts a batch of physical attempts. Each attempt takes 3.95 μs and a batch contains 500 attempts. If a batch fails (no success after 500 attempts), the QNPU sends another `ENT` instruction. Table 5.5 lists the average success probability per attempt and per batch that we found in the DQC experiments. As explained in Section 5.9.1, the NV QDevice creates either a Ψ^+ or a Ψ^- state. Table 5.5 shows statistics on how often each of these states was created during our experiments.

Figure 5.18 shows the distribution of time it takes to generate an EPR pair in the DQC experiment, where the average duration of such is 439 ms. This is the duration between starting the network process and finishing it, which includes entanglement attempts until success on the QDevice and subsequent Bell state corrections to Φ^+ (see Section 5.9.1). This duration corresponds to a fitted rate of 2.28(3) created EPR pairs per second. If only the QDevice entanglement generation is considered (i.e. without Bell state corrections and without QNPU processing overhead), this rate is 2.37(2) EPR pairs per second.

Local Gate Durations

As part of the DQC execution, the QNPU sends physical instructions to the NV QDevice for executing local quantum gates. In Table 5.6 we report on the observed durations of these gates from the perspective of the QNPU: these durations are from the time the physical instruction is sent to the QDevice until the corresponding result is received from the QDevice. We note that these durations are longer than these gates would take if they were executed directly on the QDevice (without QNodeOS, see Table 5.3) because of two reasons: (1) the limited granularity with which the QNPU and QDevice communicate (rounds of 10 μs) and (2) the fact the QDevice interleaves DD sequences in between sequences for the physical instruction itself, as explained in paperSection 5.6 the Methods section in the main text.

General Experiment Statistics

Table 5.7 lists statistics about the overall DQC experiment (all 7200 DQC circuit executions combined). We confirm our hypothesis that the overwhelming fraction of time

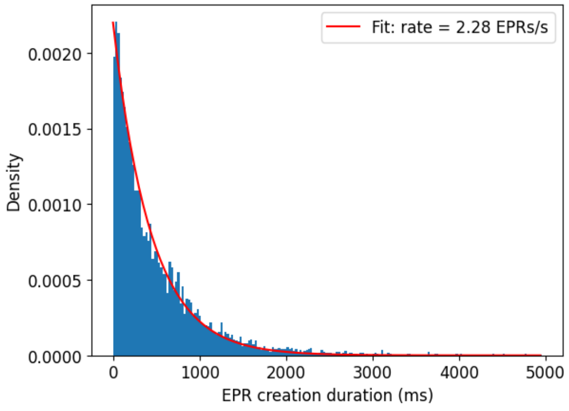


Figure 5.18: Histogram of EPR generation durations (time from first attempt until success) based on all EPR generations in the DQC experiment (using only latency-filtered data points, see Section 5.10.3). The histogram shows which fraction of all durations were in a particular duration window (window width: 25 ms). Expected EPR generation duration follows an exponential decay, with a rate parameter of 2.28(3) successes (EPR pairs) per second.

Parameter	Value
Duration of a single entanglement attempt*	3.95 μs
Number of attempts per batch*	500
Average number of failed batches until success	144
Average success probability per batch	6.95×10^{-3}
Average success probability per attempt	1.39×10^{-5}
Number of Psi+ states generation	3187 (44.3%)
Number of Psi- states generation	4013 (55.7%)
EPR generation rate (fit) (QDevice)	2.37(2) EPRs/s
EPR generation rate (fit) (QNodeOS)	2.28(3) EPRs/s
Average fraction of EPR generation time spent on sync failure	0.18

Table 5.5: Overview of values derived from the DQC experiment analysis, based on all 7200 DQC circuit executions. Entries with an asterisk (*) are values that we fixed in our experiments. The other values are observed experimental results. Average success probabilities are derived from the number of failed batches until success. EPR generation rate is distinguished between QDevice and QNodeOS. For the QDevice, it indicates the fitted (to an exponential decay function) time between the first `ENT` physical instruction and the first entanglement success (see Section 5.8.6). For QNodeOS, it indicates the fitted time between the start of the network process and the end of the network process (i.e. when entanglement has been created and Bell state corrections have been applied, see Section 5.9.1). Entanglement sync failures happen when one QDevice (server or client) wants to attempt entanglement but the other QDevice is not ready (Section 5.8.6). Such sync failures were observed intermittently during a batch of entanglement attempts.

Physical instruction	Duration (client)	Duration (server)
Measure	130 – 160 μs	80 – 100 μs
X90	80 – 100 μs	50 – 130 μs
X180	80 – 100 μs	10 – 130 μs
-X90	—	50 – 130 μs
Y90	70 – 200 μs	50 – 130 μs
Y90	—	50 – 130 μs

Table 5.6: Duration of executing local quantum gates on the NV QDevice in the DQC experiment. Durations are from sending the physical instruction from QNPU to QDevice until receiving the QDevice response. The -X90 and Y90 gates were never executed in the client DQC program.

is spent on the network process, namely generating EPR pairs. We also see that as expected, the server spends more time on user processes than the client does, since it does more local gates than the client (namely, the gates in subroutine S2).

Value	Client	Server
Total experiment duration	4243 s	4065 s
Time spent executing network process	3840 s	3825 s
Time spent executing user processes	5.041 s	7.618 s

Table 5.7: Overall durations of the DQC experiment.

5.10.7. QNPU NETWORK PROCESS ANALYSIS

In this section we focus on the execution of the network process in the QNPU as observed in the execution of DQC. The ER sockets (Section 5.8.3) are designed to facilitate the generation of entanglement belonging to a pair of user processes between two different QNPUs. In particular, the ER socket allows the QNPU to proceed with entanglement generation, while only one node may not have issued a request for entanglement yet.

During execution of the DQC application, the client QNPU has a single user process P_c for its DQC program and the server QNPU has a single user process P_s for its DQC program. Both user processes realize the repeated execution of subroutines that jointly realize the DQC circuit (paperFigure 5.4a Figure 3a of the main text).

In each single repetition of the DQC circuit, P_s executes first S1 and then S2, and P_c executes C1. P_s (in S1) and P_c (in C1) execute a NetQASM instruction for creating an entangled pair, which results in an entanglement request that is submitted to the network stack. Then, P_c and P_s go into the waiting state (see Section 5.8.3) until the entangled pair is delivered by the network process.

P_c executes a `create_epr` instruction and P_s executes a `recv_epr` instruction. Therefore, the client is seen as the *initiator* (see Section 5.8.3). P_s and P_c open a pair of ER sockets with each other when they start and keep it open for the whole experiment. P_c and P_s, being on different nodes, operate independently, and may hit their entanglement request instruction at different times. Since the client is the initiator and the server the receiver, the server is always willing to handle an entanglement request with the client. So, the network stack on both client and server will handle a request for entanglement as soon as the client submitted it to its network stack, regardless of whether the server already executed the corresponding `recv_epr` in S1.

We observe that in 3245 out of all 7200 DQC circuit executions, the client submitted the corresponding entanglement request to its network stack (in C1) *before* the server submitted its entanglement request to its own network stack (in S1), but where the server still complied by starting the network process and handling the request.

CLIENT WAITS FOR SERVER

From our architecture, we expect that it can happen that the client must wait for the server. This can be the case in the following scenario. The client executes C1 for DQC circuit iteration i and submits the entanglement request. Then, the next network time bin starts and the client QNPU starts the network process. However, the server is at this time (the beginning of the time bin) still busy with executing S2 for iteration $i-1$ (in user process P_s). Therefore the server QNPU cannot yet activate its own network process. Since the ER socket with the server is open and the client is the 'initiator', the client will send entanglement physical instructions to the QDevice anyway, but the QDevice will not be able to do actual attempts because the server QDevice is not ready (Section 5.8.6). Only when the server QNPU completes S2, it can activate the network process, which then sends entanglement physical instructions to the QDevice. Only at this point the QDevices can start actual entanglement generation. We observe that it did indeed happen that the client had to wait for the server, although we observed this behaviour in only in 60 out of 7200 DQC circuit executions.

SERVER WAITS FOR CLIENT

We expect that it can also happen that the server must wait for the client. This can be the case in the following scenario: The server executes S1 for DQC circuit iteration i and submits the entanglement request. Then, the next network time bin starts. However, the client did not yet hit the entanglement request in C1 for DQC iteration i, so there is nothing to do for the server network process. The server hence needs to wait for the next time bin, and check again if by now the client has submitted its entanglement request. We observe that in 1323 out of 7200 DQC circuit executions, the server had to wait for the client.

START OF NETWORK PROCESS

We examine the start of the network process in relation to the start of a time bin. In particular, the start of the network process may be delayed if there is still a user process running.

The network process is only activated at the beginning of a time bin. In our experiment, a time bin starts every 10 ms and lasts 10 ms. In most cases when the network process is activated, this activation happens very quickly after the time bin start (within 100 μs, as some QNPU software processing is needed). For the client QNPU, the network process never starts more than 100 μs after a time bin start. For the server, in 13 out of 7200 DQC circuit executions, the network process starts more than 100 μs after a time bin starts, since in these cases there was still a user process running. In Table 5.8, an overview of all network process statistics is given.

Parameter	**Value**
Number of times server puts EPR request to network stack before client	1774/7200
Number of times server starts entanglement before putting in EPR request	3245/7200
Number of times submitted EPR request is handled in immediate next time bin	5523/7200
Average number of bins that pass before request is handled	2.33
Number of times server needs to wait for client	1323/7200
Number of times client needs to wait for server	60/7200
Number of times client network process starts > 100 μs after time bin starts	0
Number of times server network process starts > 100 μs after time bin starts	13

Table 5.8: Statistics on the QNPU network process behavior during the whole DQC experiment, i.e. totalled over all 7200 DQC circuit iterations.

5.11. MULTITASKING EXPERIMENTS ON NV

The multitasking evaluation was done in two parts:

- **Quantum tomography while multitasking**: Executing a single DQC application (on client and server) and a single Local Gate Tomography (LGT) application (on client only) where it was verified that the LGT application produces expected quantum results (see Section 5.11.2).
- **Scaling the number of applications**: Executing N DQC applications and N LGT applications, where the classical device utilization metric was compared with a version of QNodeOS without multitasking, and where we investigated the behavior of the QNPU scheduler on the client in the context of multiple programs (see Section 5.11.3).

The network schedule was set as in the previous DQC experiment for direct comparison.

5.11.1. MOCKED ENTANGLEMENT

For the multitasking evaluation, we focused on the behavior of QNodeOS, and opted not to use the standard entanglement generation procedure in our NV QDevices as done in the DQC experiments (Section 5.10) to allow for a simpler experiment. Instead, we used a mocked entanglement generation process on the QDevices (executing entanglement actions without entanglement): weak-coherent pulses on resonance with the NV transitions, that follow the regular optical path, are employed to trigger the CPLD in the entanglement heralding time-window.

We stress that in our multitasking experiments, the exact same physical instructions are sent to the QDevice as would be done when using real entanglement, and the exact same responses are sent back. Therefore, QNodeOS needed to perform the same operations (including scheduling decisions) as it would have needed to do with real entanglement. Furthermore, we aimed to keep the rate of entanglement 'success' in the QDevices the same order of magnitude as that of the DQC experiments (10.14 EPRs/s compared to 2.37 EPRs/s in the DQC experiment) by keeping the mean-photon number of the weak-coherent pulse comparable to p_C and p_S (in the order of $\sim 10^{-4}$).

5.11.2. TOMOGRAPHY RESULTS

We perform tomography when not multitasking, in order to verify our expectation that multitasking should not affect the quantum performance of LGT: The tomography results of the LGT application in the multitasking scenario are given in paperFigure 5.5c Figure 4c of the main text. We also ran the same LGT application on the client in a non-multitasking scenario. In this case, the client ran the LGT application and there was no DQC application run at all (the server did nothing). The tomography results of LGT for the non-mulitasking scenario are given in Figure 5.19. The results are slightly different since the multitasking experiment was done on a different day than the non-multitasking experiment. However, within error bars we verify that multitasking does not affect the quantum performance of the LGT application.

5.11.3. SCALING TO MORE THAN TWO APPLICATIONS

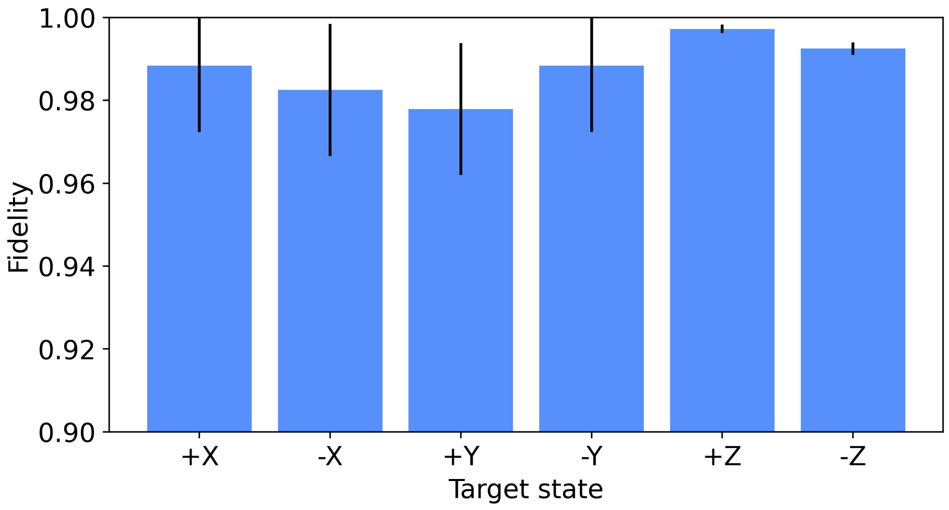


Figure 5.19: Local Gate Tomography results on the client node in a non-multitasking scenario.

QNPU PROCESSES AND STEPS

For the scaling evaluation, we did an experiment for each $N \in \{1,2,3,4,5\}$. For each experiment, the client CNPU started N DQC-client programs and N LGT programs concurrently, and the server CNPU started N DQC-server programs. In this section we discuss the observed behavior of the client and server QNPUs during these experiments. The client QNPU has $2N$ user processes (N DQC user processes and N LGT user processes), each of which continuously receives quantum blocks in the form of NetQASM subroutines (C1 for DQC processes and L1 for LGT processes). These $2N$ user processes and the single client network process are scheduled by the client QNPU scheduler. The server has N user processes (all for DQC) which are scheduled together with the server network process by the server QNPU scheduler. Figure 5.20 shows a schematic diagram of the nominal (most often occurring) pattern of scheduling.

In both S1 and C1, there is a single `create_epr` NetQASM instruction [43] for creating entanglement with the other node, followed by a `wait_all` NetQASM instruction that waits until the request entangled qubit is delivered. The `create_epr` instruction is handled by the QNPU processor by sending the entanglement request to the network stack. Upon executing the `wait_all` instruction, the user process executing this subroutine (S1 or C1) goes into the waiting state (green stop sign in Figure 5.20). When the network process completes (having created the entangled qubit), the user process can be resumed, finishing the subroutine (C1 or S1).

On the server QNPU, for each DQC user process U the following sequence is repeated:

- U is in the *idle* state;
- NetQASM subroutine S1 is submitted by the CNPU to the QNPU, moving U to *ready*;

- *U* is activated; S1 is executed until it hits the `wait_all` instruction; *U* goes into the *waiting* state;
- The network process handles the entanglement request for S1 until EPR creation succeeds; *U* goes into *ready* again;
- *U* is activated; S1 is executed until completion; *U* goes to *idle*;
- NetQASM subroutine S2 is submitted by the CNPU; *U* goes to *ready*;
- *U* is activated; S2 is executed until completion; *U* goes to *idle*.

The above sequence is for one execution of the DQC circuit (paperFigure 5.4a Figure 3a of the main text), and is hence repeated many times.

On the client QNPU, for each DQC user process *U* the following sequence is repeated:

- *U* is in the *idle* state;
- NetQASM subroutine C1 is submitted by the CNPU, moving *U* to *ready*;
- *U* is activated; C1 is executed until it hits the `wait_all` instruction; *U* goes into the *waiting* state;
- the network process handles the entanglement request for C1 until EPR creation succeeds; *U* goes into *ready* again;
- *U* is activated; C1 is executed until completion; *U* goes to *idle*.

The above sequence is for one execution of the DQC circuit (paperFigure 5.4a Figure 3a of the main text), and is hence repeated many times.

On the client QNPU, for each LGT user process *U* the following sequence is repeated:

- *U* is in the *idle* state;
- NetQASM subroutine L1 is submitted by the CNPU, moving *U* to *ready*;
- *U* is activated; L1 is executed until completion; *U* goes to *idle*.

The above sequence is for one execution of the LGT circuit (paperFigure 5.5a Figure 4a of the main text), and is hence repeated many times.

For the above sequences for user processes, only the internal order is fixed; the time in between steps depends on the QNPU scheduler, as well as the time at which the CNPU submits subroutines. Furthermore, since there are multiple user processes at the same time (for the server, only for $N > 1$), the above steps happen for each user process U_i and the steps are interleaved. Figures 5.20 to 5.22 show examples of how these user processes can be interleaved on both client and server QNPU.

DQC AND LGT INTERLEAVING

We investigate the degree of interleaving the execution of DQC and LGT, in particular how many LGT subroutines are executed when a DQC process is waiting: The client QNPU executes both DQC and LGT user processes. DQC user processes are often in the waiting state. This happens when their C1 subroutine is suspended, waiting for the network process to handle their entanglement request. The network process is only activated at the beginning of a time bin, which happens only every 10 ms, or when a user process finishes executing a subroutine, the latter not occurring very frequently for low number of programs N. Furthermore, DQC user processes can be in the idle state, namely when they completed execution of C1 for some iteration i of the DQC circuit, but are still waiting for the CNPU to send C1 for iteration $i+1$. In both these types of 'gaps' (waiting or idle), LGT subroutines can be executed (each taking ≈2.4 ms). Table 5.9 lists the maximum number of consecutive LGT subroutines that were executed in between DQC subroutines for both types of gaps.

SUBROUTINE (QUANTUM BLOCK) EXECUTION ORDER

We investigate whether the QNPU schedules quantum subroutines in a different order than they arrived from the CNPU. As expected, we find that this is the case. Although the QNPU handles subroutines from the CNPU first-come-first-served, some of these subroutines (in our experiments, precisely the DQC subroutines that wait for entanglement) are put into the waiting state. This allows the QNPU to schedule other subroutines (in our experiments, we observe LGT subroutines being executed), even if they arrived later from the CNPU than the waiting DQC subroutine. Schematic overviews of such scheduling that we observed are depicted in Section 5.12.

USER PROCESS IDLE TIMES

We examine the number of times, and the duration, that a user process is idle waiting for submission of a subroutine from the CNPU as a function of N: a user process is *idle* when there are currently no subroutines associated with the process pending to be executed. This means that the QNPU waits, at least for this user process, until the CNPU sends the next subroutine for the user process. Table 5.9 lists the number of times and durations of moments at which all client QNPU user processes are idle. This number and their durations decrease for larger values of N. This is expected since there are more active processes, and hence more subroutines being sent from the CNPU for different processes. In most cases, when finishing a subroutine for user process U, there is then another user process U' already waiting with another subroutine to execute.

NETWORK PROCESS START DELAYS

We examine the scheduling behaviour of the network process in relation to user processes. We expect that due to the fact we use a non-preemptive scheduler, a network process may not be activated at the start of a network time bin, due to a user process still being executed. We investigate the occurrence of such events in our multitasking experiment, including the delay with which the network process is started in such a scenario (see Table 5.9): When a user process submits an entanglement request to the network stack, this request is handled at the earliest when the network process is activated. This happens either at the start of the next network time bin, or when a user process finishes a subroutine. Therefore, there is often some time in between submitting the request and the network process handling it. This waiting time is in most cases bounded by 10 ms, since that is the length of a time bin, and all time bins are assigned to networking in our experiment. However, in some cases the client may still be executing a LGT subroutine when a new time bin starts, delaying the start of the network process until this subroutine has finished. We expect however that in all cases, as soon as such an LGT subroutine finishes, the QNPU scheduler then immediately schedules the network process, and not another LGT subroutine. We found that the maximum difference between time bin start and network process start is 2.59 ms, which verifies that indeed at most one LGT subroutine is sometimes executed during a time bin start (LGT subroutine execution duration being ≈2.4 ms.)

We remark that with increasing N, the network process is delayed more frequently by a LGT subroutine. This is expected due to the fact that more subroutines from different user processes await execution. Consequently, with increasing N it also happens more frequently that the client and server do not start execution of the network process in the same time bin (see below).

Client Waits for Server and Vice Versa

In order to better understand the concurrent execution of multiple applications (here DQC and LGT) and corresponding programs, we investigate scenarios and times in which the client waits for the server (or vice versa).

The client and server open an ER socket at the beginning of each DQC application. So, during runtime, there are N ER sockets opened on the server QNPU (one for each DQC process) and N ER sockets opened on the client QNPU (one for each DQC process). In each DQC application, the client QNPU user process for that DQC application is the 'initiator' (see Section 5.8.3). This means that as soon as the client user process submits a request for entanglement (from within C1), both server and client QNPU start their network process to handle it (at the start of the next time bin, and provided the network process should not first handle a request from a user process from another DQC application).

It can happen that the client QNPU and server QNPU do not start their network process at the same time bin. This mostly happens when one of the nodes is still busy executing a user process subroutine when a time bin starts, as explained above. If this happens, the QNPU that did already start their network process sends entanglement instructions to their QDevice, but this will not result in physical entanglement attempts since the other QDevice is not available (leading to a entanglement sync failure, see Section 5.8.6). Table 5.9 lists the number of times that this happened.

For each of the N DQC applications that are running on client and server, and for each execution of the DQC circuit in those applications, there is a single entanglement request from the client (in C1) and a single entanglement request from the server (in S1). For each of these request pairs, the client at some point starts the network process and handles this request, and the server at some point starts the network process and handles its corresponding request. For each such pair of requests, the following scenarios can happen:

1. Client and server QNPU start their network process in the same time bin (one of

them may start a bit later than the start of the time bin because it needs to complete a quantum subroutine).

2. The client starts its network process in time bin k but the server starts it at some time bin $> k$. This happens when the server still has a qubit in memory when time bin k starts. Therefore, the server cannot activate its network process yet. A qubit still being in memory happens when the server QNPU has executed S1 for some DQC process (which produced an entangled qubit) but has not yet executed S2 (in which the qubit is measured and hence freed).

3. The server starts its network process in time bin k but the client starts it at time bin $k+1$. This happens (although rarely) when the client user process puts the entanglement request to the network stack just before the start of k. The server will immediately start attempts at k, but the client itself is still processing and 'misses' k; the client then only starts at time bin $k+1$.

Table 5.9 lists how often the above scenarios happen for each N.

5.12. MULTITASKING SCHEDULING PATTERNS

Parameter	**N = 1**	**N = 2**	**N = 3**	**N = 4**	**N = 5**
average no. LGT subroutines in between any DQC subroutines	0.83	1.42	1.59	1.65	1.65
max no. consecutive LGT subroutines in between DQC subroutines	3	4	6	7	8
max no. consecutive LGT subroutines when a DQC is in waiting state	2	3	4	4	4
% of times that ≥ 1 LGT subroutines fills time waiting for time bin	56	81	99	99	100
no. times that network process is delayed by a LGT subroutine	88/360	212/720	554/1080	940/1440	1170/1800
no. time windows in which all user processes are idle	399	56	4	1	0
Average length of idle time window (ms)	10	5.9	5.9	8.3	—
Maximum length of idle time window (ms)	152	31	15	8.3	—
% client and server start network process at same time bin	95.8	58.2	42.1	42.4	38.5
% server started network process 1 time bin after client	0.8	37.9	50.1	50.9	53.1
% server started network process > 1 time bins after client	0.0	3.3	7.7	6.6	8.4
% client started network process 1 time bin after server	3.3	0.6	0.1	0.1	0.0

Table 5.9: Overview of values derived from the multitasking experiments in which N DQC applications (on client and server) and N LGT applications (client only) were executed concurrently, for $N \in \{1,2,3,4,5\}$.

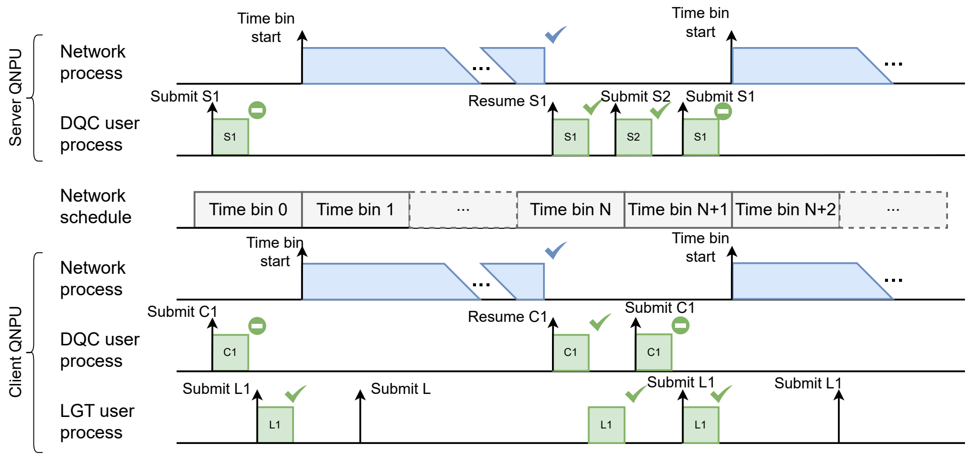


Figure 5.20: Nominal scheduling pattern on the client and server QNPUs when multitasking 1 DQC application (on client and server) and 1 LGT application (on client only). Pictured is a slice of time (moving to the right) in which a whole DQC circuit execution is realized, and 3 LGT circuit executions. Up-arrows indicate that the process becomes *ready* (either since a subroutine was submitted from the CNPU, or because a requested entangled qubit becomes available). Green blocks are NetQASM subroutines. Blue blocks are entanglement generation. Ticks indicate completion of a subroutine (user process) or entanglement request (network process). Stop sign means the user process goes into the *waiting* state. Time not to scale. Time bin length is 10 ms. Duration of L1 is ≈2.4 ms. Duration of entanglement generation is non-deterministic. On the server QNPU the following happens. S1 arrives from CNPU; DQC user process becomes *ready*. DQC user process is activated and executes S1. The entanglement instruction inside S1 is reached; entanglement request is sent to network stack; DQC user process becomes *waiting*. When time bin 1 starts, network process becomes *ready*. There is a pending entanglement request, so network process is activated; QDevice attempts entanglement until success (after non-deterministic number of time bins, blue tick). Requested entangled qubit is available: DQC user process becomes *ready* again; is activated; executes S1 until completion; becomes *idle*. QNPU receives subroutine S2 from CNPU; activates DQC user process; executes S2 until completion. At this point, the QNPU completed execution of the current repetition of the DQC circuit. QNPU then receives again a subroutine S1 (for the next DQC circuit iteration), and the same pattern repeats. On the client QNPU the following happens. C1 arrives from CNPU; DQC user process becomes *ready*. DQC user process is activated and executes C1. The entanglement instruction inside C1 is reached; entanglement request is sent to network stack; DQC user process becomes *waiting*. L1 arrives from CNPU; LGT user process becomes *ready*. LGT user process is activated; fully executes L1. When time bin 1 starts, network process becomes *ready*. There is a pending entanglement request, so network process is activated; QDevice attempts entanglement until success (blue tick). While network process is active, another L1 block arrives from CNPU (for next LGT circuit iteration) so LGT user process becomes *ready*. LGT user process is not activated since network process is still running. Upon entanglement success, the requested qubit is available; DQC user process is activated to complete C1. QNPU has now completed execution of the current repetition of the DQC circuit. LGT user process is activated to execute L1 which was still pending. The same pattern repeats.

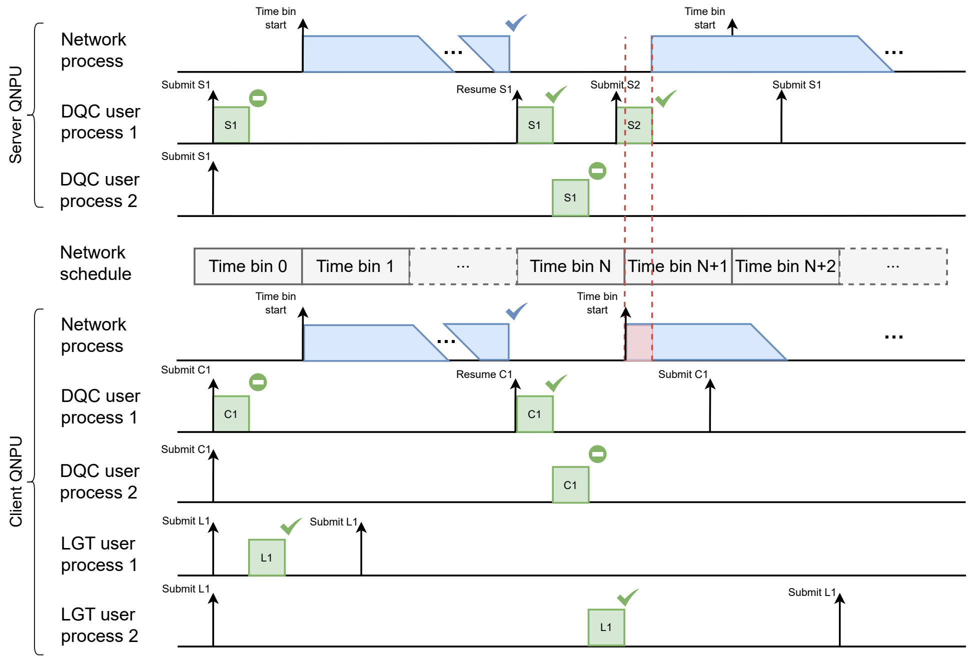


Figure 5.21: Example scheduling pattern of scenario with 2 DQC applications and 2 LGT applications (the symbol and color coding is the same as in Figure 5.20). In this case, the client needs to wait (red shaded area) for the server to finish S2 of DQC user process 1, before they can do entanglement generation for DQC user process 2. Scenario: 2 DQC applications (A1 and A2) are concurrently executed (A1: DQC-server program executed by server DQC user process 1 and DQC-client program executed by client DQC user process 1; A2: DQC-server program executed by server DQC user process 2 and DQC-client program executed by client DQC user process 2). Client and server successfully create entanglement for some DQC circuit execution i for A1 (just before time bin N starts). Client finishes C1 for user process 1, and meanwhile the server finishes S1 for user process 1. The client has completed its part of DQC circuit execution i for A1, but the server still needs to wait for S2 from the CNPU. Then, the client executes C1 for user process 2, which is the start of circuit execution j for A2; user process 2 becomes waiting. Meanwhile the server executes S1 for user process 2 which becomes waiting. The client needs to wait until the start of the next time bin $(N+1)$ until it can activate the network process to handle the request. In the meantime, it can execute an L1 block. Time bin $N+1$ starts and the client handles the request. However, the server has received S2 for execution i of A1, and starts executing it just before the time bin starts. Only after finishing it, the server can start the network process, which picks up the request for A2. While S2 is executing, the client QDevice tries to do entanglement attempts, but gets entanglement sync failures (Section 5.8.6) since the server QDevice is busy with S2.

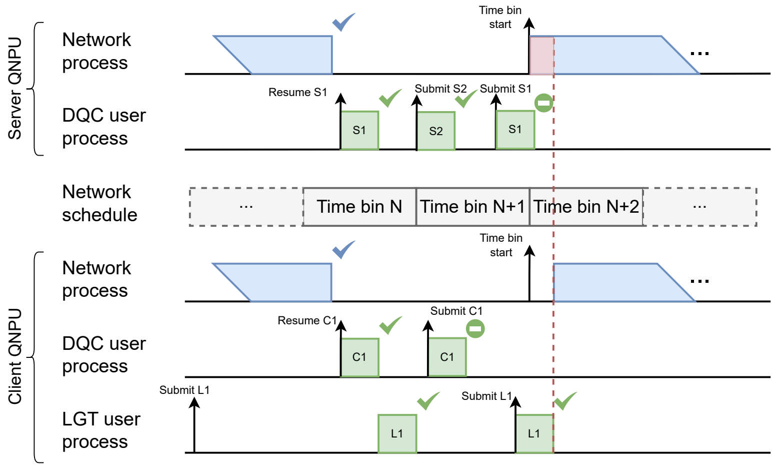


Figure 5.22: Example scheduling pattern of multitasking one DQC application (on client and server) and one LGT application (on client only), where the server must wait for client to finish its LGT user process (red area); the symbol and color coding is the same as in Figure 5.20. At the start of time bin $N + 1$, the server activates the network process to handle the request that was put by the previous S1 execution. However the client only starts some time later during the time bin, since it first needs to finish executing L1 for the LGT user process.

REFERENCES

[1] H. J. Kimble. "The Quantum Internet". In: *Nature* 453.7198 (2008), pp. 1023–1030. DOI: 10.1038/nature07127.

[2] S. Wehner, D. Elkouss, and R. Hanson. "Quantum Internet: A Vision for the Road Ahead". In: *Science* 362.6412 (2018), pp. 1–9. DOI: 10.1126/science.aam9288.

[3] R. van Meter. *Quantum Networking*. John Wiley and Sons, Ltd, 2014. DOI: 10.1002/9781118648919.

[4] S. Barz, E. Kashefi, A. Broadbent, J. F. Fitzsimons, A. Zeilinger, and P. Walther. "Demonstration of Blind Quantum Computing". In: *Science* 335.6066 (2012), pp. 303–308. DOI: 10.1126/science.1214707.

[5] P. Drmota, D. Nadlinger, D. Main, B. Nichol, E. Ainley, D. Leichtle, A. Mantri, E. Kashefi, R. Srinivas, G. Araneda, et al. "Verifiable blind quantum computing with trapped ions and single photons". In: *Physical Review Letters* 132.15 (2024). Publisher: APS, p. 150604. DOI: 10.1103/PhysRevLett.132.150604.

[6] D. Nadlinger. "Device-independent key distribution between trapped-ion quantum network nodes". PhD thesis. University of Oxford, 2022.

[7] S. Hermans, M. Pompili, H. Beukers, S. Baier, J. Borregaard, and R. Hanson. "Qubit teleportation between non-neighbouring nodes in a quantum network". In: *Nature* 605.7911 (2022), pp. 663–668. DOI: 10.1038/s41586-022-04697-y.

[8] M. Iuliano, M.-C. Slater, A. J. Stolk, M. J. Weaver, T. Chakraborty, E. Loukiantchenko, G. C. d. Amaral, N. Alfasi, M. O. Sholkina, W. Tittel, et al. "Qubit teleportation between a memory-compatible photonic time-bin qubit and a solid-state quantum network node". In: *arXiv preprint arXiv:2403.18581* (2024). DOI: 10.48550/arXiv.2403.18581.

[9] D. Matsukevich, P. Maunz, D. Hayes, L.-M. Duan, and C. Monroe. "Quantum teleportation between distant matter qubits". In: *Science* 323.5913 (2009). Publisher: American Association for the Advancement of Science, pp. 486–489. DOI: 10.1126/science.1167209.

[10] S. Langenfeld, S. Welte, L. Hartung, S. Daiss, P. Thomas, O. Morin, E. Distante, and G. Rempe. "Quantum Teleportation between Remote Qubit Memories with Only a Single Photon as a Resource". In: *Physical Review Letters* 126.13 (Mar. 2021), p. 130502. DOI: 10.1103/PhysRevLett.126.130502.

[11] W. Pfaff et al. "Unconditional quantum teleportation between distant solid-state quantum bits". In: *Science* 345.6196 (Aug. 2014), pp. 532–535. DOI: 10.1126/science.1253512.

[12] K. S. Chou, J. Z. Blumoff, C. S. Wang, P. C. Reinhold, C. J. Axline, Y. Y. Gao, L. Frunzio, M. H. Devoret, L. Jiang, and R. J. Schoelkopf. "Deterministic teleportation of a quantum gate between two logical qubits". In: *Nature* 561.7723 (Sept. 2018). Publisher: Nature Publishing Group, pp. 368–373. DOI: 10.1038/s41586-018-0470-y.

[13] A. Broadbent, J. Fitzsimons, and E. Kashefi. "Universal Blind Quantum Computation". In: *FOCS*. IEEE, 2009, pp. 517–526. DOI: 10.1109/FOCS.2009.36.

[14] M. W. Doherty, N. B. Manson, P. Delaney, F. Jelezko, J. Wrachtrup, and L. C. Hollenberg. “The nitrogen-vacancy colour centre in diamond”. In: *Physics Reports* 528 (2013). DOI: 10.1016/j.physrep.2013.02.001.

[15] L. Childress and R. Hanson. “Diamond NV centers for quantum computing and quantum networks”. In: *MRS bulletin* 38.2 (2013), pp. 134–138. DOI: 10.1557/mrs.2013.20.

[16] D. Fioretto. “Towards a flexible source for indistinguishable photons based on trapped ions and cavities”. PhD thesis. University of Innsbruck, 2020.

[17] D. L. Moehring, P. Maunz, S. Olmschenk, K. C. Younge, D. N. Matsukevich, L.-M. Duan, and C. Monroe. “Entanglement of Single-Atom Quantum Bits at a Distance”. In: *Nature* 449 (2007), pp. 68–71. DOI: 10.1038/nature06118.

[18] S. Ritter, C. Nölleke, C. Hahn, A. Reiserer, A. Neuzner, M. Uphoff, M. Mücke, E. Figueroa, J. Bochmann, and G. Rempe. “An Elementary Quantum Network of Single Atoms in Optical Cavities”. In: *Nature* 484.7393 (2012), pp. 195–200. DOI: 10.1038/nature11023.

[19] J. Hofmann, M. Krug, N. Ortegel, L. Gérard, M. Weber, W. Rosenfeld, and H. Weinfurter. “Heralded Entanglement Between Widely Separated Atoms”. In: *Science* 337.6090 (2012), pp. 72–75. DOI: 10.1126/science.1221856.

[20] R. Stockill, M. J. Stanley, L. Huthmacher, E. Clarke, M. Hugues, A. J. Miller, C. Matthiesen, C. Le Gall, and M. Atatüre. “Phase-Tuned Entangled State Generation between Distant Spin Qubits”. In: *Phys. Rev. Lett.* 119.1 (2017), p. 010503. DOI: 10.1103/PhysRevLett.119.010503.

[21] L. J. Stephenson, D. P. Nadlinger, B. C. Nichol, S. An, P. Drmota, T. G. Ballance, K. Thirumalai, J. F. Goodwin, D. M. Lucas, and C. J. Ballance. “High-Rate, High-Fidelity Entanglement of Qubits Across an Elementary Quantum Network”. In: *Phys. Rev. Lett.* 124.11 (2020), p. 110501. DOI: 10.1103/PhysRevLett.124.110501.

[22] M. Pompili et al. “Realization of a Multinode Quantum Network of Remote Solid-State Qubits”. In: *Science* 372.6539 (2021), pp. 259–264. DOI: 10.1126/science.abg1919.

[23] V. Krutyanskiy et al. “Entanglement of Trapped-Ion Qubits Separated by 230 Meters”. In: *Physical Review Letters* 130.5 (Feb. 2023), p. 050803. DOI: 10.1103/PhysRevLett.130.050803.

[24] J.-L. Liu, X.-Y. Luo, Y. Yu, C.-Y. Wang, B. Wang, Y. Hu, J. Li, M.-Y. Zheng, B. Yao, Z. Yan, et al. “Creation of memory–memory entanglement in a metropolitan quantum network”. In: *Nature* 629.8012 (2024), pp. 579–585. DOI: 10.1038/s41586-024-07308-0.

[25] A. J. Stolk, K. L. van der Enden, M.-C. Slater, I. t. Raa-Derckx, P. Botma, J. van Rantwijk, B. Biemond, R. A. Hagen, R. W. Herfst, W. D. Koek, et al. “Metropolitan-scale heralded entanglement of solid-state qubits”. In: *arXiv preprint arXiv:2404.03723* (2024). DOI: 10.48550/arXiv.2404.03723.

[26] C. Knaut, A. Suleymanzade, Y.-C. Wei, D. Assumpcao, P.-J. Stas, Y. Huan, B. Machielse, E. Knall, M. Sutula, G. Baranes, et al. “Entanglement of nanophotonic quantum memory nodes in a telecom network”. In: *Nature* 629.8012 (2024), pp. 573–578. DOI: 10.1038/s41586-024-07252-z.

[27] M. Ben-Or and A. Hassidim. “Fast Quantum Byzantine Agreement”. In: *STOC*. ACM, 2005, pp. 481–485. DOI: 10.1145/1060590.1060662.

[28] A. Poremba. “Quantum proofs of deletion for learning with errors”. In: *arXiv preprint arXiv:2203.01610* (2022). DOI: 10.48550/arXiv.2203.01610.

[29] P. A. Guérin, A. Feix, M. Araújo, and Č. Brukner. “Exponential communication complexity advantage from quantum superposition of the direction of communication”. In: *Physical review letters* 117.10 (2016). Publisher: APS, p. 100502. DOI: 10.1103/PhysRevLett.117.100502.

[30] A. M. Childs. “Secure Assisted Quantum Computation”. In: *Quantum Inf. Comput.* 5.6 (2005), pp. 456–466. DOI: 10.26421/QIC5.6-4.

[31] W.-Z. Liu, Y.-Z. Zhang, Y.-Z. Zhen, M.-H. Li, Y. Liu, J. Fan, F. Xu, Q. Zhang, and J.-W. Pan. “Toward a Photonic Demonstration of Device-Independent Quantum Key Distribution”. In: *Phys. Rev. Lett.* 129.5 (2022), p. 050502. DOI: 10.1103/PhysRevLett.129.050502.

[32] W. Zhang, T. van Leent, K. Redeker, R. Garthoff, R. Schwonnek, F. Fertig, S. Eppelt, W. Rosenfeld, V. Scarani, C. C.-W. Lim, et al. “A Device-Independent Quantum Key Distribution System for Distant Users”. In: *Nature* 607.7920 (2022), pp. 687–691. DOI: 10.1038/s41586-022-04891-y.

[33] B. Jing, X.-J. Wang, Y. Yu, P.-F. Sun, Y. Jiang, S.-J. Yang, W.-H. Jiang, X.-Y. Luo, J. Zhang, X. Jiang, et al. “Entanglement of three quantum memories via interference of three single photons”. In: *Nature Photonics* 13.3 (2019), pp. 210–213. DOI: 10.1038/s41566-018-0342-x.

[34] T. Van Leent et al. “Entangling single atoms over 33 km telecom fibre”. In: *Nature* 607.7917 (July 2022), pp. 69–73. DOI: 10.1038/s41586-022-04764-4.

[35] A. Dahlberg et al. “A Link Layer Protocol for Quantum Networks”. In: *SIGCOMM*. ACM, 2019, pp. 159–173. DOI: 10.1145/3341302.3342070.

[36] W. Kozlowski, A. Dahlberg, and S. Wehner. “Designing a Quantum Network Protocol”. In: *CoNEXT*. ACM, 2020, pp. 1–16. DOI: 10.1145/3386367.3431293.

[37] *Aliro Security, Advanced Secure Networking*. URL: https://www.aliroquantum.com (visited on 10/30/2024).

[38] A. Pirker and W. Dür. “A Quantum Network Stack and Protocols for Reliable Entanglement-Based Networks”. In: *New Journal of Physics* 21.3 (2019), p. 033003. DOI: 10.1088/1367-2630/ab05f7.

[39] M. Pompili et al. “Experimental Demonstration of Entanglement Delivery Using a Quantum Network Stack”. In: *npj Quantum Information* 8.1 (2022), p. 121. DOI: 10.1038/s41534-022-00631-2.

[40] I. Monga, E. Saglamyurek, E. Kissel, H. Haffner, and W. Wu. “QUANT-NET: A testbed for quantum networking research over deployed fiber”. In: *Proceedings of the 1st Workshop on Quantum Networks and Distributed Quantum Computing*. QuNet ’23. New York, NY, USA: Association for Computing Machinery, Sept. 10, 2023, pp. 31–37. DOI: 10.1145/3610251.3610561.

[41] M. Castells. “The Impact of the Internet on Society: A Global Perspective”. In: *Ch@nge: 19 Key Essays on How the Internet Is Changing Our Lives*. Madrid: BBVA, 2013.

[42] *Quantum Software Development Kits in 2024*. AIMultiple: High Tech Use Cases & Tools to Grow Your Business. 2024. URL: https://research.aimultiple.com/quantum-sdk/ (visited on 07/11/2024).

[43] A. Dahlberg, B. van der Vecht, C. Delle Donne, M. Skrzypczyk, I. te Raa, W. Kozlowski, and S. Wehner. “NetQASM—A Low-Level Instruction Set Architecture for Hybrid Quantum–Classical Programs in a Quantum Internet”. In: *Quantum Science and Technology* 7.3 (2022), p. 035023. DOI: 10.1088/2058-9565/ac753f.

[44] M. Caleffi, M. Amoretti, D. Ferrari, D. Cuomo, J. Illiano, A. Manzalini, and A. S. Cacciapuoti. “Distributed quantum computing: a survey”. In: *arXiv preprint arXiv:2212.10609* (2022). DOI: 10.48550/arXiv.2212.10609.

[45] M. Teller, V. Messerer, K. Schüppert, Y. Zou, D. A. Fioretto, M. Galli, P. C. Holz, J. Reichel, and T. E. Northup. “Integrating a fiber cavity into a wheel trap for strong ion–cavity coupling”. In: *AVS Quantum Science* 5.1 (2023). DOI: 10.1116/5.0121534.

[46] Y. Ma, E. Kashefi, M. Arapinis, K. Chakraborty, and M. Kaplan. “QEnclave-A practical solution for secure quantum cloud computing”. In: *npj Quantum Information* 8.1 (2022). Publisher: Nature Publishing Group UK London, p. 128. DOI: 10.1038/s41534-022-00612-5.

[47] D. A. Lidar and T. A. Brun. *Quantum Error Correction*. Cambridge University Press, 2013.

[48] L. Botelho, A. Glos, A. Kundu, J. A. Miszczak, Ö. Salehi, and Z. Zimborás. “Error mitigation for variational quantum algorithms through mid-circuit measurements”. In: *Physical Review A* 105.2 (2022). Publisher: APS, p. 022441. DOI: 10.1103/PhysRevA.105.022441.

[49] K. Bharti et al. “Noisy intermediate-scale quantum algorithms”. In: *Reviews of Modern Physics* 94.1 (Feb. 15, 2022). Publisher: American Physical Society, p. 015004. DOI: 10.1103/RevModPhys.94.015004.

[50] K. Bertels, A. Sarkar, T. Hubregtsen, M. Serrao, A. A. Mouedenne, A. Yadav, A. Krol, I. Ashraf, and C. G. Almudever. “Quantum computer architecture toward full-stack quantum accelerators”. In: *IEEE Transactions on Quantum Engineering* 1 (2020). Publisher: IEEE, pp. 1–17. DOI: 10.1109/TQE.2020.2981074.

[51] X. Fu et al. “eQASM: An Executable Quantum Instruction Set Architecture”. In: *HPCA*. IEEE, 2019, pp. 224–237. DOI: 10.1109/HPCA.2019.00040.

[52] E. Giortamis, F. Romão, N. Tornow, and P. Bhatotia. "QOS: A Quantum Operating System". In: *arXiv preprint arXiv:2406.19120* (2024). DOI: 10.48550/arXiv.2406.19120.

[53] W. Kong, J. Wang, Y. Han, Y. Wu, Y. Zhang, M. Dou, Y. Fang, and G. Guo. "Origin Pilot: a Quantum Operating System for Effecient Usage of Quantum Resources". 2021. arXiv: 2105.10730.

[54] P. C. Humphreys, N. Kalb, J. P. J. Morits, R. N. Schouten, R. F. L. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson. "Deterministic Delivery of Remote Entanglement on a Quantum Network". In: *Nature* 558.7709 (2018), pp. 268–273. DOI: 10.1038/s41586-018-0200-5.

[55] M. Skrzypczyk and S. Wehner. "An Architecture for Meeting Quality-of-Service Requirements in Multi-User Quantum Networks". 2021. arXiv: 2111.13124.

[56] P. Drmota, D. Main, D. Nadlinger, B. Nichol, M. Weber, E. Ainley, A. Agrawal, R. Srinivas, G. Araneda, C. Ballance, et al. "Robust quantum memory in a trapped-ion quantum network node". In: *Physical Review Letters* 130.9 (2023). Publisher: APS, p. 090803. DOI: 10.1103/PhysRevLett.130.090803.

[57] V. Krutyanskiy, M. Meraner, J. Schupp, V. Krcmarsky, H. Hainzer, and B. P. Lanyon. "Light-matter entanglement over 50 km of optical fibre". In: *npj Quantum Information* 5.1 (Aug. 2019), p. 72. DOI: 10.1038/s41534-019-0186-3.

[58] G. Vardoyan, M. Skrzypczyk, and S. Wehner. "On the Quantum Performance Evaluation of two Distributed Quantum Architectures". In: *Performance Evaluation* 153 (2022), pp. 1–26. DOI: 10.1016/j.peva.2021.102242.

[59] J. D. McCullough, K. H. Speierman, and F. W. Zurcher. "A design for a multiple user multiprocessing system". In: *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I.* 1965, pp. 611–617. DOI: 10.1145/1463891.1463957.

[60] J. B. Dennis. "Segmentation and the design of multiprogrammed computer systems". In: *Journal of the ACM (JACM)* 12.4 (1965). Publisher: ACM New York, NY, USA, pp. 589–602. DOI: 10.1145/321296.321310.

[61] J. B. Dennis and E. C. Van Horn. "Programming semantics for multiprogrammed computations". In: *Communications of the ACM* 9.3 (1966). Publisher: ACM New York, NY, USA, pp. 143–155. DOI: 10.1145/365230.365252.

[62] A. S. Tanenbaum and A. S. Woodhull. *Operating Systems Design and Implementation (3rd Edition)*. USA: Prentice-Hall, Inc., Nov. 2005.

[63] M. Teller, D. A. Fioretto, P. C. Holz, P. Schindler, V. Messerer, K. Schüppert, Y. Zou, R. Blatt, J. Chiaverini, J. Sage, et al. "Heating of a trapped ion induced by dielectric materials". In: *Physical Review Letters* 126.23 (2021), p. 230505. DOI: 10.1103/PhysRevLett.126.230505.

[64] Z. Instruments. *HDAWG 750 MHz Arbitrary Waveform Generator.* HDAWG 750 MHz Arbitrary Waveform Generator. 2019. URL: https://www.zhinst.com/europe/en/products/hdawg-arbitrary-waveform-generator.

[65] C. L. Liu and J. W. Layland. "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment". In: *J. ACM* 20.1 (1973), pp. 46–61. DOI: 10.1145/321738.32174.

[66] R. C. Daley and J. B. Dennis. "Virtual memory, processes, and sharing in Multics". In: *Communications of the ACM* 11.5 (1968). Publisher: ACM New York, NY, USA, pp. 306–312. DOI: 10.1145/363095.363139.

[67] J. L. Peterson and A. Silberschatz. *Operating system concepts*. Addison-Wesley Longman Publishing Co., Inc., 1985. DOI: 10.5555/3526.

[68] G. L. Chesson. "The network UNIX system". In: *ACM SIGOPS Operating Systems Review* 9.5 (1975). Publisher: ACM New York, NY, USA, pp. 60–66. DOI: 10.1145/1067629.806522.

[69] P. Leach, P. Levine, B. Douros, J. Hamilton, D. Nelson, and B. Stumpf. "The architecture of an integrated local network". In: *IEEE Journal on selected Areas in Communications* 1.5 (1983). Publisher: IEEE, pp. 842–857. DOI: 10.1109/JSAC.1983.1146002.

[70] S. Massar and S. Popescu. "Optimal Extraction of Information from Finite Quantum Ensembles". In: *Physical Review Letters* 74.8 (Feb. 1995), pp. 1259–1263. DOI: 10.1103/PhysRevLett.74.1259.

[71] K. Ramamritham and J. A. Stankovic. "Scheduling algorithms and operating systems support for real-time systems". In: *Proceedings of the IEEE* 82.1 (1994). Publisher: IEEE, pp. 55–67. DOI: 10.1109/5.259426.

[72] *ADwin-Pro II*. Jäger GmbH. URL: https://www.adwin.de/us/produkte/proII.html (visited on 02/28/2023).

[73] G. De Lange, Z. H. Wang, D. Ristè, V. V. Dobrovitski, and R. Hanson. "Universal Dynamical Decoupling of a Single Solid-State Spin from a Spin Bath". In: *Science* 330.6000 (Oct. 2010), pp. 60–63. DOI: 10.1126/science.1192739.

[74] M. H. Abobeih, J. Cramer, M. A. Bakker, N. Kalb, M. Markham, D. J. Twitchen, and T. H. Taminiau. "One-Second Coherence for a Single Electron Spin Coupled to a Multi-Qubit Nuclear-Spin Environment". In: *Nature Commun.* 9.1 (2018), pp. 1–8. DOI: 10.1038/s41467-018-04916-z.

[75] A. Corna. *Efficient Generation of Dynamic Pulses*. 2021. URL: https://www.zhinst.com/europe/en/blogs/efficient-generation-dynamic-pulses.

[76] *FreeRTOS Real-Time Operating System for Microcontrollers*. Amazon Web Services. URL: https://www.freertos.org/ (visited on 02/28/2023).

[77] *MicroZed Development Board*. Avnet. URL: https://www.avnet.com/wps/portal/us/products/avnet-boards/avnet-board-families/microzed/ (visited on 02/28/2023).

[78] M. Teller. "Measuring and modeling electric-field noise in an ion-cavity system (PhD thesis)". PhD thesis. University of Innsbruck, July 2021.

[79] M. Paris and J. Rehacek. *Quantum state estimation*. Vol. 649. Springer Science & Business Media, 2004.

[80] C. Cabrillo, J. I. Cirac, P. Garcia-Fernandez, and P. Zoller. “Creation of entangled states of distant atoms by interference”. In: *Physical Review A* 59.2 (1999), p. 1025. DOI: 10.1103/PhysRevA.59.1025.

[81] S. Bose, P. Knight, M. Plenio, and V. Vedral. “Proposal for teleportation of an atomic state via cavity decay”. In: *Physical Review Letters* 83.24 (1999), p. 5158. DOI: 10.1103/PhysRevLett.83.5158.

[82] S. Hermans, M. Pompili, L. D. S. Martins, A. R. Montblanch, H. Beukers, S. Baier, J. Borregaard, and R. Hanson. “Entangling remote qubits using the single-photon protocol: an in-depth theoretical and experimental study”. In: *New Journal of Physics* 25.1 (2023), p. 013011. DOI: 10.1088/1367-2630/acb004.

[83] L. Robledo, H. Bernien, I. Van Weperen, and R. Hanson. “Control and Coherence of the Optical Transition of Single Nitrogen-Vacancy Centers in Diamond”. In: *Physical review letters* 105.17 (2010), p. 177403. DOI: 10.1103/PhysRevLett.105.177403.

[84] W. Kozlowski and S. Wehner. “Towards Large-Scale Quantum Networks”. In: *NANOCOM*. ACM, 2019, pp. 1–7. DOI: 10.1145/3345312.3345497.

[85] C. H. Bennett and G. Brassard. “Quantum Cryptography: Public Key Distribution and Coin Tossing”. In: *Proceedings of the International Conference on Computers, Systems and Signal Processing*. Bangalore, India, Dec. 1984, pp. 175–179.

[86] A. K. Ekert. “Quantum Cryptography Based on Bell’s Theorem”. In: *Phys. Rev. Lett.* 67.6 (1991), pp. 661–663. DOI: 10.1103/PhysRevLett.67.661.

[87] R. Stankiewicz, P. Chołda, and A. Jajszczyk. “QoX: What is It Really?” In: *IEEE Communications Magazine* 49.4 (2011), pp. 148–158. DOI: 10.1109/MCOM.2011.5741159.

[88] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. “Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels”. In: *Phys. Rev. Lett.* 70 (13 Mar. 1993), pp. 1895–1899. DOI: 10.1103/PhysRevLett.70.1895.

[89] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating Systems Concepts (Ninth Edition)*. Wiley, 2014.

[90] Android. *Android Hardware abstraction layer (HAL) overview*. Accessed: 2025-01-14. 2025. URL: https://source.android.com/docs/core/architecture/hal (visited on 01/14/2025).

[91] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. W. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson. “Entanglement Distillation Between Solid-State Quantum Network Nodes”. In: *Science* 356.6341 (2017), pp. 928–932. DOI: 10.1126/science.aan0070.

[92] X. Fu et al. “An Experimental Microarchitecture for a Superconducting Quantum Processor”. In: *MICRO*. ACM, 2017, pp. 813–825. DOI: 10.1145/3123939.3123952.

[93] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta. “Open Quantum Assembly Language”. 2017. arXiv: 1707.03429.

[94] N. Khammassi, G. G. Guerreschi, I. Ashraf, J. W. Hogaboam, C. G. Almudever, and K. Bertels. “cQASM v1.0: Towards a Common Quantum Assembly Language ”. 2018. arXiv: 1805.09607.

[95] *NetQASM SDK.* QuTech. URL: https://github.com/QuTech-Delft/netqasm (visited on 02/28/2023).

[96] *SquidASM: a Simulator that Executes NetQASM Applications.* QuTech. URL: https://github.com/QuTech-Delft/squidasm (visited on 07/14/2024).

[97] *pthreads(7) - Linux manual page.* URL: https://man7.org/linux/man-pages/man7/pthreads.7.html (visited on 07/15/2024).

[98] *eRPC (Embedded RPC).* Freescale Semiconductor, Inc. URL: https://github.com/EmbeddedRPC/erpc (visited on 07/08/2024).

[99] K. Azuma, S. E. Economou, D. Elkouss, P. Hilaire, L. Jiang, H.-K. Lo, and I. Tzitrin. “Quantum repeaters: From Quantum Networks to the Quantum Internet”. In: *Rev. Mod. Phys.* 95 (4 Oct. 2023), pp. 045006-1–045006-66. DOI: 10.1103/RevModPhys.95.045006.

[100] P. Murali, N. M. Linke, M. Martonosi, A. J. Abhari, N. H. Nguyen, and C. H. Alderete. “Full-Stack, Real-System Quantum Computer Studies: Architectural Comparisons and Design Insights”. In: *ISCA*. ACM, 2019, pp. 527–540. DOI: 10.1145/3307650.3322273.

[101] S. Baier, C. E. Bradley, T. Middelburg, V. V. Dobrovitski, T. H. Taminiau, and R. Hanson. “Orbital and Spin Dynamics of Single Neutrally-Charged Nitrogen-Vacancy Centers in Diamond”. In: *Phys. Rev. Lett.* 125 (19 Nov. 2020), pp. 193601-1–193601-6. DOI: 10.1103/PhysRevLett.125.193601.

[102] L. C. Bassett, F. J. Heremans, C. G. Yale, B. B. Buckley, and D. D. Awschalom. “Electrical Tuning of Single Nitrogen-Vacancy Center Optical Transitions Enhanced by Photoinduced Fields”. In: *Phys. Rev. Lett.* 107 (26 Dec. 2011), pp. 266403-1–266403-5. DOI: 10.1103/PhysRevLett.107.266403.

[103] S. W. Warren. “Effects of arbitrary laser or NMR pulse shapes on population inversion and coherence”. In: *The Journal of Chemical Physics* 81 (1984). DOI: 10.1063/1.447644.

[104] L. M. K. Vandersypen and I. L. Chuang. “NMR techniques for quantum control and computation”. In: *Rev. Mod. Phys.* 76 (4 Jan. 2005), pp. 1037–1069. DOI: 10.1103/RevModPhys.76.1037.

[105] T. Coopmans et al. “NetSquid, a NETwork Simulator for QUantum Information using Discrete events”. In: *Communications Physics* 4.1 (2021), p. 164. DOI: 10.1038/s42005-021-00647-8.

6 Conclusions & Outlook

6.1. Summary

The scope of this thesis is to give an overview of the current state-of-the-art and progress towards the realization of the Quantum Internet using diamond-based qubits. The demonstrations presented in this dissertation are made possible by a test-bed setup based on the NV center defect. To this end, several topics have been discussed:

- in **Chapter 3**, the generation of a distributed 4-partite Greenberger-Horne-Zeilinger state and the realization of a non-local Controlled-NOT gate are presented. The setup is composed of two NV nodes that are separated by 2m. Each NV node consists of one communication qubit and one memory qubit, where the latter is realized by controlling the nuclear spin of a nearby ^{13}C in the diamond lattice. In this framework, non-local operations, specifically the realization of quantum gates between remote qubits, the memory qubits, can be accomplished by exploiting quantum teleportation protocols. The minimal requirements for such an implementation envisage a Bell pair, local gates within each quantum processor, and classical communication between the remote nodes to feed back in real-time single-qubit gates based on measurement outcomes. The results open the way for the exploration of further quantum network applications, extending the key capabilities of diamond-based quantum nodes.

- **Chapter 4** presents a photonic interface between an NV center and a time-bin qubit compatible with the photon emission from a Thulium-doped crystal or a Rubidium gas quantum memory. The benchmark of the interface is completed by performing a quantum teleportation protocol including real-time feed-forward. The study, supported by theoretical modeling, shows insights into the determining factors of the interface, making the investigation valid for a broader range of similar platforms.

- **Chapter 5** introduces the implementation of the first quantum network operating system, QNodeOS. QNodeOS's capabilities include the execution of quantum network applications on quantum processors in platform-independent high-level software, and multitasking of users' requests to optimize the hardware utilization. The architecture is benchmarked via a form of delegated quantum computation, utilizing one qubit per node in a server-client configuration. The protocol allows for the validation of the elementary instructions that are needed for programming more complex applications. Such basic instructions include: preparation, manipulation and measurement of the qubits, remote entanglement generation, compilation of feedforwarded instructions based on measurement outcomes. A first step to bring quantum network technology to society.

6.2. Scaling the Quantum Internet

To fulfill the vision of the Quantum Internet, its components must ultimately be deployed on a global scale and operated seamlessly by end-users. Achieving this goal requires progress not only in quantum physics and computer science, but also in material science, electronics, cryogenics, and laser engineering. Bridging the gap between laboratory demonstrations and large-scale deployment is therefore a highly interdisciplinary challenge.

All demonstrations with NV-based quantum processors presented in this thesis are realized in a laboratory environment, as shown in Figure 6.1. The picture immediately conveys the message of this section: current demonstrators exhibit a maturity and readiness level that is not yet compatible with deployment in societal or industrial settings. Several factors limit their scalability, including platform performance and qubit number, the physical footprint of the hardware (clearly visible in Figure 6.1), system reliability and robustness, long-term stability, cost, and environmental sustainability.
Despite providing an excellent testbed to advance research on many aspects of quantum networking, the NV platform shows performance limitations that currently hinder the delivery of useful, application-relevant network functionalities. One urgent parameter to improve is the rate of detected single photons for entanglement generation. A common strategy to mitigate this issue is to embed the NV center in optical cavities, enhancing its emission into the ZPL and improving photon extraction efficiency [1]. Recent results demonstrate a 10-fold improvement in detection probability, with a 30-fold improvement within reach, when coupling a single NV to a fiber-based open micro-cavity [2]. A description of such a setup is included in Ref. [3], which shows a similar spatial layout to Figure 6.1 and therefore does not yet address the scalability-in-space challenge.

An alternative route towards compact and scalable architectures relies on integrated photonics. From the emitter perspective, NV centers are not inherently suitable for nanophotonic structures such as waveguides or photonic crystal cavities, as their strong sensitivity to electric fields constrains their position far from the diamond surface. Other color centers in diamond, specifically the group-IV defects (SiV [4], GeV [5], SnV [6], and PbV [7]), offer a promising alternative. Their inversion-symmetric electronic structure results in reduced spectral diffusion and allows integration into nanophotonic devices, providing intrinsically improved optical interfaces, besides showing a higher Debye-Waller factor as a starting point. Currently, the most advanced platform in this direction is the

SiV center, which has already demonstrated remote entanglement generation at metropolitan distances [8], as well as a conditional form of blind quantum computation [9].
Another key advantage of group-IV defects is the compatibility of nanofabricated diamond devices with heterogeneous photonic materials (e.g. silicon nitride), enabling on-chip integration of passive photonic components, quantum frequency converters, and photodetectors [10]. Such hybrid platforms pave the way for compact, portable, and fully integrated end-node chips. The main drawback, however, is that group-IV color centers typically require operation at temperatures much lower than those needed for NV centers, except for the PbV center, whose drawback is mainly due to the difficulty of implanting Pb isotopes without significant damage to the diamond structure [7, 11].
This introduces significant challenges in the cryogenic domain, ranging from efficient cooling and thermal stability to scalability, that must be addressed before these platforms can be deployed in real-world networks.

6.2.1. A scheme to entangle them all

The single-click protocol is widely adopted in all recent NV-center demonstrations [12–17], including this thesis. As also shown in Chapter 4, a double-click scheme in the current photon detection probability regime proposes almost unsustainable success rates even for laboratory demonstrations. However, as presented in Chapter 2, the higher success rate of the single-click rate is paid for with limitations on the maximum achievable fidelity. Therefore, is there a unique entanglement generation scheme that wins over all?
To answer this question, we only focus on time-bin and Fock-state encoding schemes. A broader picture of other possible entangling schemes is presented in Ref. [18].
Considering the simplest scenario of no experimental errors besides imperfect visibility V of the photons emitted by the two nodes, the achievable fidelity $\mathscr{F}$ for the double-click scheme scales as $\frac{1}{2}+\frac{1}{2}\cdot V$. Assuming a realistic value $V=0.9$, we obtain $\mathscr{F}=0.95$.
In the single-click protocol, the maximum achievable fidelity is $\frac{1}{2}\cdot(1-\alpha)(1+\sqrt{V})$. For $\alpha\leq 0.025$, the single-click protocol achieves higher fidelity than the double-click. In Figure 6.2, the success probability as a function of the detection probability is illustrated for the double-click protocol and for several values of α in the single-click protocol.

What emerges from the plot in Figure 6.2 is that with detection probabilities above 0.1, the advantage of the single-click protocol in success probability vanishes at low α. For α above 0.3, the single-click protocol is again dominant in success probability, however the maximum achievable fidelity is $\mathscr{F}=0.68$. Entanglement fidelity can be increased with a memory-assisted protocol for entanglement distillation [13, 19–21] requiring the generation of multiple low-fidelity entangled states, storage of the generated entangled state and rounds of classical communication, thus reducing the rate of high-fidelity entangled state generation due to increased overhead time. Additionally, given the necessity of swapping the generated entangled states into memory qubits, it requires fast and high-fidelity single and two-qubit gates.
Multiplexing offers an interesting solution for speeding up entanglement generation rates. In our configuration, multiplexing entails the parallel addressing of multiple emitters [22, 23] and it can be used with both schemes. In this scenario, a major drawback for the

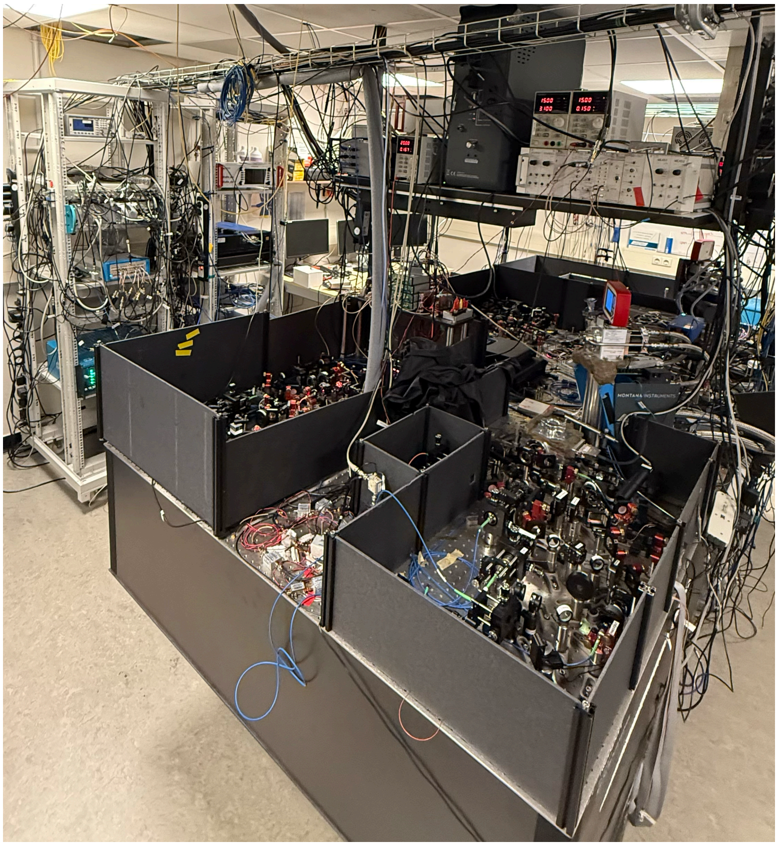


Figure 6.1: **A single quantum network end node.** Picture of Bob (Client) featured in Chapter 3 (5). The footprint of a single quantum network node includes half of the optical table, the racks captured on the side, the visible electronics on top of the optical table and the PC next to the racks.

single-click protocol, however, is the need for phase stabilization protocols and optical phase calibrations for each emitter before the node is able to deliver any user's requests. These calibrations occupy the mid-point detectors, generating extra downtime [17, 24] which scales with the number of modes (emitters). The double-click protocol does not require phase stabilization and, therefore, can be used without the need to subtract operational time to the mid-point, as all the required calibrations are performed locally at

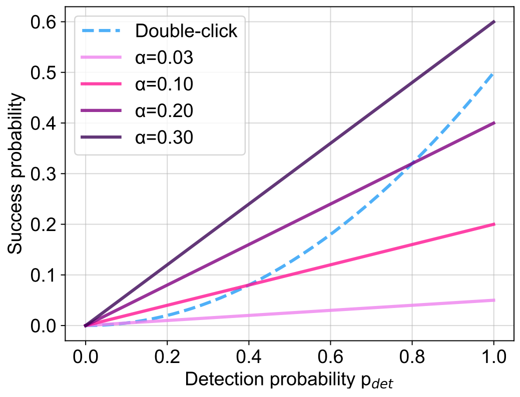


Figure 6.2: **Success probability comparison.** Curves of the success probability versus detection probability p_{det} are shown for the double-click protocol and the single-click protocol with varying α. For high p_{det}, the double-click protocol outperforms the single-click protocol with low α. For $\alpha = 0.3$, the single-click protocol achieves a higher success probability than the double-click protocol, although the fidelity remains low without distillation.

each node.
In this perspective, a possible path forward for scaling solid-state-based quantum network nodes is to focus on improving the detection probability of single photons, whose direct benefit is the improved success probability of entanglement generation. Considering the results shown in Chapter 3, a higher entanglement rate enables to deterministically performing distributed quantum computation protocols, for instance. In this way, the strict requirements on the data qubit coherence time are relaxed, given that in the current framework deterministic approaches are guaranteed for $N_{1/e}$ above 10^5 [25].

6.3. AUTOMATION: QUANTUM & AI

The experiments performed in Chapters 3 and 5 required constant human presence and manual intervention, whereas the experiment in Chapter 4 represents a first step toward continuous, autonomous operation of a quantum network node. In realistic network scenarios, the operational uptime of each node is a critical performance metric: any interruption directly impacts the success probability of remote entanglement generation and thus the viability of quantum communication protocols at scale. As the number of nodes increases and network traffic becomes denser, maintaining a high entanglement-generation rate while minimizing downtime becomes essential for demonstrating a practical quantum advantage.

Considering the setup in Chapter 3 as a representative example of a fully functional network node, a substantial portion of the experimental overhead arises from the human supervision required to keep the optical transitions of the two remote nodes resonant with each other. This significantly prolongs procedures such as the Charge-Resonance (CR) check. Although long-distance entanglement distribution ultimately relies on quantum frequency conversion to the telecom band [8, 17], Stark tuning remains a versatile tool: it enables fine control of the local strain environment, improving the cyclicity of optical transitions and readout fidelity. This tunability relaxes the otherwise stringent requirement of locating a "good" emitter on a chip, thereby increasing yield. The application of a DC voltage induces a Stark shift of the ground and excited state energy levels of the NV center [26, 27]. However, the surroundings of the NV are not a quiet place: nearby defects such as substitutional nitrogen (P1 centers), charge traps, or other parasitic impurities also respond to the applied electric field. Over time, these defects can undergo ionization or slow charge rearrangements, thereby modifying the local electric-field environment. Because the NV's optical and spin properties are highly sensitive to changes in its charge environment, constituting an asset for quantum sensing applications [28], but a liability for long-term quantum network operation, these environmental fluctuations require continuous adjustments of the DC voltage to maintain the desired resonance frequency. These variations typically manifest as slow drifts, which can be tracked and corrected using a PID controller, as also implemented in this thesis. However, abrupt and sporadic large "voltage jumps" (≥0.5V) also occur on the timescale of minutes. These jumps require the experimenter to manually search for the new voltage that restores the desired resonance frequency, representing the main bottleneck that prevents the system from running unattended. Both the probability and magnitude of these jumps vary significantly among samples, as they depend sensitively on fabrication conditions, defect densities, implantation depth, and surface preparation, parameters that are not always reproducible.

Adaptive DC voltage is used in group-IV devices that have the capability of mechanically strain-tuning their emitters [29–33]. A similar mechanism has also been shown for other defects in solid-state platforms, such as color centers in silicon [34]. Spin properties compatible with quantum information processing and remote entanglement generation have been demonstrated in several strain regimes [35–38]. For the SnV, microwave control has been demonstrated at temperatures up to 4K for highly strained emitters (strain comparable to the spin-orbit coupling of 830GHz) [36]. Strain is essential for microwave control of the spin in group-IV color centers since it mixes the orbital states, allowing direct transitions between the qubit states [39]. Although the devices with adaptive strain-tuning capabilities cannot apply enough strain to have a substantial change in the spin properties and enable efficient microwave control by themselves, their advantage in controlling the optical frequencies is predominant, bringing improvements in both continuous operation of the network node and relaxed requirements on the emitter's natural resonance frequency. The group-IV color centers have a very low susceptibility to variation of the local charge environment [40], however, sudden jumps and slow drifts are still possible, with a lower frequency compared to the NV. As shown in Ref. [33], a PID loop on the CR check counts can stabilize the optical lines for hours, compensating for the slow drifts. The sporadic bigger jumps for their unclear nature and unpredictability still require human interaction for their correction.

Artificial intelligence (AI) is nowadays obtaining increased attention and its application field is expanding, landing also in quantum technologies [41]. In this regard, a promising pathway to mitigate these manual interventions and extensive search for emitters is the introduction of automated control systems based on reinforcement-learning (RL) agents [42, 43]. A suitably trained RL agent could replace the human operator by reacting more quickly to resonance drifts, autonomously executing recovery procedures after voltage jumps, and potentially predicting these jumps from statistical patterns in the time traces of CR-check count rates. Such an approach could enable genuinely autonomous, long-term operation of quantum-network nodes, pushing them closer to deployment-ready technologies. Moving forward, this can be expanded to an AI-powered full quantum node manager, which can be incorporated in the system architecture described in Chapter 5, for instance. The task of such a manager is to identify and predict operational up and down times of the network node, and schedule calibration routines in an adaptive way, based on the analysis of a larger parameter space, perhaps including features and variable that are not even taken into account at the moment [44], when decisions are made exclusively by human intervention, opening the way for new and unexplored optimization paths.

REFERENCES

[1] M. Ruf, N. H. Wan, H. Choi, D. Englund, and R. Hanson. “Quantum networks based on color centers in diamond”. In: *Journal of Applied Physics* 130.7 (Aug. 2021), p. 070901. DOI: 10.1063/5.0056534.

[2] J. Fischer, Y. Herrmann, C. F. J. Wolfs, S. Scheijen, M. Ruf, and R. Hanson. “Spin-photon correlations from a Purcell-enhanced diamond nitrogen-vacancy center coupled to an open microcavity”. In: *Nature Communications* 16.1 (Nov. 2025), p. 11680. DOI: 10.1038/s41467-025-66722-8.

[3] Y. Herrmann et al. “A low-temperature tunable microcavity featuring high passive stability and microwave integration”. In: *AVS Quantum Science* 6.4 (Dec. 2024), p. 041401. DOI: 10.1116/5.0233296.

[4] D. D. Sukachev, A. Sipahigil, C. T. Nguyen, M. K. Bhaskar, R. E. Evans, F. Jelezko, and M. D. Lukin. “Silicon-Vacancy Spin Qubit in Diamond: A Quantum Memory Exceeding 10 ms with Single-Shot State Readout”. In: *Physical Review Letters* 119.22 (Nov. 2017), p. 223602. DOI: 10.1103/PhysRevLett.119.223602.

[5] M. K. Bhaskar et al. “Quantum Nonlinear Optics with a Germanium-Vacancy Color Center in a Nanoscale Diamond Waveguide”. In: *Physical Review Letters* 118.22 (May 2017), p. 223603. DOI: 10.1103/PhysRevLett.118.223603.

[6] A. E. Rugar, C. Dory, S. Sun, and J. Vučković. “Characterization of optical and spin properties of single tin-vacancy centers in diamond nanopillars”. In: *Physical Review B* 99.20 (May 2019), p. 205417. DOI: 10.1103/PhysRevB.99.205417.

[7] P. Wang et al. “Transform-Limited Photon Emission from a Lead-Vacancy Center in Diamond above 10 K”. In: *Physical Review Letters* 132.7 (Feb. 2024), p. 073601. DOI: 10.1103/PhysRevLett.132.073601.

[8] C. M. Knaut et al. “Entanglement of nanophotonic quantum memory nodes in a telecom network”. In: *Nature* 629.8012 (May 2024), pp. 573–578. DOI: 10.1038/s41586-024-07252-z.

[9] Y.-C. Wei et al. “Universal distributed blind quantum computing with solid-state qubits”. In: *Science* 388.6746 (May 2025), pp. 509–513. DOI: 10.1126/science.adu6894.

[10] J.-H. Kim, S. Aghaeimeibodi, J. Carolan, D. Englund, and E. Waks. “Hybrid integration methods for on-chip quantum photonics”. In: *Optica* 7.4 (Apr. 2020), p. 291. DOI: 10.1364/OPTICA.384118.

[11] P. Wang, T. Taniguchi, Y. Miyamoto, M. Hatano, and T. Iwasaki. “Low-Temperature Spectroscopic Investigation of Lead-Vacancy Centers in Diamond Fabricated by High-Pressure and High-Temperature Treatment”. In: *ACS Photonics* 8.10 (Oct. 2021), pp. 2947–2954. DOI: 10.1021/acsphotonics.1c00840.

[12] P. C. Humphreys, N. Kalb, J. P. J. Morits, R. N. Schouten, R. F. L. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson. “Deterministic delivery of remote entanglement on a quantum network”. In: *Nature* 558.7709 (June 2018), pp. 268–273. DOI: 10.1038/s41586-018-0200-5.

[13] N. Kalb, A. A. Reiserer, P. C. Humphreys, J. J. W. Bakermans, S. J. Kamerling, N. H. Nickerson, S. C. Benjamin, D. J. Twitchen, M. Markham, and R. Hanson. "Entanglement Distillation Between Solid-State Quantum Network Nodes". In: *Science* 356.6341 (2017), pp. 928–932. DOI: 10.1126/science.aan0070.

[14] M. Pompili et al. "Realization of a Multinode Quantum Network of Remote Solid-State Qubits". In: *Science* 372.6539 (2021), pp. 259–264. DOI: 10.1126/science.abg1919.

[15] S. L. N. Hermans, M. Pompili, H. K. C. Beukers, S. Baier, J. Borregaard, and R. Hanson. "Qubit teleportation between non-neighbouring nodes in a quantum network". In: *Nature* 605.7911 (May 2022), pp. 663–668. DOI: 10.1038/s41586-022-04697-y.

[16] M. Pompili et al. "Experimental demonstration of entanglement delivery using a quantum network stack". In: *npj Quantum Information* 8.1 (Oct. 2022), p. 121. DOI: 10.1038/s41534-022-00631-2.

[17] A. J. Stolk et al. "Metropolitan-scale heralded entanglement of solid-state qubits". In: *Science Advances* 10.44 (Nov. 2024), eadp6442. DOI: 10.1126/sciadv.adp6442.

[18] H. K. Beukers, M. Pasini, H. Choi, D. Englund, R. Hanson, and J. Borregaard. "Remote-Entanglement Protocols for Stationary Qubits with Photonic Interfaces". In: *PRX Quantum* 5.1 (Mar. 2024). Publisher: American Physical Society, p. 010202. DOI: 10.1103/PRXQuantum.5.010202.

[19] E. T. Campbell and S. C. Benjamin. "Measurement-Based Entanglement under Conditions of Extreme Photon Loss". In: *Physical Review Letters* 101.13 (Sept. 2008), p. 130502. DOI: 10.1103/PhysRevLett.101.130502.

[20] N. H. Nickerson, J. F. Fitzsimons, and S. C. Benjamin. "Freely Scalable Quantum Technologies Using Cells of 5-to-50 Qubits with Very Lossy and Noisy Photonic Links". In: *Physical Review X* 4.4 (Dec. 2014), p. 041041. DOI: 10.1103/PhysRevX.4.041041.

[21] F. Rozpędek, T. Schiet, L. P. Thinh, D. Elkouss, A. C. Doherty, and S. Wehner. "Optimizing practical entanglement distillation". In: *Physical Review A* 97.6 (June 2018), p. 062333. DOI: 10.1103/PhysRevA.97.062333.

[22] S. B. V. Dam, P. C. Humphreys, F. Rozpędek, S. Wehner, and R. Hanson. "Multiplexed entanglement generation over quantum networks using multi-qubit nodes". In: *Quantum Science and Technology* 2.3 (Sept. 2017), p. 034002. DOI: 10.1088/2058-9565/aa7446.

[23] A. Ruskuc, C.-J. Wu, E. Green, S. L. N. Hermans, W. Pajak, J. Choi, and A. Faraon. "Multiplexed entanglement of multi-emitter quantum network nodes". In: *Nature* 639.8053 (Mar. 2025), pp. 54–59. DOI: 10.1038/s41586-024-08537-z.

[24] A. J. Stolk, J. J. B. Biemond, K. L. v. d. Enden, L. v. Dooren, E. J. v. Zwet, and R. Hanson. *Extendable optical phase synchronization of remote and independent quantum network nodes over deployed fibers*. arXiv:2408.12464 [quant-ph]. Aug. 2024. DOI: 10.48550/arXiv.2408.12464.

[25] C. E. Bradley et al. “Robust quantum-network memory based on spin qubits in isotopically engineered diamond”. In: *npj Quantum Information* 8.1 (Oct. 2022), p. 122. DOI: 10.1038/s41534-022-00637-w.

[26] P. Tamarat et al. “Stark Shift Control of Single Optical Centers in Diamond”. In: *Physical Review Letters* 97.8 (Aug. 2006), p. 083002. DOI: 10.1103/PhysRevLett.97.083002.

[27] L. C. Bassett, F. J. Heremans, C. G. Yale, B. B. Buckley, and D. D. Awschalom. “Electrical Tuning of Single Nitrogen-Vacancy Center Optical Transitions Enhanced by Photoinduced Fields”. In: *Phys. Rev. Lett.* 107 (26 Dec. 2011), pp. 266403-1–266403-5. DOI: 10.1103/PhysRevLett.107.266403.

[28] F. Dolde et al. “Electric-field sensing using single diamond spins”. In: *Nature Physics* 7.6 (June 2011), pp. 459–463. DOI: 10.1038/nphys1969.

[29] S. Meesala et al. “Strain engineering of the silicon-vacancy center in diamond”. In: *Physical Review B* 97.20 (May 2018), p. 205444. DOI: 10.1103/PhysRevB.97.205444.

[30] Y.-I. Sohn et al. “Controlling the coherence of a diamond spin qubit through its strain environment”. In: *Nature Communications* 9.1 (May 2018), p. 2012. DOI: 10.1038/s41467-018-04340-3.

[31] B. Machielse et al. “Quantum Interference of Electromechanically Stabilized Emitters in Nanophotonic Devices”. In: *Physical Review X* 9.3 (Aug. 2019), p. 031022. DOI: 10.1103/PhysRevX.9.031022.

[32] G. Clark et al. “Nanoelectromechanical Control of Spin–Photon Interfaces in a Hybrid Quantum System on Chip”. In: *Nano Letters* 24.4 (Jan. 2024), pp. 1316–1323. DOI: 10.1021/acs.nanolett.3c04301.

[33] J. M. Brevoord et al. “Large-range tuning and stabilization of the optical transition of diamond tin-vacancy centers by in situ strain control”. In: *Applied Physics Letters* 126.17 (Apr. 2025). DOI: 10.1063/5.0251211.

[34] A. Buzzi, C. Papon, M. Pirro, O. Hooybergs, H. Raniwala, V. Saggio, C. Errando-Herranz, and D. Englund. “Spectral tuning and nanoscale localization of single color centers in silicon via controllable strain”. In: *Nature Communications* 16.1 (Oct. 2025), p. 8829. DOI: 10.1038/s41467-025-63871-8.

[35] E. I. Rosenthal et al. “Microwave Spin Control of a Tin-Vacancy Qubit in Diamond”. In: *Physical Review X* 13.3 (Aug. 2023), p. 031022. DOI: 10.1103/PhysRevX.13.031022.

[36] X. Guo et al. “Microwave-Based Quantum Control and Coherence Protection of Tin-Vacancy Spin Qubits in a Strain-Tuned Diamond-Membrane Heterostructure”. In: *Physical Review X* 13.4 (Nov. 2023), p. 041037. DOI: 10.1103/PhysRevX.13.041037.

[37] I. Karapatzakis et al. “Microwave Control of the Tin-Vacancy Spin Qubit in Diamond with a Superconducting Waveguide”. In: *Physical Review X* 14.3 (Aug. 2024), p. 031036. DOI: 10.1103/PhysRevX.14.031036.

[38] H. K. C. Beukers et al. “Control of Solid-State Nuclear Spin Qubits Using an Electron Spin- 1 / 2”. In: *Physical Review X* 15.2 (Apr. 2025), p. 021011. DOI: 10.1103/PhysRevX.15.021011.

[39] F. Gu. *The Group-IV-Vacancy Color Center in Diamond.* arXiv:2509.00443 [quant-ph]. Aug. 2025. DOI: 10.48550/arXiv.2509.00443.

[40] L. De Santis, M. Trusheim, K. Chen, and D. Englund. “Investigation of the Stark Effect on a Centrosymmetric Quantum Emitter in Diamond”. In: *Physical Review Letters* 127.14 (Sept. 2021), p. 147402. DOI: 10.1103/PhysRevLett.127.147402.

[41] Y. Alexeev et al. “Artificial intelligence for quantum computing”. In: *Nature Communications* 16.1 (Dec. 2025), p. 10829. DOI: 10.1038/s41467-025-65836-3.

[42] R. S. Sutton and A. Barto. *Reinforcement learning: an introduction.* Nachdruck. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2014.

[43] A. K. Shakya, G. Pillai, and S. Chakrabarty. “Reinforcement learning algorithms: A brief survey”. In: *Expert Systems with Applications* 231 (Nov. 2023), p. 120495. DOI: 10.1016/j.eswa.2023.120495.

[44] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. “Adversarial examples are not bugs, they are features”. In: *Advances in neural information processing systems.* Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

ACKNOWLEDGEMENTS

Four years are long enough to see someone change, grow, heal, win and fail, lose and fall in love again, but also short enough to still remember the first day as if it were yesterday. I take this space to give my sincerest “thank you” to all the people who made these years and my PhD experience memorable.
First of all, I want to thank **Ronald**, my supervisor. I am truly grateful for the opportunities and the support you gave me from day one. Your mentorship was invaluable and made me a better scientist day after day. You are always ready to listen without judgment, showing care for my personal well-being before success in the lab. I am impressed by how you always try to make time in your busy calendar for a very-short-notice talk, even giving up your rest when a 9-hour time difference separated us, something I will never forget and for which I will be forever thankful. Your dedication and forward vision are truly exceptional and inspiring, and I am convinced they will make your transition into industry a success. I wish you all the best, always cheering for you.
To **Stephanie**, my co-promotor. Being part of the QNodeOS project is one of the biggest highlights of my PhD, and thank you for never stopping believing in it when the lab seemed hopeless. Your energy and determination are truly motivating and contagious, turning everything you set your mind to into reality. I am looking forward to seeing what your next achievement will be.
I want to thank the rest of my doctoral committee for accepting the invitation and taking the time to read my thesis, which I am looking forward to discussing with you at my defence: **Miriam Blaauboer**, **Leo DiCarlo**, **Hannes Bernien**, **Eleni Diamanti**, **Giordano Scappucci**. An extra thank you to **Miriam** and **Leo**: being your teaching assistant for FQI was a real pleasure. Also, thank you to **Hannes**, building my experiments on your PhD legacy and showing you how much the lab has changed was an honor.
My defence would not be drama-free without my paranymphs (aka damigelle), **Nina** and **Caroline**. Thank you for being by my side at this crucial time. **Nina**, we have had a connection since the day we met at my interview, when I was immediately struck by your determination and passion for your cleanroom work. Despite working on completely opposite projects, we were always there for each other, sharing the joys and tears of these incredible four years. I could always count on you when something was on my mind, and I always appreciated your honesty and transparency. Your attention to detail, both at work and in everyday life, makes you a precious gem (like a diamond), and I know everyone feels lucky to have you around. Near or far, I am truly excited to see what comes next for you, and I wish you all the best.
Caroline, we formally met during the lab tour I gave for FQI, and I was impressed by your sharp and timely questions. We grew much closer in the last year, when we traveled together around California and discovered we share the same tastes, from unhinged and overpriced smoothies to reformer pilates classes, all of it while going for breathtaking (literally and figuratively) hikes in the national parks. That trip also left me impressed

by your crisis management and organizational skills, which proved crucial in many situations. At work, I admire the energy you put into your master's thesis project and the passion and confidence you now bring to Delft Networks. I know you will achieve great things, and I wish you all the best.

In these four years, many other people have had a huge impact on me and deserve a special mention. **Kian**, the first local who truly made me feel welcome in the Netherlands. Your friendship and support have helped me tremendously throughout the years, knowing I can always count on you. Even from Canada, I feel you very close through early-morning/late-night texts and long video calls (when we remember to properly schedule them!). I can openly share anything on my mind, especially when it comes to fashion ~~sense~~ crimes, which we both know are crucial for science communication. On that note, I really enjoyed giving talks and interviews with you; I feel I learned a lot by watching your confidence and savoir-faire under the spotlight. Our road trips in the US are among my best memories, and thank you for always checking the room (bug) situation for all of us. During those trips, your business mindset always emerged, setting certain *vibez*^{TM}. For me, that was a clear sign that your skills and determination will turn your ambitions into reality, and if there were a stock for that, I would certainly have added it to my portfolio.

Marie-Christine, you took me under your wing and taught me so much during my first years, laying the foundation for all my achievements. Your patience, rigor, enthusiasm, and determination to truly understand things are deeply inspiring. You had the remarkable ability to make even the most tedious tasks enjoyable, turning every bug or unexpected setup behavior into a valuable learning opportunity. Those long days in LT5, when we completely lost track of time and barely saw sunlight during the winter, remain, despite everything, among the happiest memories of my PhD. At the same time, you were organizing your dreamy Vienna wedding and sharing details during our coffee breaks. I am very grateful to have been part of it. To you, **Josh**, and **Alexander**, I wish you all the joy and love possible.

A PhD is much harder without a welcoming group around you. I was very lucky to be surrounded by the most inspiring and kind people, whom I would like to thank one by one.

The Cavity Boys, **Julius** and **Yanik**, trusted companions for Friday night outings and *Feuerzangenbowle*: sharing the PhD journey with you was a real pleasure. **Julius**, your passion and commitment to science go well beyond a stereotypical German attitude. You are always ready to help anyone in need, from lab-related issues to organizing fun team-building activities. Delft Networks will benefit enormously from your spirit. **Yanik**, your lab-management skills and deep knowledge of interesting research papers and ground-breaking documentaries never cease to amaze me. Always bring your contagious smile and questionable *Italian-wannabe* band music taste with you, and never forget to place a Team Diamond sticker along the way. **Arian**, the person I always turned to when something weird happened in the lab. You carry the Team Diamond experience and strong opinions, and conversations with you are always stimulating, regardless of the topic. You are also incredibly fun to have around, thanks for all the laughs during our road trips. Thank you as well for giving me the opportunity to be your paranymph, and I wish you all the success in Singapore. The OG Networkies, **Matteo Sr.** and **Sophie**. **Matteo Sr.**, we incredibly share a similar scientific path before we even met, from our studies to

our PhD paths, which I have always found fascinating. Thank you for believing in me after visiting the lab in Rome during my Master's thesis. I wish you and Grace all the best in Boston. **Sophie**, from the early days of learning from you about the complexity of the Networks setup to seeing you return as a Group Leader, I have always admired your confidence and passion for science. Good luck with your new adventure, I am always rooting for you. To the half-Networkies, half-Tin-Team member **Hans**: your passion and broad knowledge are a guarantee in any situation, and I always felt safe knowing I could count on your prompt answers. I am very happy you, **Sophie** and little **Hannah** are part of Christopher's life and mine. To the other postdoc in my life, **Alejandro**: your time here was an *odi et amo*, with the Networks setup presenting all sorts of ~~problems~~challenges, but also the excitement of knowing great things would come once the storm passed. Thanks for staying strong during that difficult time. Our conversations about life and about were a great way to forget the struggle for a while. I hope bank life is treating you better than a leaky cryostat. **Lorenzo**, the Italian postdoc: our overlap was short, but long enough to attend a conference together in Torino and have fun. All the best for your position in France. **Alex**, the new postdoc: I am truly impressed by your very, very long bike rides, and I will keep following your journeys on the tracker. Thank you for the lovely dinners discovering The Hague's restaurants, and good luck with your next project in VLT1! Speaking of VLT1, I wish the most exciting and successful PhD to you, **Timo**. It is impressive how quickly you became confident with such a complex setup.
Luckily, I was not always confined to the lab or in front of a PC, and I had the chance to share the B001 office with two wonderful group members. From day one, **Matteo Jr.**, you were a constant presence both during the week and on weekends. Thank you for the office chats, Sunday lunches, and jamming sessions (despite me being really bad at playing instruments!). Your creativity, especially in science, will take you very far, I am sure of it. **Leo**, you have been around for so long, and yet you are still so early in your PhD (and in B001). It is amazing how quickly and confidently you mastered cleanroom magic. Good luck with all your projects, and thank you for the fun time in California together with **Niv**. **Niv**, thank you as well for sharing a boogie hotel room with me. You are truly brave to switch from theory to experiment, keep up the enthusiasm, I am sure you will achieve exciting results. To the soon-to-be seniors, best of luck to **Christian** and **Tim**. **Christian**, defects in silicon may be a tough beast to tame, but seeing how consistently on top of everything you are is a clear sign that the project is in excellent hands. **Tim**, you can be very proud of your results so far, and I am sure that with your creativity and determination, many more will follow. **Julia**, thank you especially for joining me on a visit to the MET Museum in New York, and best of luck with your next step, both professionally and personally, your energy will make you shine. **Dani**, you are truly special and make the group so much more fun. Your determination and passion for understanding every detail of your work are admirable. Best of luck with your projects!
HansonLab is also home to many MSc and BSc students who work extremely hard and achieve great results. I had the opportunity to supervise **Constantijn** for his MSc project and **Zeno** for his BSc project. It was wonderful to see how you mastered your projects. I also learned a lot from you, and I hope I was able to pass on some of my enthusiasm and excitement for science. I wish you both every success in your careers. I would also like to thank all the other students and wish them the best of luck: **Leanne**, **Mark**, **Jonas**,

Alexandre, **Miguel**, **Cees**, **Joan**, **Pepijn**, **Stijn**, **Elvis**, **Zarije**, **Colin**, **Sezer**, **Sarel**, **Otmar**. A special mention to **Cees**: your roasting Sinterklaas poem remains one of my all-time favorites. And to **Stijn**, you truly have a natural talent for barbecuing, besides whispering to fibers. Thanks to the research assistants I had the chance to overlap with: **Noé**, apart from your precious software skills, I really appreciated iced rosé wine during the boat trip to Rotterdam. **Marta**, I am really excited to see how you will also master experimental physics!

Team Diamond is actually a much bigger family, and I would like to start by thanking **Tim Taminiau** and his group. **Tim**, having you co-supervise the Networks team was truly invaluable. Your perspective and expertise in nuclear spin control consistently brought out the best in the projects and strongly motivated me to improve. Good luck with your next project, and with making SiC even more successful than NV. **Nicholas**, we held on tightly through the highs and lows of the Networks setup until its very last day. You pulled through, and together we managed to achieve one more great result with the link efficiency experiment. All the best with finishing your thesis and with your next career move! The same holds for **Benjamin**. Special thanks also to **Jiwon**: your help in answering many questions for Zeno's machine-learning project and joining the Networks team in its final phase made a huge impact. It was truly a pleasure to have you around, and I really enjoyed our casual chats about Korean TV series and culture. Best of luck with your next step. I also had the pleasure of sharing the office with **Gerben** and **Madhu**. **Gerben**, it was impressive to see the quality of your cavities and finally understand the differences among your very precise naming schemes. I wish you much success in finishing your PhD. **Madhu**, already in your first months you showed remarkable dedication and determination. I am sure you will have a brilliant PhD. **Laurens**, we met during my interview, when you were about to conclude your cavity MSc project, and I was immediately impressed by your energy. You should be very proud of your dedication to science and its communication. All the best to you as well in finishing your PhD. **Margriet**, your determination will take you very far, both in science and on a climbing wall. Good luck with everything, I know you will make it to the top of El Capitan. **Christina**, your kindness and altruism are truly precious. You, **Dan**, and **Madhu** should be very proud of your outstanding results, demonstrating a compelling application of NV centers. **Tobi**, thank you for showing Christopher and me some hidden gems in Tokyo. Best of luck to you and **Jasper** with the SiC project. **Kilian**, thank you for joining the German crew every Friday night. Your expertise in optics and photonics is truly invaluable. All the best for your next step. To the TaminiauLab alumni with whom I had the opportunity to share part of my journey: **Conor**, you graduated during my first week of PhD and returned to QuTech during my final year. It was wonderful to get to know you better, talking with you is always a pleasure, regardless of the topic. Your scientific talent and exceptional management skills will undoubtedly make Delft Networks a great success. **Mohamed**, we worked together shortly after your graduation on the machine-learning code for nuclear spin detection. I had a lot of fun trying to understand and run that code with you. All the best to you in the US. **Sjoerd**, thank you for teaching me how to properly enjoy my first Uitje. You will do great at Delft Networks! **Hans** (Bartling), we somehow managed to have an unspoken standing appointment at AH every Sunday. Best of luck at

TNO! **Guido**, I unfortunately missed the sea-elephant tour you organized in San Francisco, but I wish you all the best at Stanford. **Anta**, we met during your PhD in the Tittel lab while sharing the 795 nm laser. I was truly impressed by your patience back then. It is always a pleasure to meet you at conferences, best of luck in Barcelona.
It is wonderful to see Team Diamond growing with new brilliant minds that stimulate exciting scientific discussions. Thanks to **Carlos Errando-Herranz** and his group, **Jan**,**Elena** and **Matteo**. **Jan**, your motivation and energy, both for science and social activities, is really impressive. I know for sure you will have a brilliant PhD! **Lin**, your knowledge and expertise are a great resource. Congrats on your grant and best of luck!
During my PhD, I had the opportunity to collaborate with many talented people. I would like to thank everyone involved in the QNodeOS collaboration, with a special mention to **Carlo** and **Bart**. I will never forget our endless debugging sessions, you can be very proud of how you pulled through. I would also like to thank **Ingmar te Raa-Derckx**, **Wojciech Kozlowski**, **Przemysław Pawełczak**, and **Tracy Northup** for their support and innovative ideas.
For the interface project, I collaborated with **Wolfgang Tittel** and his group, and I would like to thank all the authors not yet mentioned: **Tanmoy**, **Matthew**, **Elsie**, **Gustavo**, **Mariya**, and **Nir**. I also had many stimulating scientific discussions with members of the Wehner Group, whom I would like to thank and wish all the best for the future: **Scarlett**, **Thomas**, **Tzula**, **Janice**, **Soubhadra**, and **Luca**. A special mention to **Scarlett** for our non-work-related conversations, it is very nice to have you around.
Being part of **QIA** gave me the opportunity to work at the forefront of Quantum Internet development while meeting many people beyond the QNodeOS project. **Nico**, thank you for your patience when monthly updates took longer than expected, best of luck with the QNE project! **Jeremiah**, thank you for the opportunity to join you in Copenhagen; I truly enjoyed our conversations. **Arianna**, you did a wonderful job with the QNodeOS press release, I now follow all your adventures at ESA on Instagram. Best of luck! I also collaborated closely with QBlox, especially with **Fokko**, **Mafalda**, and **Vatshal**, thank you for showing me the industry side of science and best of luck with your next projects!
Being a teaching assistant for FQI also gave me the chance to meet inspiring people: thank you **Sean**, **Ruggero**, **Tim** (Vroomans), and **Yuejie**. Correcting homework and exams over sushi or pizza dinners with you was a great motivation to keep going. A special thanks to **Tim** for showing me your work and discussing the similarities and differences between our platforms.
QuTech is filled with incredibly knowledgeable people who elevate scientific work to another level. Leaky cryostats were common during these four years, but nothing is impossible with the amazing cryo engineers **Olaf** and **Jason**, thank you for your support. I would also like to thank the software engineers who worked behind the scenes to keep our codebase robust and were always ready to help when deep expertise was needed: **Henri**, **Pieter**, **Peter**, and **Ravi**. Labs are safe spaces thanks to the careful work of **Régis** and **Hitham**. **Hitham**, thank you also for your efforts in making QuTech a socially safe environment. Thanks to **Roy**, **Siebe**, **Jelle**, and **Vinod** for their support during lab chaos. I am also grateful to the electronics engineers **Raymond S.**, **Raymond V.**, and **Roy**, your knowledge and passion were invaluable on many occasions. Organization runs smoothly thanks to our management assistants, especially **Chantal**, **Esther**, and

Sara. I was able to participate in many outreach activities thanks to the communication team: **Erik**, **Maarten**, **Aldo**, and **Leonie**. Thanks also to the 2022 Uitje Committee: **Liza**, **Maxim**, **Christopher**, **Figen**, and **Siddhart**, it was really fun to organize Uitje with you, and you should be very proud of yourselves for how it turned out.
I had the pleasure of meeting many other amazing people here at QuTech, who are always up for a nice chat or having fun at QuTech events: **Alberto**, **Davide**, **Pablo**, **Hanifa**, **Maia**, **Desa**, **Nicola**, **Francisco**, **Ravisankar**, **Irene**, **Brennan**, **Fenglei**, **Vicky**, and **Sasha**. I would also like to thank all the scientists I met at conferences around the world for the stimulating discussions we shared.
Outside work, I truly enjoyed taking ballet classes, and I would like to thank my ballet teacher, **Renée Dircks**, and the Level 1+ group, especially those I had the pleasure of sharing the stage with: **Margrete**, **Reanne**, **Lili**, **Kaili**, **Arishta**, and **Marianna**. My friends from home were also an important part of this journey. Even though we may not be in touch every day, those moments we shared had a profoundly beneficent impact on me. **Gimmy**, we met in high school thanks to your exceptionally brilliant mind and outstanding passion for physics. Many years later, we unexpectedly found ourselves both pursuing a PhD in the Netherlands, and I am truly grateful that we could spend time together and explore this incredible country. Our canoeing experience in Utrecht (and in some other villages far from there) remains one of my most precious memories. I wish you all the best for your Princeton adventure, and I look forward to hearing about your amazing trips around the world. **Alessandro**, from being my compagno di banco to becoming a designer I truly admire, you have grown tremendously. Thank you for always supporting me and for being there whenever I am back in town. I am sure the future holds something exciting for you. **Morena** and **Martha**, friends since high school, I am really glad I can count on you after so many years. It is always nice to catch up with you. I wish you all the best, and you can be very proud of your strength and determination.

Tutto questo non sarebbe mai stato possibile senza il supporto profondo e costante della mia famiglia. Ai **miei genitori**, grazie per avermi lasciato la libertà di scegliere la mia strada, anche quando questo significava andare lontano. Grazie per la fiducia silenziosa, per i vostri consigli sempre sinceri, e per esserci sempre stati, senza condizioni, nei momenti in cui avevo bisogno di sentirmi a casa.
A mia sorella **Elisa**, la mia migliore amica. Questo viaggio è iniziato insieme a Roma e ci sei sempre stata anche solo con una videochiamata nel buio del lab. Sono fiera della donna che sei diventata e della scienziata che diventerai. Come un poeta una volta disse: *A me va bene anche distanti, tanto ti porto con me.*
To **Christopher**, the sweet and lovely surprise of this Delft adventure. *You dug me out of my grave and saved my heart from the fate of Ophelia.* I cannot imagine my life without you, you have been my rock, my best supporter, my new home. I am really excited to see what the future holds for us. Ich liebe dich.

Curriculum Vitæ

Mariagrazia Iuliano

Born in Benevento, Italy on April 26, 1997

Education

2021 – 2026	PhD in Physics Delft University of Technology, the Netherlands *Thesis*: Quantum Internet: a step closer: Demonstrations and applications with diamond qubits *Promotor*: Prof. dr. ir. R. Hanson
2019–2021	MSc in Physics, *cum laude* Sapienza University of Rome, Italy *Thesis*: Boson Sampling in a reconfigurable continuously-coupled 3D photonic circuit *Advisor*: Prof. F. Sciarrino
2016–2019	BSc in Physics, *cum laude* Sapienza University of Rome, Italy *Thesis*: Quantum Key Distribution via satellite communication *Advisor*: Prof. F. Sciarrino
2011–2016	High school, *cum laude* Liceo Scientifico "G. Rummo", Benevento, Italy

List of Publications

7. **M. Iuliano**, N. Demetriou, H.B. van Ommen, C. Karels, T.H. Taminiau, R. Hanson, *Unconditionally teleported quantum gates between remote solid-state qubit registers*, arXiv:2601.04848

6. J.M. Brevoord, L.G.C. Wienhoven, N. Codreanu, T. Ishiguro, E. van Leeuwen, **M. Iuliano**, L. De Santis, C. Waas, H.K.C. Beukers, T. Turan, C. Errando-Herranz, K. Kawaguchi, R. Hanson, *Large-range tuning and stabilization of the optical transition of diamond tin-vacancy centers by in situ strain control*, Appl. Phys. Lett. 126, 174001 (2025)

5. H.B. van Ommen*, G.L. Van De Stolpe*, N. Demetriou, H.K.C. Beukers, J. Yun, T.R.J. Fortuin, **M. Iuliano**, A. R.-P. Montblanch, R. Hanson, T.H. Taminiau, *Improved electron-nuclear quantum gates for spin sensing and control*, PRX Quantum 6, 020309 (2025).

4. H.K.C. Beukers*, C. Waas*, M. Pasini, H.B. van Ommen, Z. Ademi, **M. Iuliano**, N. Codreanu, J.M. Brevoord, T. Turan, T.H. Taminiau, R. Hanson, *Control of solid-state nuclear spin qubits using an electron spin-1/2*, Phys. Rev. X 15, 021011 (2025).

3. C. Delle Donne*, **M. Iuliano***, B. van der Vecht*, G. M. Ferreira, H. Jirovská, T. J. W. van der Steenhoven, A. Dahlberg, M. Skrzypczyk, D. Fioretto, M. Teller, P. Filippov, A. R.-P. Montblanch, J. Fischer, H. B. van Ommen, N. Demetriou, D. Leichtle, L. Music, H. Ollivier, I. te Raa, W. Kozlowski, T. H. Taminiau, P. Pawełczak, T. E. Northup, R. Hanson, S. Wehner, *An operating system for executing applications on quantum network nodes*, Nature volume 639, 321–328 (2025).

2. **M. Iuliano***, M.-C. Slater*, A.J. Stolk, M.J. Weaver, T. Chakraborty, E. Loukiantchenko, G. C. do Amaral, N. Alfasi, M.O. Sholkina, W. Tittel, R. Hanson, *Qubit teleportation between a memory-compatible photonic time-bin qubit and a solid-state quantum network node*, npj Quantum Inf. 10, 107 (2024).

1. F. Hoch*, S. Piacentini*, T. Giordani, Z.-N. Tian, **M. Iuliano**, C. Esposito, A. Camillini, G. Carvacho, F. Ceccarelli, N. Spagnolo, A. Crespi, R. Osellame, F. Sciarrino, *Reconfigurable continuously-coupled 3D photonic circuit for boson sampling experiments*, npj Quantum Inf. 8, 55 (2022)

Patents

1. H.B. van Ommen, N. Demetriou, T.H. Taminiau, **M. Iuliano**, A.R.-P. Montblanch, R. Hanson, *Quantum Network unit, method and system*, WO2025206952A1

* equally contributing authors

ISBN 978-94-6518-233-9