A Nonblocking Multistage Switching Network for Distributed Quantum Computing

Yu Liu, Yingling Mao, Xu Xu, Xiaojun Shang, *Member, IEEE*, Fan Ye, *Fellow, IEEE*, and Yuanyuan Yang, *Life Fellow, IEEE*

***Abstract*— Quantum computing, utilizing the unique properties of quantum mechanics, has the potential to revolutionize various fields. However, current quantum processors face challenges in scaling the number of qubits, limiting their practical applications. In response, Distributed Quantum Computing (DQC) has emerged as a promising paradigm where multiple interconnected Quantum Processing Units (QPUs) collaborate to execute quantum circuits. In this paper, we focus on designing networks to interconnect QPUs for the implementation of DQC. We find that in real-world experiments and systems, the photon collection and coupling efficiency is low, leading to significant performance degradation in direct connection networks. To address this limitation, we propose a novel multistage switching network tailored for DQC, which has low system complexity and high entanglement generation rates. The proposed switching network comprises $\log_2(N)$ stages and $N/2$ binary switches at each stage, where N represents the number of QPUs. We prove that the proposed network is nonblocking and develop an efficient routing algorithm with a time complexity of $\mathcal{O}(N\log(N))$. Additionally, we show the success probability of entanglement generation in the proposed switching network. Extensive simulations demonstrate that our network significantly outperforms the highly efficient circuit-switching Beneš network and three direct connection networks.**

***Index Terms*— Quantum computing, quantum networks, and interconnection networks.**

I. Introduction

QUANTUM computing leverages quantum mechanical phenomena such as quantum entanglement and superposition to perform computation. Quantum computers offer significant advantages in performing some specialized tasks that classical computers are unable to solve within a feasible timeframe. For example, they can factorize large integers [1], perform approximate optimization [2], and execute Gaussian boson sampling [3] with substantially greater efficiency. The practical applications of quantum computing are far-reaching, with the potential to address global challenges. For instance, by solving large integer factorization problems, quantum computers could break certain encryption schemes, and quantum simulations and machine learning algorithms could facilitate drug discovery processes [4]. Numerous entities are making significant strides in the development of quantum computers. For example, Google has unveiled 'Sycamore' [5], a quantum computer with 70 qubits, and IBM has introduced 'Osprey', a quantum processor equipped with 433 qubits [6].

Despite the significant development of quantum computers, current quantum processors still fall short of the capabilities for real-world applications. For example, it is estimated that breaking RSA-2048 encryption may require millions of physical qubits [7]. However, it is challenging to build a large-scale quantum processor with sufficient physical qubits due to various reasons, e.g., qubits interaction, resource requirement, fabrication, and control challenges [8]. Until now, no quantum computing platform has achieved scalable expansion of qubit numbers without compromising performance or incurring disproportionate costs, energy consumption, or footprint [9].

A promising approach to building large-scale computing systems capable of supporting real-world applications is the Distributed Quantum Computing (DQC) paradigm. Under this paradigm, multiple interconnected Quantum Processing Units (QPUs) collaborate to execute quantum circuits, which each QPU cannot execute individually [10], [11]. There are various platforms for constructing quantum processors, including superconducting qubits [6], trapped ions [9], nitrogen-vacancy (NV) centers [12], and neutral atom [13]. This paper focuses on trapped ion quantum computers for several reasons. First, trapped ion quantum processors have a high quantum volume [14], a result of their long coherence time, high gate fidelity, and high qubit connectivity [15]. As of June 25, 2023, the top three quantum processors, in terms of quantum volume, are based on trapped-ion technology. Second, it is possible to establish remote entanglement between two distinct trapped ion-based QPUs at a relatively high rate [16]. Specifically, under the trapped ion platform, two QPUs can be connected by linking them with the same Bell State Analyzer (BSA) using optical fibers (as shown in Fig. 4). If two QPUs are connected to the same BSA, a shared Bell state can be generated between them, which is then used for performing remote quantum gate operations (see Section III for details). It is worth noting that the results of this paper could also be directly applied to other platforms such as NV centers and neutral atom quantum platforms.

Received 29 March 2024; revised 12 November 2024; accepted 27 January 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor D. Elkouss. Date of publication 27 February 2025; date of current version 20 August 2025. This work was supported in part by the National Science Foundation under Grant 2231040 and Grant 1191278. *(Yu Liu and Yingling Mao contributed equally to this work.) (Corresponding author: Yu Liu.)*

Yu Liu is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, and also with the Department of ECE, Stony Brook University, Stony Brook, NY 11790 USA (e-mail: yu-y.liu@polyu.edu.hk).

Yingling Mao, Xu Xu, Fan Ye, and Yuanyuan Yang are with the Department of ECE, Stony Brook University, Stony Brook, NY 11790 USA (e-mail: yingling.mao@stonybrook.edu; xu.xu@stonybrook.edu; fan.ye@stonybrook.edu; yuanyuan.yang@stonybrook.edu).

Xiaojun Shang is with the Department of CSE, The University of Texas at Arlington, Arlington, TX 76019 USA (e-mail: xiaojun.shang@uta.edu).

Digital Object Identifier 10.1109/TON.2025.3544319

In this paper, we focus on designing networks to interconnect QPUs for the implementation of DQC. There are two types of networks: direct connection networks (static networks) and switching networks. Direct connection networks such as line, ring, and grid networks have static connections between neighboring nodes. In a static network, when two non-neighboring QPUs request a shared Bell state, routes between them must first be selected. Subsequently, a shared Bell state is generated on each link along the chosen routes, followed by the execution of quantum swapping at the intermediate QPUs. In contrast, switching networks offer dynamic connections. By carefully configuring the switches based on the requirements of the QPUs, QPUs that request a shared Bell state can be connected directly to the same BSA, thereby establishing a link between them. Switching networks present two primary advantages over static networks. First, QPUs in static networks require multiple photon collection modules and associated control modules, increasing the system complexity and posing significant implementation challenges. Second, static networks suffer from low entanglement generation success rates due to the low photon collection and coupling efficiency experienced in real-world systems, a problem that our switching networks effectively mitigate.

Given the advantages of switching networks, this paper is centered on designing switching networks and the corresponding algorithms to enhance the efficiency of DQC systems. While some existing switching networks, such as the Beneš network, could potentially be adapted to interconnect QPUs for implementing DQC, their direct application is unsuitable due to their high cumulative insertion loss. This issue is particularly detrimental to the fragile photonic links between QPUs. Consequently, there is a clear necessity to design switching networks that are specifically tailored for DQC. Unlike traditional circuit-switching networks that establish direct communication paths between sources and destinations, our DQC network demands a distinct architecture. Specifically, two QPUs must be linked to the same BSA to generate a shared Bell state. Our goal is to devise a network capable of simultaneously accommodating all potential entanglement requests from a collection of distributed QPUs.

This paper contributes to DQC in several significant ways. First, we propose a multistage switching network specifically designed for DQC and a corresponding routing algorithm. This network interconnects N QPUs and $N/2$ BSAs via $\log_2(N)$ stages of switches. Secondly, we prove that the proposed switching network is nonblocking and can simultaneously support all possible quantum entanglement requests from the connected QPUs. In addition, we devise an efficient routing algorithm for the network, which operates with a time complexity of $\mathcal{O}(N\log(N))$. Lastly, we evaluate the performance of our proposed switching network through extensive simulations under real-world parameters. Our results indicate that the proposed switching network has a significant performance advantage over existing networks, including the traditional Beneš network and various static networks. Specifically, the average time for successful entanglement generation in our network is significantly faster than that in the Beneš, grid, ring, and line networks.

The remainder of this paper is organized as follows. Section II provides an overview of related works. Section III discusses trapped ion-based distributed quantum computing. Section IV shows the limitations of static networks. Section V presents our proposed network design along with a dedicated algorithm. Section VI presents the simulation results. Finally, Section VII concludes the paper.

II. Related Work

DQC is recognized as a promising paradigm for supporting large-scale quantum circuits [10]. A central challenge of DQC is establishing entanglement between QPUs [17], [18]. Shared Bell states are required to perform remote quantum gates on qubits located in different QPUs. There have been several studies dedicated to link layer entanglement generation using various quantum technologies [12], [16], [19], [20]. Notably, Stephenson et al. demonstrated generating Bell states of trapped-ion qubits at an average rate of 182 Hz [16]. In [20], Hannegan et al. introduced a networking architecture leveraging neutral-atom-based nondestructive single-photon detection and single-photon storage to improve entanglement rates in quantum networks based on trapped ions. Simulation results based on experimental parameters showed that the proposed architecture can significantly increase the remote entanglement generation rates.

Moreover, there has been research focusing on generating remote entanglement between two non-directly connected nodes via entanglement swapping in the context of the quantum internet [21], [22], [23], [24], [25], [26]. In [24], Pant et al. considered the entanglement routing problem in the quantum internet and proposed a routing algorithm allowing multiple quantum processor pairs to generate entanglement simultaneously. In [23], Shi et al. studied the entanglement routing problem for concurrent entanglement request pairs and arbitrary network topologies and introduced an entanglement routing algorithm tailored to the unique properties of quantum networks. In [22], Farahbakhsh et al. designed an opportunistic entanglement routing algorithm for the quantum internet and showed that the opportunistic approach outperforms conventional approaches. However, these works focused on static networks. In contrast, this paper underscores the advantages of switching networks for DQC and, therefore, focuses on switching networks.

Switching networks such as the Beneš, Omega, and Banyan networks have been widely studied in the fields of telecommunications, data center networks, and network-on-chip systems, primarily for establishing direct connections between source and destination nodes. The Beneš network is a non-blocking network with $2\log_2(N)$-1 stages, ensuring connections between N inputs and their corresponding outputs without blocking or contention [27]. The Omega [28] and Banyan [29] networks are a blocking switching network with $\log(N)$ stages. In this paper, our goal is to generate shared Bell states between QPUs. To achieve this, we need to connect both QPUs to the same BSA. The Beneš network can be adapted to DQC by treating the QPUs as inputs and the BSAs as outputs. However, this adaptation leads to a low entanglement generation rate, as the photon loss probability

in switches increases exponentially with the number of switch stages. As for the Omega network, it is inherently blocking in the DQC scenario we are examining.

There are also some prominent works related to using switches for DQC. In [30], Duan et al. proposed a hierarchical approach to interconnecting trapped ion registers and photon detectors using a switch network, but they did not provide a specific switching network topology. In [31], Dong et al. experimentally demonstrated an 8-input Mach-Zehnder mesh network for remote entanglement generation, showcasing the feasibility of interconnecting QPUs using a switching network. However, the paper primarily focused on an 8-input system and did not present the routing algorithm and its performance for an arbitrary number of QPUs. In [32], Bartolucci et al. proposed switch networks for single photonic fusion-based quantum computers, while this paper focuses on interconnecting multiple quantum processors.

III. Trapped Ion-Based Distributed Quantum Computing

Among several suitable platforms for constructing quantum computers, such as superconducting qubits, NV centers, and topological qubits, trapped ion-based processors stand out for their long coherence time, excellent qubit connectivity, and high gate fidelity. For instance, the trapped ion-based H1-2 Quantinuum device demonstrates significant improvements over the superconducting qubit-based ibmq_mumbai device. Specifically, the single-qubit gate, two-qubit gate, and readout error of the H1-2 Quantinuum device are 2x, 12.9x, and 7.2x smaller, respectively, compared to those of the ibmq_mumbai device [33]. Quantum volume is a well-accepted metric originally proposed by IBM [14] to measure the power of near-term quantum computers. Quantum volume, denoted by V_Q, is defined as

$$V_Q = \min\left[N, 1/(N\epsilon_{\text{eff}})\right]^2.$$

In this equation, N represents the number of qubits, and ϵ_{eff} denotes the average error rate of a two-qubit gate. So far, the top three quantum processors in terms of quantum volume were based on trapped ion qubits. Despite having a relatively large number of qubits, superconducting quantum computers often demonstrate lower quantum volumes than their trapped ion-based counterparts for two primary reasons. Firstly, the gate fidelity of trapped ion-based quantum computers generally surpasses that of superconducting quantum computers. Secondly, trapped ion-based quantum computers exhibit excellent qubit connectivity, allowing for efficient execution of quantum gate operations. Notably, quantum gate operations can only be performed on connected qubits. In the case of the IBM-Melbourne superconducting platform, as shown in Fig. 1.a, additional entanglement swapping gates are required to perform a quantum gate operation on two non-neighboring qubits, which leads to an increase in the average error rate of two-qubit gates, i.e., ϵ_{eff}. In contrast, the qubit connectivity of trapped ion-based quantum processors follows a mesh topology as shown in Fig. 1.b. Consequently, this paper focuses on trapped ion-based QPUs.

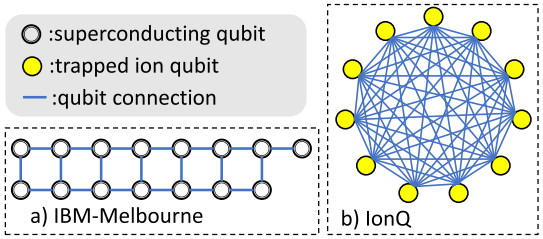


Fig. 1. Qubit connectivity comparison.

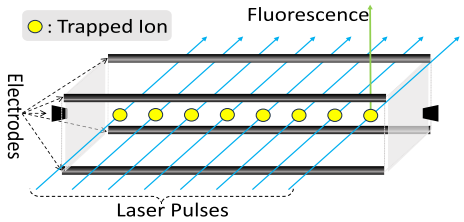


Fig. 2. Trapped Ion-based processor.

A. Trapped Ion Quantum Computer

Next, we will provide a brief introduction to the trapped ion-based quantum computer. Under the trapped ion platform, the electronic energy levels represent the states of each trapped ion-based qubit, with the ions being trapped by an oscillating radio-frequency electric field [9]. We can trap a number of ions, typically located in a line as depicted in Fig. 2. There may be other topologies for the trapped ions, but they may raise additional difficulties, such as introducing extra decoherence sources. The trapped ion qubits exhibit long coherence times, e.g., ranging from 0.2s to around 600s, partially depending on the energy levels used to represent the basis states [9].

To readout qubits, a laser beam with a particular frequency is applied to the ion from one direction, as depicted in Fig. 2. The ion will fluoresce if it is in state $|0\rangle$, and the ion will not interact with the laser if it is in state $|1\rangle$. Then we detect photons in a direction orthogonal to the laser beam. If photons are detected, the ion is in state $|0\rangle$, and in state $|1\rangle$ otherwise. The readout fidelity is pretty high, e.g., 99.99% [34]. As for single qubit logic gates, we can perform gate operations on a qubit by applying laser or microwave radiation depending on the type of trapped ion used, where the fidelity can be up to 99.9999% [15]. Two-qubit gates, performed by leveraging the Coulomb interaction between ions, can achieve a fidelity up to 99.9% [35]. Much like single-qubit gates, these operations are executed by applying carefully tailored laser pulses to the target ions. This method allows us to perform two-qubit gates between any pair of qubits, thereby establishing all-to-all connectivity as shown in Fig. 1.b. While it is feasible to trap a significant number of ions, the implementation of the required optical and electronic control poses significant challenges. Therefore, DQC becomes a necessity.

B. Remote Quantum Gate

While the DQC paradigm offers advantages in terms of increased qubit capacity, it also introduces the challenge of applying gate operations to qubits located in separate QPUs,

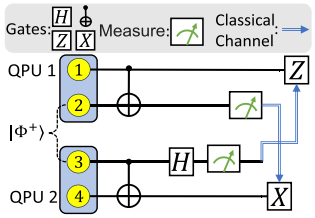


Fig. 3. Remote CNOT gate.

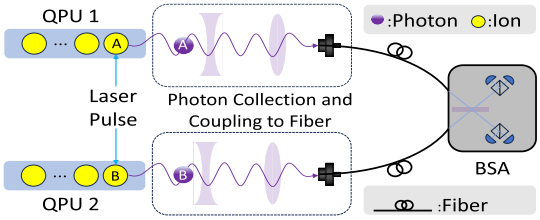


Fig. 4. Herald entanglement generation.

namely remote gates. All quantum circuits can be implemented using single-qubit gates and CNOT gates [36]. Hence, the focus of our discussion will be on the implementation of remote two-qubit CNOT gates. Remote two-qubit quantum gates can be implemented by leveraging a shared Bell state. Fig. 3 depicts the circuit for implementing a remote CNOT gate on two qubits, q_1 and q_4, located in different QPUs, while consuming a shared Bell state, denoted as $|\Phi^+\rangle$. It should be noted that the shared Bell state does not necessarily have to be $|\Phi^+\rangle$, as any of the four Bell states, i.e., $|\Phi^+\rangle$, $|\Phi^-\rangle$, $|\Psi^+\rangle$, and $|\Psi^-\rangle$, can be utilized. Due to space constraints, the proof of this statement is omitted.

C. Herald Entanglement Generation

Since each remote gate on qubits in two QPUs consumes a shared Bell state, we will briefly introduce how to generate entanglement between two trapped ion qubits in different QPUs. The entanglement generation of two trapped ion qubits is illustrated in Fig. 4. The protocol for link-layer entanglement generation is as follows. First, laser pulses from the same source are split and used to simultaneously excite qubit A in QPU 1 and qubit B in QPU 2. If successful, this process will generate two entangled ion-photon pairs, establishing entanglement between ion A and photon A and between ion B and photon B. Subsequently, the emitted photons are collected and coupled into fibers. If the BSA detects the two photons, there are four possible outcome patterns of the same probability, and only two patterns herald a successful Bell's state entanglement generation [16]. That is, if two photons are detected by the BSA, the probability of a successful Bell state generation is 0.5. If the BSA observes the desirable patterns, it sends an acknowledgment signal to the two QPUs, and they can use the shared Bell state to perform remote two-qubit gates. Otherwise, the BSA sends a negative acknowledgment signal to the two QPUs, and they can repeat the entanglement generation process.

We use p_p to represent the probability that each time we excite an ion, it successfully generates an entangled ion-photon pair, and the photon is successfully collected and coupled to the fiber. We use p_f to denote the probability of a photon successfully traversing the fiber and reaching the BSA. Furthermore, p_d is used to represent the overall efficiency of the BSA, specifically, the probability that the detectors of the BSA will successfully detect two photons, given that two photons have reached the BSA. For each entanglement generation attempt, the success probability, denoted by p_a, is as follows:

$$p_a = \frac{1}{2} p_p^2 p_f^2 p_d. \tag{1}$$

D. Scalable Distributed Quantum Computing

To execute DQC, the process starts with the representation of quantum algorithms as quantum circuits, utilizing the quantum gates available to the specific platform being used—in the case of this paper, the trapped ion platform. Each quantum platform can support only certain types of quantum gates, which possess unique parameters such as fidelity and gate operation time [37]. Following this, the logical qubits of the quantum circuit are mapped onto the physical qubits within the QPUs. It is worth noting that to perform quantum error correction, multiple physical qubits may represent a single logical qubit [38]. The above-mentioned two steps can be accomplished synergistically to enhance the overall performance.

Subsequently, quantum gates are applied to the qubits as determined by the quantum circuits. If a gate is applied to qubits located on different QPUs, it consumes a shared Bell state, which is generated through the interconnection network. If two QPUs are directly connected to a BSA, the procedure outlined in Section III-C can be used. However, if they are not directly connected, as in the case of QPU 1 and QPU 3 in Fig. 5, entanglement swapping in intermediate processors becomes necessary. As shown in Fig. 5, to generate a share Bell state between QPU 1 and QPU 3, we initially create two such shared states: one between QPU 1 and QPU 2, and another between QPU 2 and QPU 3. Following this, entanglement swapping is performed on QPU 2. If successful, this results in a shared Bell state between QPU 1 and QPU 3.

The physical layer of distributed quantum computing is depicted in Fig. 6. Utilizing this architecture, we can assemble a set of QPUs into a powerful distributed quantum computing system. Moreover, it is possible to partition the entire quantum computing system into multiple virtual quantum slices. These slices can execute quantum algorithms in parallel, similar to network slicing and virtual machines in classical computing systems.

IV. Performance Limitations of Static Networks: Extreme Case Analysis

In this section, we delve into an analysis of the performance of static networks as employed in quantum internet. In particular, we examine their limitations under low photon collection and coupling efficiency, p_p. While low p_p is encountered in real-world systems, they are often overlooked in studies focusing on the quantum internet. Furthermore, we demonstrate that

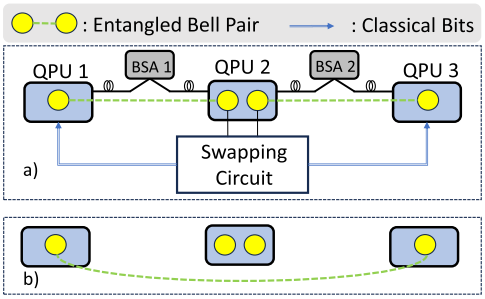


Fig. 5. Entanglement swapping.

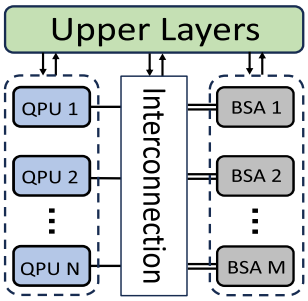


Fig. 6. DQC architecture.

TABLE I
NOTATION TABLE

Symbol	Description
p_p	Probability of successful photon generation, collection, and coupling.
p_f	Probability of a photon successfully traversing the fiber.
p_d	Probability of the BSA detectors successfully detecting two photons.
p_a	Probability of successful entanglement generation in each link-layer attempt.
T_e	Duration of the external phase.
$\gamma_{\max}$	Maximum photon production rate.
n_e	Maximum number of entanglement generation attempts during the external phase.
p_e	Probability of successfully generating a shared Bell on each link between the S-D QPUs in each slot.
p_s	Probability of successful entanglement swapping.
p_r	Probability of successfully generating a shared Bell between the S-D QPUs in each slot.
p_b/p_ϕ	Probability of a bit/phase flip error in shared entangled qubit pairs on each link

entanglement swapping within static networks can result in a degradation of the fidelity of the generated shared qubit pairs. Important notations are listed in Table I.

A. Static Networks

To establish interconnections between QPUs, a common approach employed in the context of quantum internet [23], [24] is to create static links between them. This involves positioning a BSA at the midpoint of each link as seen in the line network example in Fig. 7.a. Mesh topologies, requiring $(N-1)!$ links and N collection modules per QPU, are non-scalable. Hence, non-mesh designs, like grid or circular networks, are used, where entanglement swapping

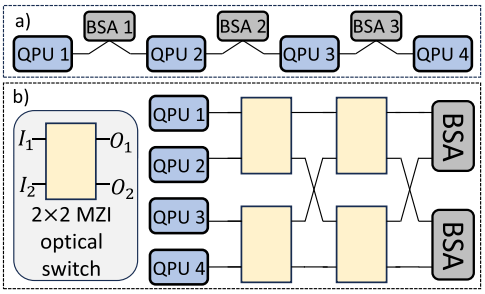


Fig. 7. Static and switching networks.

in intermediate QPUs enables shared Bell state generation on indirectly linked QPUs.

A typical protocol [24] for generating a shared Bell state between two S-D QPUs (Source-Destination QPUs) connected by intermediate QPUs (also known as repeaters in the context of quantum internet) in the static topology is as follows. The system operates in slotted time, where each time slot should be less than the coherence time of the qubits. Each time slot has two phases, namely external and internal phases. First, in the external phase, each link on the routes between the S-D QPUs attempts to generate shared entangled pairs independently using the protocol in Section III-C. Let T_e be the duration of the external phase, and γ_{max} be the maximum photon production rate of each QPU for entanglement generation [20]. Therefore, the maximum number of link-level entanglement generation attempts during the external phase is $n_e \triangleq T_e \cdot \gamma_{max}$. If we successfully generate shared Bell states for all the links on a route between the S-D QPUs, we perform entanglement swapping on each intermediate QPU along the route in the internal phase. We use p_s to denote the success probability of each entanglement swapping process.

Next, we analyze the probability of successfully generating a shared Bell state on a route between the S-D QPUs within a given slot, denoted by p_r. Assume that there are $N-1$ repeaters[1] [23], [24] (intermediate QPUs) in the route between the S-D QPUs. Then, the success probability of a link in the route successfully generating a shared Bell state during the external phase, denoted by p_e, is as follows:

$$p_e = 1 - (1 - p_a)^{n_e}. \tag{2}$$

Here, p_a from (1) represents the success probability of each entanglement generation attempt. Then, the success probability p_r is as follows

$$p_r = (p_e)^N {p_s}^{N-1}. \tag{3}$$

Here, $(p_e)^N$ represents the probability that each link of the N links on the route has successfully generated a shared Bell state, and ${p_s}^{N-1}$ is the probability that the quantum swappings on the $N-1$ intermediate QPUs are successful. Generally speaking, $p_a = 0.5 p_p^2 p_f^2 p_d$ is small. Therefore, to get some intuition about p_r, we express p_r as Taylor series

[1]Quantum repeaters function as specialized quantum processors that employ entanglement swapping to establish long-distance quantum entanglement across a network.

at $p_a = 0$ as follows:

$$p_r = p_s^{N-1}(n_e p_a)^N(1 + o(p_a)) \sim p_s^{N-1}(0.5 n_e p_p^2 p_f^2 p_d)^N. \quad (4)$$

Next, we will discuss why repeaters are favored in the context of quantum internet, while it may not be the best choice for distributed quantum computing. Assume L is the distance between the S-D QPUs, and the $N-1$ intermediate QPUs are equally distributed between these two QPUs. As a result, the distance between each pair of adjacent QPUs is L/N. From [23] and [24], we have, $p_f = e^{-k_f \frac{L}{N}}$, where k_f is the parameter measuring the fiber loss. In the context of the quantum internet, QPUs are typically located far apart, and p_f often becomes the bottleneck, and p_p is not considered in their formulation, i.e., $p_p = 1$ in [23] and [24]. Under this case, by adding $N-1$ repeaters between the S-D QPUs, the success rate p_r is as follows:

$$p_r \sim p_s^{N-1}(0.5 n_e p_d)^N e^{-2k_f L}. \quad (5)$$

From (5), if $0.5 n_e p_d p_s > 1$ which is true in general, the success probability increases as the number of intermediate nodes, N, increases when p_f is close the 0.

Despite the advantage of using intermediate nodes in the context of quantum internet, it may cause performance degeneration in distributed quantum computing. Within the framework of DQC, QPUs are typically positioned in close proximity to each other. For instance, they might be located within the same room, mere meters apart, or even consolidated onto a single board [9]. Additionally, quantum frequency conversion can be utilized to decrease the probability of photon loss in the fiber [20]. Assuming a fiber loss rate of 30 dB/Km, the likelihood of experiencing photon loss in a 10-meter fiber segment is approximately 0.933. Therefore, p_f is no longer the bottleneck in this case. On the other hand, it is difficult to collect the photon emitted by trapped ions and couple it into the fiber [20], [39], [40], [41]. For example, a typical value of p_p is measured to be 0.021 in [41]. From (4), when p_p is sufficiently small, the success rate decreases as N increases. Even if n_e is sufficiently large (assuming infinite coherence time) such that $p_a = p_e = 1$, we find that $p_r = p_s^{N-1}$ still limits the success probability of each time slot. As an illustration, assuming $p_s = 0.9$ as reported in [42], for a network consisting of $N = 16$ hops and assuming an infinitely large n_e, the probability p_r is 0.185.

Furthermore, the process of entanglement swapping on the intermediate QPUs may result in fidelity degradation. The generated shared qubit pair on each link is subject to imperfections, making them susceptible to bit and phase flip errors [43]. Assume we want to establish the state $|\Phi^+\rangle$ along each link. Nonetheless, there is a probability p_b for a bit flip error in the generated qubit pair on each link, where we ignore the phase flip error for simplicity. Consequently, the density matrix representing the shared qubit pair on each link is expressed as:

$$\rho_0 = (1-p_b)|\Phi^+\rangle\langle\Phi^+| + p_b|\Psi^+\rangle\langle\Psi^+|, \quad (6)$$

resulting in a fidelity of $(1-p_b)$. Let us consider the scenario where the objective is to generate a shared qubit pair between source and destination QPUs, interconnected by a line network with $N-1$ intermediate QPUs. Initially, shared entangled qubits are generated on each link, each with a density matrix ρ_0. Following successful entanglement swapping at each of the $N-1$ intermediate QPUs, the resulting end-to-end entanglement between the source and destination QPUs is described by the density matrix

$$\rho_N = f_N|\Phi^+\rangle\langle\Phi^+| + (1-f_N)|\Psi^+\rangle\langle\Psi^+|, \quad (7)$$

where $f_N = 2^{N-1}\left(\frac{1}{2} - p_b\right)^N + \frac{1}{2}$ is the fidelity of ρ_N, with p_b denoting the probability of bit-flip errors per link. The proof of (7) is omitted due to space limitations, and the main idea of the proof is mathematical induction. As N goes to ∞, the fidelity of ρ_N will go to $1/2$, and the generated entanglement becomes useless. In addition, based on (7), we can prove the average fidelity of successfully generating shared Bell states between each pair of QPUs in a line network with N links, denoted as $\mathbb{E}[F(N)]$.

Theorem 1: The average fidelity of successfully generated shared Bell states between each pair of QPUs within a linear static network comprising N+1 QPUs is given by:

$$\mathbb{E}[F(N)] = \frac{1-2p_b}{2Np_b} + \frac{(1-2p_b)\left((1-2p_b)^{N+1}-1\right)}{4N(N+1)p_b^2} + \frac{1}{2}.$$

The proof of Theorem 1 is omitted due to space limitations. As N goes to ∞, the average fidelity $\mathbb{E}[F(N)]$ goes to 0.5.

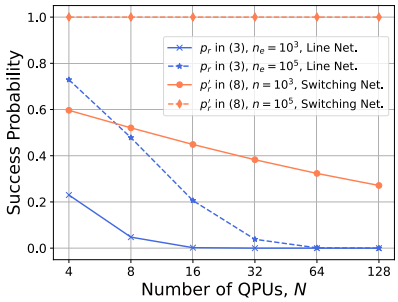


Fig. 8. Success probability vs. number of QPUs.

B. Switching Networks

The second way to interconnect QPUs and BSAs is using optical MZI switches [31]. The success probability of attempting n times using the switching network that we proposed in Section V-A is shown in (10). We express (10) as Taylor series at $p_p = 0$ as follows:

$$p_r'(n) = 0.5 n p_p^2 p_f^2 p_i^{2log_2(N)} p_d + o(p_p^2). \quad (8)$$

In this equation, p_i is a probability related to the insertion loss of switches. As observed in Equation (10), when p_p is small, the success probability in our proposed switching network contains the factor p_p^2, while the success probability of the line network as given in Equation (4) contains the factor p_p^{2N}. Therefore, in the extreme case where p_p is close to 0 and becomes the bottleneck, the proposed switching network exhibits superior performance compared to the static line network.

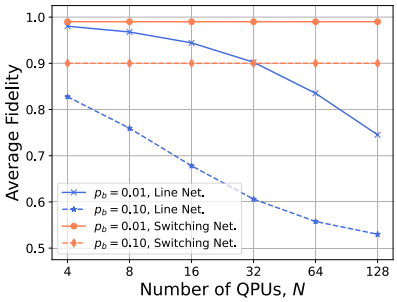


Fig. 9. Average Fidelity vs. number of QPUs.

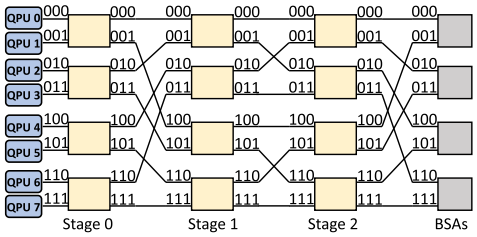


Fig. 10. 8-input QMSN.

Next, we present numerical results comparing the success entanglement generation probabilities in (3) and (10) for the static line network and the proposed switching network, respectively. These results were derived using real-world parameters from experiments. As shown in Fig. 8, the switch network outperforms the static line topology. In addition, assuming that there bit flip error exists, we compare the average fidelity of successfully generated shared Bell states between each pair of QPUs within a linear static network and our proposed switching network. As depicted in Fig. 9, the average fidelity of the shared qubit pairs generated within our proposed switching network exceeds that of the static line network. Note that in Fig. 9, only bit-flip errors are considered, and while switching networks may not affect quantum properties on certain platforms, they may introduce fidelity degradation on others. The analysis in this section is intended to demonstrate the advantages of switching networks over static configurations, and we will investigate the platform-specific fidelity impacts of switches in our future work. Consequently, our proposed switching network demonstrates the capability to produce entangled qubit pairs of superior quality and with a higher probability of success.

V. Switching Network and Algorithm Design

In this section, we design a multistage switching network for DQC and analyze its performance.

A. Multistage Switching Network Design

We consider building a multistage switching network using 2×2 binary switches. Each binary switch has two inputs and two outputs and can operate in two states: Straight and Cross. In the Straight state, input 1 connects to output 1, and input 2 connects to output 2. Conversely, in the Cross state, input 1 links to output 2, and input 2 connects to output 1. We focus on interconnecting N QPUs with $N/2$ BSA. Each QPU has an output, and each BSA has two inputs as shown in Fig. 4.

To facilitate our discussion, we define two functions, $S_l(i, n, N)$ and $S_r(i, n, N)$, which represent the left circular shift and right circular shift, respectively. The function $S_l(i, n, N)$ generates a number by first expressing integer i as a binary number with $\log(N)$ bits, then performing an in-place left circular shift on the last n bits, while the remaining bits outside of the last n are left unchanged. For instance, if N equals 16 and we want to calculate $S_l(6, 3, 16)$, we represent 6 as a $\log(16)$-bit (4-bit) binary number 0110_2. We then apply an in-place left circular shift on the last three bits, resulting in 0101_2, or 5 in decimal notation. Similarly, the $S_r(i, n, N)$ function generates a number by first expressing integer i as a binary number with $\log(N)$ bits. It then applies an in-place right circular shift on the last n bits, leaving the remaining bits outside of the last n unchanged. For example, if we want to calculate $S_r(5, 3, 16)$, we represent 5 as 0101_2. Then we apply an in-place right circular shift on the last three bits, resulting in 0110_2, or 6 in decimal notation.

Next, we present the structure of the proposed multistage switching network, named Quantum Multistage Switching Network (QMSN). Fig. 10 shows an instance of an 8-input QMSN. The N-input QMSN consists of $\log(N)$ stages of switches, and each stage has $N/2$ binary switches. The two inputs of the i-th switch in each stage are labeled by $2i$ and $2i+1$, where $i \in [N/2] \triangleq \{1, 2, \cdots, N/2\}$. Similarly, The two outputs of the i-th switch in each stage are labeled by $2i$ and $2i+1$, where $i \in [N/2]$. Moreover, the inputs of the i-th BSA are labeled by $2i$ and $2i+1$, where $i \in [N/2]$. Next, we illustrate the interconnections between the QPUs, switching stages, and BSAs. For any given $i \in [N]$, QPU i is connected to input i of stage 0. Then, the i-th output of stage 0 is connected to the $S_r\,(i, \log(N), N)$-th input of stage 1. For example, in Fig. 10, outputs $0, 1, \cdots, 7$ of stage 0 are connected to inputs $(S_r(0, 3, 8), S_r(1, 3, 8), \cdots, S_r(7, 3, 8)) = (000_2, 100_2, 001_2, 101_2, 010_2, 110_2, 011_2, 111_2)$ of stage 1, respectively. For each stage $j \in \{1, 2, \cdots, \log(N) - 1\}$, the i-th output connects to the $S_l(i, j+1, N)$-th input of the next stage (or BSA stage if $j = \log(N) - 1$). For instance, in Fig. 10, outputs $(0, 1, \cdots .7)$ of stage 1 are connected to inputs $(S_l(0, 2, 8), S_l(1, 2, 8), \cdots, S_l(7, 2, 8))= (000_2, 010_2, 001_2, 011_2, 100_2, 110_2, 101_2, 111_2)$ of stage 2.

The network we've designed bears similarities to the Beneš and Banyan networks. However, there exists a distinct difference: while the Beneš and Banyan networks aim to connect inputs to corresponding outputs, the QMSN is purposefully designed to route two paired inputs to the exact same BSA for herald entanglement generation. That is, the proposed network has a different purpose from the Beneš and Banyan networks. In addition, the Beneš network with N inputs has $2\log_2(N) - 1$ stages, while the proposed network has $\log_2(N)$ stages. Therefore, compared with the Beneš, the proposed approach has a lower cumulative insertion loss and a high entanglement generation rate. Moreover, the looping routing algorithm for the Beneš network and the self-routing algorithm are not feasible for our proposed network, and we need to propose a dedicated routing algorithm.

As for the placement of laser sources for the proposed QMSN, an additional multistage switching network, e.g., as a reversed QMSN, can be employed to direct laser pulses to the QPUs. $N/2$ laser sources first pass through beam splitters, and then the QMSN routes them to the appropriate QPU pairs to excite the qubits.

B. Performance Analysis

Next, we show that QMSN is nonblocking and design a routing algorithm for it.

Consider a DQC system with N QPUs. In the system, at most $N/2$ pairs of QPUs may request shared Bell states at any given time. Therefore, we specifically examine the case where N QPUs concurrently request $N/2$ shared Bell states. The number of all possible request patterns is $1 \cdot 3 \cdot 5 \cdot \ldots \cdot (N-1)$, equivalent to the total number of ways to partition N distinct elements into $N/2$ pairs. For any request pattern, if a network can simultaneously connect each pair of QPUs to the same BSA, we refer to it as nonblocking. We use $\mathbf{R}$ to denote the request pattern, which is the set of QPU pairs requesting shared Bell state. As QPU pairs are rerouted to the inputs of each stage of switches, for simplicity in notation, we will refer to QPU pairs as 'input pairs' at each stage. For instance, $\mathbf{R} = \{(0,3),(1,7),(2,6),(4,5)\}$ represents that the input pairs $(0,3),(1,7),(2,6)$, and $(4,5)$ are concurrently requesting shared Bell states.

In what follows, we demonstrate that the proposed QMSN is nonblocking – specifically, it is rearrangeably nonblocking. To prove it, we first introduce the following lemmas. In particular, Lemma 1 guides the design of the routing algorithm for stage 0, while Lemma 2 guides the design for the subsequent stages.

Lemma 1: For any request pattern $\mathbf{R}$ consisting of N paired inputs, we can partition the inputs into two sets such that:

- for each $i \in [N/2]$, input $2i$ and input $2i+1$ are always in different sets,
- each pair of inputs is divided between the two sets.

Proof: We begin by constructing a constraint graph with N vertices, each representing one of the N inputs. We add edges to the graph in the following manner. First, for each $i \in [N/2]$, add a red-colored edge between nodes $2i$ and $2i+1$. Then, for each pair of inputs in the request pattern $\mathbf{R}$, add a blue-colored edge between them. Note that there are N edges and N vertices in the graph, and each node is connected to two edges of different colors. As an example, let's consider a scenario where $N = 16$ and $\mathbf{R} = \{(0,9),(1,2),(3,5),(4,B),(6,D)(7,C),(8,A),(E,F)\}$, where $A, B, \cdots, F$ represent hexadecimal numbers. The resulting constraint graph would be as follows:

The lemma can then be restated as demonstrating that the constraint graph is 2-colorable for every possible request pattern. In a graph where each node has a degree of 2, and the number of nodes equals the number of edges, the only possible configuration consists of cycles. Nodes with terminal or branching structures cannot exist, as they would necessitate more or fewer than two edges per node, which contradicts the stated conditions. Next, we show that each cycle in the graph must contain an even number of nodes by proceeding with a proof by contradiction. Assume that a cycle has an odd number of nodes. If we start at a node and follow the cycle, the first edge is red, the second edge is blue, and so on, alternating between colors. However, when we reach the end of the cycle (i.e., come back to the starting node), since the number of nodes (and thus edges) in the cycle is odd, the color of the last edge must be the same as the color of the first edge. This contradicts our condition that each node must be connected to edges of two different colors. Therefore, each cycle has an even number of nodes. Since each cycle has an even number of nodes, we are able to traverse through the nodes in each cycle, applying alternating colors as we progress, which is also the algorithm to partition the set.

Taking Fig. 11 as an example, we traverse the circles by order of (0,1,2,3,5,4,B,A,8,9), (6,7,C,D), and (E,F). We put (0,2,5,B,8,6,C,E) to the first set, and other nodes to the second set. □

We formally state the algorithm we used in the proof of Lemma 1 in Algorithm 1. In the algorithm, the set partition decision for each input $i \in \mathbf{R}$ is denoted by $\mathbf{d}_0[i] \in \{0,1\}$. For each input i, if $\mathbf{d}_0[i] = 0$, we switch it to the upper output, i.e., output $2\lfloor i/2 \rfloor$. Otherwise, $\mathbf{d}_0[i] = 1$, and we switch input i

Algorithm 1 switchAlg^0 (for Stage 0)

Input: $\mathbf{R}$
Output: $\mathbf{d}_0$, routing decision for stage 0
Initialization: $\mathbf{d}[j] \leftarrow$ NaN for each input j in $\mathbf{R}$

```
// constructing a constraint graph
foreach input i in R do
    if i mod 2 == 0 then
        add an edge between node i and node i + 1;
    end
end
foreach request pair (i, j) ∈ R do
    add an edge between node i and node j;
end
// making routing decisions
d_0[0] ← 0 ;     /* decision for input 0 */
i* ← 0 ;         /* latest processed input */
while ∃i such that d_i = NaN do
    if i* has a neighbor j and d[j] == NaN then
        d_0[j] ← |1 − d_0[i*]|;
        i* ← j;
    else
        i* ← randomly chosen j where d_0[j] == NaN;
        d_0[i*] ← 0;
    end
end
return d_0;
```

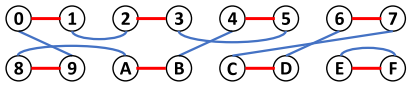


Fig. 11. Type I constraint graph for Stage 0.

Algorithm 2 switchAlg$^+$ (for Stage $n, n > 0$)

Input: $\mathbf{R}_n$, represented by inputs of stage n
Output: $\mathbf{d}_n$, routing decisions for stage $n > 0$
Initialization: $\mathbf{d}_n[j] \leftarrow$ NaN for each input j in $\mathbf{R}_n$

// constructing a constraint graph
foreach *input* i *in* $\mathbf{R}_n$ **do**
  if $i \mod 2 == 0$ **then**
    add an edge between node i and node $i+1$;
  end
end
foreach *request pair* (i, j) *in* $\mathbf{R}_n$ **do**
  add an edge between node i and node j;
end
// traversing nodes in cycles
foreach *cycle in the graph* **do**
  $start \leftarrow$ a randomly selected node in the cycle;
  $i^\star \leftarrow start$;
  $j^\star \leftarrow j$ such that $(i^\star, j) \in \mathbf{R}_n$;
  $\mathbf{d}_n[i^\star], \mathbf{d}_n[j^\star] \leftarrow 0$;
  $d^\star \leftarrow 0$; /* decision for the latest traversed two paired inputs */
  repeat
    if $j^\star \mod 2 == 0$ **then**
      $i^\star = j^\star + 1$;
    else
      $i^\star = j^\star - 1$;
    end
    $j^\star \leftarrow j$ such that $(i^\star, j) \in \mathbf{R}_n$;
    $\mathbf{d}_n[i^\star], \mathbf{d}_n[j^\star] \leftarrow |1 - d^\star|$;
    $d^\star \leftarrow |1 - d^\star|$;
  until $i^\star == start$;
end
return $\mathbf{d}_n$;

to the lower output, i.e., output $2\lfloor i/2 \rfloor + 1$. Given the routing decision at stage 0, the original request pattern, denoted by $\mathbf{R}$, can be represented using the input indices of stage 1, which are symbolized as $\mathbf{R}_1$. Similarly, when the routing decisions for the first $n-1$ stages are established, the original request pattern $\mathbf{R}$ can be represented by the input indices of stage n, which are denoted as $\mathbf{R}_n$. In addition, we use $\mathbf{d}_n$ to denote the routing decision for stage n. Taking Fig. 11 for an example, if we switch inputs $(0, 2, 5, B, 8, 6, C, E)$ to the upper outputs, the original request pattern $\mathbf{R} = \{(0, 9), (1, 2), (3, 5), (4, B), (6, D)(7, C), (8, A), (E, F)\}$ can be represented by the input indexes of stage 1 as $\mathbf{R}_1 = \{(0, C)(1, 8)(2, 9)(3, E)(4, D)(5, A)(6, B)(7, F)\}$, as shown in Fig. 12.

Next, we focus on stage $n > 0$, where the request pattern $\mathbf{R}_n$ is represented by the input indexes of stage n. We have the following Lemma.

Lemma 2: For any request pattern $\mathbf{R}_n$ consisting of N paired inputs, where one input from each pair belongs to $[N/2]$ and the other to $[N] \setminus [N/2]$, we can partition the inputs into two sets such that:

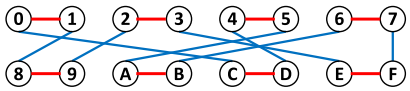


Fig. 12. Type II constraint graph for Stage $n, n > 0$.

Algorithm 3 Routing Algorithm

Input: number of inputs: N, request pattern: $\mathbf{R}$
Output: routing decision $\mathbf{D} = \{\mathbf{d}_0, \ldots, \mathbf{d}_{\log(N)-1}\}$
Initialization: $\mathbf{R}_n \leftarrow \emptyset$ for $n \in \{0, 1, \ldots, \log(N) - 1\}$

// routing decision for stage 0
$\mathbf{d}_0$ = switchAlg$^0(\mathbf{R})$;
foreach (i, j) *in* $\mathbf{R}$ **do**
  $i^\star \leftarrow 2\lfloor i/2 \rfloor + \mathbf{d}_0[i]$;
  $j^\star \leftarrow 2\lfloor j/2 \rfloor + \mathbf{d}_0[j]$;
  $\mathbf{R}_1 \leftarrow \mathbf{R}_1 \cup (S_r(j^\star, \log(N), N), S_r(i^\star, \log(N), N))$;
end
// routing decision for stage $n > 0$
for $n \in \{1, 2 \cdots \log(N) - 1\}$ **do**
  // 2^{n-1} subnetworks at stage n
  if $n == 1$ **then**
    $\mathbf{d}_1 \leftarrow$ switchAlg$^+(\mathbf{R}_1)$;
  else
    for $k \in \{0, 1 \cdots, 2^{n-1} - 1\}$ **do**
      for $(i, j) \in \mathbf{R}_n$ **do**
        if $\lfloor \frac{i}{2} \rfloor \mod 2^{n-1} == k$ **then**
          Add pair (i, j) to $\mathbf{R}_n^k$;
        end
      end
      $\mathbf{d}_n^k \leftarrow$ switchAlg$^+(\mathbf{R}_n^k)$;
      $\mathbf{d}_n \leftarrow \mathbf{d}_n \cup \mathbf{d}_n^k$;
    end
  end
  // switching to stage $n+1$
  foreach (i, j) *in* $\mathbf{R}_n$ **do**
    $i^\star \leftarrow 2\lfloor i/2 \rfloor + \mathbf{d}_n[i]$;
    $j^\star \leftarrow 2\lfloor j/2 \rfloor + \mathbf{d}_n[j]$;
    $\mathbf{R}_{n+1} \leftarrow \mathbf{R}_{n+1} \cup (S_l(i^\star, n+1, N), S_l(j^\star, n+1, N))$;
  end
end
Return $\mathbf{D}$;

- for each $i \in [N/2]$, input $2i$ and input $2i+1$ are in different sets,
- each pair of inputs from $\mathbf{R}_n$ is contained within the same set.

Proof: The proof is similar to that of Lemma 1. The basic idea is to construct a constraint graph that has N vertex. There is an edge of red color between nodes $2i$ and $2i+1$ for each $i \in [N/2]$. In addition, for each pair of inputs in the request pattern, we add an edge of blue color between them. For example, let $N = 16$ and the request pattern $\mathbf{R}_1$ is $\{(0, C)(1, 8)(2, 9)(3, E)(4, D)(5, A)(6, B)(7, F)\}$, the constraint graph is as follows:

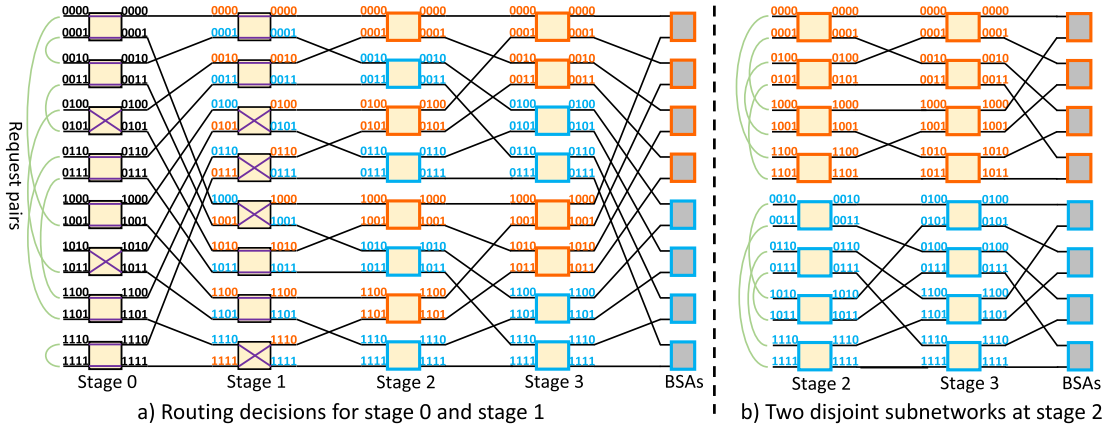


Fig. 13. An example of the designed routing algorithm for a 16-input QMSN.

Then, we can prove the graph contains only cycles, each of which contains a number of nodes that is a multiple of four. Assume a cycle has x blue edges, where one end of each edge is in $[N/2]$ and the other end is in $[N] \setminus [N/2]$. We have the cycle contains x nodes in $[N/2]$ and x nodes in $[N] \setminus [N/2]$, resulting title $2x$ nodes in the cycle. Since inputs $2i$ and $2i+1$ are in the same cycle, we have that the cycle has $2y$ inputs in $[N/2]$, where y is a positive integer. That is, the number of nodes in each cycle is $2x = 2 \cdot 2 \cdot y = 4y$.

Since the graph contains only cycles and each of which contains a number of nodes that is a multiple of four, we are able to traverse through the nodes in each cycle, starting with a random node, with the initial step moving along the direction of the blue edge. As we progress, we assign the first two consecutive nodes connected by a blue edge to Set A and the subsequent two nodes to Set B. We continue this pattern, alternating between Set A and Set B. In this way, input $2i$ and input $2i+1$ are in different sets, and each pair of inputs from $\mathbf{R}_n$ is contained within the same set, which proves the Lemma.

Taking Fig. 12 as an example, we start at 0 and traverse the constraint graph by the order of (0,C,D,4,5,A,B,6,7,F,E,3,2,9,8,1) and put (0-C,5-A,7-F,2-9) to the set A and others to the set B. □

Next, we show the QMSN network is non-blocking and design a switching network based on Lemma 1 and Lemma 2.

Theorem 2: For each request pattern $\mathbf{R}$ with N paired inputs, the designed N-input QMSN is nonblocking.

Proof: Based on Lemma 1, we can partition $[N]$ into two sets, Set 1 and Set 2, such that for each pair $(i,j) \in R$, i and j are situated in different sets. Furthermore, input $2i$ and input $2i+1$ reside in different sets for each $i \in [N/2]$. For each $i \in [N]$, we define $i' \triangleq 2\lfloor i/2 \rfloor$. For stage 0, each input i in the first set is switched to output i', and each input i in the second set is switched to output $i'+1$. No contention occurs at this stage since any potential contention could only occur between inputs $2i$ and $2i+1$, and these inputs are allocated to different sets (switch to different outputs). In addition, because inputs in each pair are in different sets, for each pair in $\mathbf{R}$, one QPU of the pair is switched to an even output connecting to one of the inputs in $[N/2]$ of stage 1 and the other one is switched to an odd output connecting to one of the inputs in $[N] \setminus [N/2]$ of stage 1. For instance, consider the request pattern depicted in Figure11. The two sets would be $\{0,2,5,6,8,B,C,E\}$ directed to even outputs and $\{1,3,4,7,9,A,D,F\}$ directed to odd outputs, as the switch configurations of stage 0 in Fig. 13.a.

Next, we can ignore stage 0 and focus on the problem of routing the paired inputs of stage 1 to the BSAs, where one input of each pair is in $[N/2]$ and the other one is in $[N] \setminus [N/2]$. For example, the original request pairs in Fig. 13.a connect to input pairs $\mathbf{R}_1 = \{(0,C),(1,8),(2,9),(3,E),(4,D),(5,A),(6,B),(7,F)\}$ in stage 1, and $\mathbf{R}_1$ is the request pattern represented by input index in stage 1. Based on Lemma 2, we can partition the inputs of stage 1 into two sets such that $2i$ and $2i+1$ are in separate sets and paired inputs $(i,j) \in R_1$ are in the same sets. Taking Fig. 13.a as an example, using the constraint graph shown in Fig. 12, we partition the inputs of stage 1 into Set A=$\{(0,C),(2,9),(5,A),(7,F)\}$ and Set B=$\{(1,8),(3,E),(4,D),(6,B)\}$ labeled by red color and blue color, respectively. Then, the inputs in set A and set B are switched to outputs $\{2i, i \in [2/N]\}$ and outputs $\{2i+1 | i \in [N/2]\}$ in stage 1, respectively. The last digit of these output indices in binary form for set A and set B is 0 and 1, respectively. That is, inputs in Set A are connected exclusively to switches whose i-th least significant bit (LSB) is 0 at stage i (depicted as red switches), e.g., red switches in stage 2 have their 2nd LSB set to 0. The topology of the red switches after stage 1 corresponding with set A is shown in Fig. 13.b, which is the same as the last $\log(N/2)-1$ stages of $N/2$-input QMSN, e.g., Fig. 10 without Stage 0. Similarly, inputs in set B are only connected to switches with the i-th LSB is 1 at stage i, and the topology of the switches and BSAs corresponding with set B is shown in Fig. 13.b, which is the same as the last $\log(N/1)-1$ stages of $N/2$-input QMSN. That is, the problem is divided into two smaller problems with $N/2$ inputs, where one input of each request pair is in the first $N/4$ inputs, and the other one is in the last $N/4$ inputs. For each of the two $N/2$-input networks with $(\log(N)-2)$ stages, we can construct a Type II constraint graph to get the configuration of its switches in its first stage

and get two $N/4$-input networks each of $(\log(N)-3)$ stages. Following this method, we can get $N/4$ 4-input networks at the end. In each of the 4-input networks, which comprise two switches, one input from every request pair connects the first switch, while the other input connects to the last switch, which is non-blocking. Therefore, each pair of QPUs can be routed to a unique BSA, which proves the theorem. The approach for deciding the switching configuration of the stages after the first stage is formally stated in Algorithm 2. □

The routing algorithm is similar to the process of the proof of Theorem 2 and is formally stated in Algorithm 3. It takes $\mathcal{O}(N)$ time steps for each stage, and the time complexity of the routing algorithm is $\mathcal{O}(N\log(N))$.

Next, we analyze the success probability of each entanglement generation attempt in QMSN. Parameters p_d, p_f, and p_p are defined the same as that for (1). We use p_i to denote the probability that a photon successfully traverses a 2×2 MZI switch, which is determined by the switch's insertion loss. For each pair of QPUs, their emitted photons must pass through a total of $2\log_2(N)$ switches. The probability that the photons successfully pass through all the switches is given by $p_i^{2\log_2(N)}$. Then, the success probability of each entanglement generation attempt, denoted by p'_a, is

$$p'_a = 0.5p_p^2 p_f^2 p_i^{2\log_2(N)} p_d. \tag{9}$$

If we keep attempting for n times, the success probability of generating at least one entangled pair, denoted by $p'_r(n)$, is

$$p'_r(n) = 1-(1-p'_a)^n. \tag{10}$$

Since the entanglement generation in the switching network does not involve entanglement swapping on the intermediate QPUs. Let's assume that in the generated shared qubit pairs, the probabilities of bit flip and phase flip errors are independent, denoted by p_b and p_ϕ, respectively. We have that the fidelity of the generated shared qubit pairs in the proposed switching network, denoted by F_{QMSN}, is

$$F_{QMSN} = 1-(1-p_b)(1-p_\phi). \tag{11}$$

The simplicity of QMSN, which doesn't require entanglement swapping on the intermediate QPUs, makes the protocol for generating entanglement simpler compared to other static quantum networks. A straightforward protocol for generating entanglement in QMSN could be repeated attempts until successful entanglement is achieved.

VI. Performance Evaluation

In this section, we evaluate the performance of the proposed switching network using parameters derived from real-world experimental results. Simulation code is publicly available on GitHub at https://github.com/yuliu3/QMSN.

A. Simulation Settings

We compare the switching network with the Beneš network and line, ring, and grid topologies from the literature. In switching networks, each QPU has a single communication qubit, whereas in line, ring, and grid networks, QPUs can have up to two, two, and four communication qubits with corresponding photon collection modules. The implementation of static networks is inherently more complex, necessitating additional optical and electronic controls, along with photon collection and coupling modules. Unlike quantum networks, where throughput is a crucial performance index [23], [24], distributed computing requires generating shared Bell states according to the order of the corresponding remote gates in the quantum circuit. Consequently, we adopt the average time for Successful Entangled qubit pair Generation (SEG) as a performance metric to evaluate the networks. There may be some remote gates that can be performed simultaneously, and the proposed switching network can generate the desired shared Bell states simultaneously without introducing new problems. Different request pairs in the static networks are competing for the resources such as communication qubits and links, e.g., it is impossible to generate shared Bell states between QPU 1 and QPU 3 and between QPU 2 and QPU 4 simultaneously in Figure 7.a. We investigate the scenario where request pairs arrive sequentially. This particular condition favors the static topologies over the switching networks because the switching network can accommodate $N/2$ simultaneous requests without compromising the average time required for SEG. In contrast, attempts to simultaneously generate multiple entangled pairs in static topologies can lead to resource contention, consequently increasing the average time required for SEG.

The Beneš network operates by treating the BSAs as outputs and routing inputs that require a shared Bell state to the same BSA. The Beneš network is adapted to have $2(\log_2(N)-1)$ stages, as the last stage is substituted with BSAs. We assume that static networks use the protocol in [24] to generate shared Bell states. Under the protocol, there are one, two, and up to four routes between each pair of S-D QPUs in the line network, ring network, and grid network, respectively. All routes are simultaneously engaged in entanglement generation. Based on Figure 3 in [44], we assume the external phase of the static networks lasts for 10^{-3}s. From [20], the maximum photon production rate γ_{max} is 2 MHz. The propagation latency of photons is omitted since the QPUs are placed in close proximity. The internal phase duration for static networks, typically in tens of microseconds [45], includes the time taken for entanglement swapping plus communication time, where the entanglement process swapping typically takes tens of microseconds [45]. Assume that the duration of the internal phase is negligible, as it favors the static networks due to the absence of entanglement swapping on intermediate processors (repeaters) in the switching networks.

B. Simulation Results

We first simulate a system comprising of 16 QPUs. Specifically, we set the photon production rate γ as 1 Mhz and the external phase duration time T_e to be 1 ms. Let $p_i = 0.9$ and $p_p = 0.04$ [39], [40]. Additionally, from [20], we set $p_f = 0.98$, $p_d = 0.8$, and $p_s = 0.9$. To evaluate the performance of the proposed QMSN, we conduct simulations using the Quantum Fourier Transform (QFT) circuit [46]. In this setup, 64 qubits are distributed across 8 QPUs, with each QPU hosting 8 qubits. To minimize the number of remote gate operations, we map the i-th qubit and the

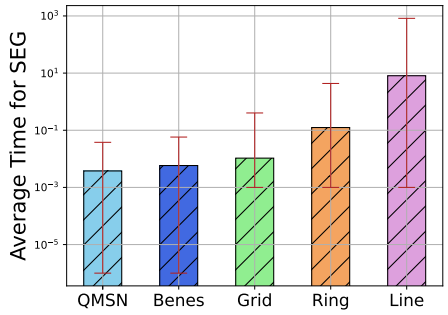


Fig. 14. Average SEG time within the QFT Circuit [46].

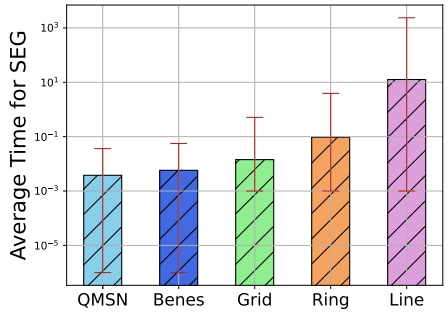


Fig. 15. Average SEG time within a real-world circuit [47].

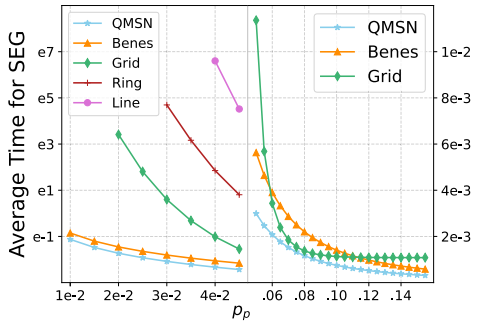


Fig. 16. SEG Time vs. p_p.

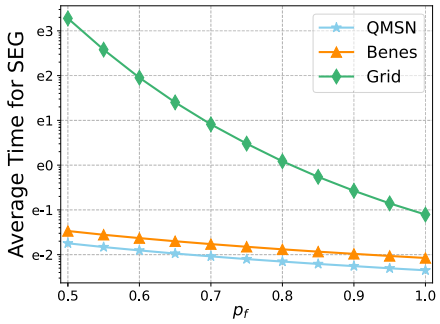


Fig. 17. SEG Time vs. p_f.

$(63-i)$-th qubit to the same QPU, reducing the need for remote swap gates at the circuit's end. Given the symmetric nature of the QFT circuit, this mapping strategy does not affect performance, allowing us to assign the qubit pairs—i-th and $(63-i)$-th qubits—to QPUs arbitrarily. Figure 14 presents a statistical summary of the time for SEG across various network topologies. The length of the bar signifies the average time taken for SEG, while the red error bars indicate the range from minimum to maximum times observed. The average

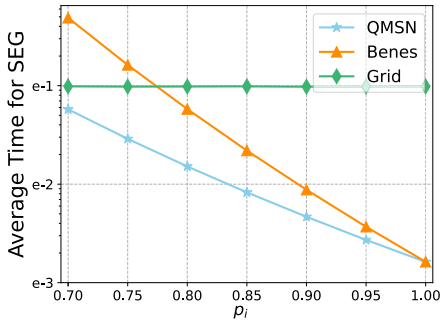


Fig. 18. SEG Time vs. p_i.

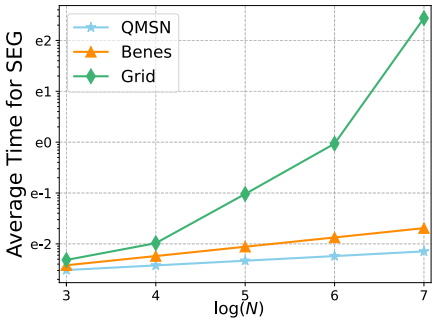


Fig. 19. SEG Time vs. $log_2(N)$.

time for SEG of the Beneš, grid, ring, and line networks are $1.52\times$, $2.75\times$, $3.31e{+}01\times$, and $3.26e{+}03\times$ of that in the proposed QMSN, respectively. Next, we use the remote gate order of a real-world quantum circuit from [47]. We first allocate the 16 logical qubits across 16 QPUs, where multiple physical qubits within each QPU represent a single logical qubit for quantum error correction. The remote CNOT gates are then generated according to their order in the quantum circuit. Figure 15 shows that, in this scenario, the average time for SEG in the Beneš, grid, ring, and line networks are $1.53\times$, $3.76\times$, $2.50e{+}01\times$, and $3.33e{+}03\times$ of that in QMSN, respectively.

Subsequently, we investigate the performance of the topologies with 32 QPUs, each hosting 4 qubits, under varying p_p. Other parameters are the same as those used for Figure 14. Figure 16 demonstrates that the proposed switching network consistently outperforms the baselines across all considered p_p values. Note that p_p are typically less than 0.05 [40]. Due to the significantly higher average time for SEG in the ring and line networks compared to other networks, our subsequent discussions will focus on comparing QMSN with the Beneš network and the static grid network.

Next, we evaluate the time for SEG in a system with 16 QPUs across varying photon loss probabilities in the fiber, i.e., p_f. Other parameters remain the same as those used for Figure 14. A typical fiber loss value for 422 nm photons emitted by Ba^+ ions is 30 dB/km [16]. For fiber lengths within the range of 5 m to 10 m, p_f varies between 0.933 and 0.966. As depicted in Figure 17, the proposed switching network demonstrates a lower average time for SEG compared to the Beneš and grid networks. Then, we investigate the performance under different switch insertion losses, while keeping all other parameters consistent with those utilized for

Figure 14. As shown in Figure 18, the time required for SEG in the switching network is consistently less than that in the static networks for all examined p_i settings. Since there is no switch in the static networks, the average time for SEG of the grid network remains constant. According to [48], the insertion loss for each MZI binary switch can be reduced to less than 1 dB. The insertion loss can be further reduced by refining the fabrication, optimizing the PIC design, and utilizing resonant structures [31]. When the insertion loss is 1 dB, p_i is around 0.8, resulting in the average time for SEG of the grid network being 4.6 times that of QMSN. Next, we evaluate the scalability of the proposed QMSN using different N, where N represents the number of QPUs. All other parameters are the same with those utilized for Figure 14. As shown in Figure 19, as N increases, the average time of the static grid network increases exponentially faster than that of the switching networks.

The time required for SEG in the switching network is consistently less than that in the static network topologies for all the examined settings. Under certain settings, the grid network may exhibit competitive performance, e.g., in Figure 16, when $p_p = 0.075$, the time for SEG in the switching network is only 14% less than that of the grid network. This can be attributed to two reasons. Firstly, the parameter settings are atypical. For instance, parameter p_p is typically smaller 0.05. Secondly, the comparison is not fair as QPUs in the grid network have up to four communication qubits, while those in the switching network are limited to just one. Should the QPUs within the switching network be enhanced to support multiple communication qubits, a consequent reduction in the time for SEG could be anticipated.

VII. Conclusion

Distributed quantum computing is a promising paradigm to increase the scale of quantum computing systems. In this paper, we focus on designing networks to interconnect trapped ion-based QPUs for the implementation of efficient DQC. We show that static networks suffer from high system complexity and deliver poor performance due to the low photon collection and coupling efficiency experienced in real-world systems. Therefore, we design a novel multistage switching network dedicated to DQC. We prove that the proposed network is nonblocking and design an efficient routing algorithm for the proposed network. In addition, we conducted extensive real-world data-driven simulations. Results show that the proposed switching network significantly outperforms popular baselines.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.

[2] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin, "Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices," *Phys. Rev. X*, vol. 10, no. 2, Jun. 2020, Art. no. 021067.

[3] H.-S. Zhong et al., "Phase-programmable Gaussian boson sampling using stimulated squeezed light," *Phys. Rev. Lett.*, vol. 127, no. 18, Oct. 2021, Art. no. 180502.

[4] Y. Cao, J. Romero, and A. Aspuru-Guzik, "Potential of quantum computing for drug discovery," *IBM J. Res. Develop.*, vol. 62, no. 6, pp. 1–6, Nov. 2018.

[5] A. Morvan et al., "Phase transition in random circuit sampling," 2023, *arXiv:2304.11119*.

[6] IBM Newsroom, "IBM unveils 400 qubit-plus quantum processor and next-generation IBM quantum system two," Nov. 2022. Accessed: Feb. 23, 2025. [Online]. Available: https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two

[7] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Secur. Privacy*, vol. 16, no. 5, pp. 38–41, Sep. 2018.

[8] Z. Yang, M. Zolanvari, and R. Jain, "A survey of important issues in quantum computing and communications," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 2, pp. 1059–1094, 2nd Quart., 2023.

[9] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Appl. Phys. Rev.*, vol. 6, no. 2, Jun. 2019, Art. no. 021314.

[10] M. Caleffi et al., "Distributed quantum computing: A survey," 2022, *arXiv:2212.10609*.

[11] Y. Mao, Y. Liu, and Y. Yang, "Qubit allocation for distributed quantum computing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2023, pp. 1–10.

[12] P. C. Humphreys et al., "Deterministic delivery of remote entanglement on a quantum network," *Nature*, vol. 558, no. 7709, pp. 268–273, 2018.

[13] J. P. Covey, H. Weinfurter, and H. Bernien, "Quantum networks with neutral atom processing nodes," 2023, *arXiv:2304.02088*.

[14] N. Moll et al., "Quantum optimization using variational algorithms on near-term quantum devices," *Quantum Sci. Technol.*, vol. 3, no. 3, Jul. 2018, Art. no. 030503.

[15] T. P. Harty et al., "High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit," *Phys. Rev. Lett.*, vol. 113, no. 22, Nov. 2014, Art. no. 220501.

[16] L. J. Stephenson et al., "High-rate, high-fidelity entanglement of qubits across an elementary quantum network," *Phys. Rev. Lett.*, vol. 124, no. 11, Mar. 2020, Art. no. 110501.

[17] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini, and G. Bianchi, "Quantum internet: Networking challenges in distributed quantum computing," *IEEE Netw.*, vol. 34, no. 1, pp. 137–143, Jan. 2020.

[18] D. Cuomo et al., "Optimized compiler for distributed quantum computing," *ACM Trans. Quantum Comput.*, vol. 4, no. 2, pp. 1–29, Jun. 2023.

[19] S. Krastanov et al., "Optically heralded entanglement of superconducting systems in quantum networks," *Phys. Rev. Lett.*, vol. 127, no. 4, Jul. 2021, Art. no. 040503.

[20] J. Hannegan, J. D. Siverns, J. Cassell, and Q. Quraishi, "Improving entanglement generation rates in trapped-ion quantum networks using nondestructive photon measurement and storage," *Phys. Rev. A, Gen. Phys.*, vol. 103, no. 5, May 2021, Art. no. 052433.

[21] Y. Zhao, G. Zhao, and C. Qiao, "E2E fidelity aware routing and purification for throughput maximization in quantum networks," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Oct. 2022, pp. 480–489.

[22] A. Farahbakhsh and C. Feng, "Opportunistic routing in quantum networks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2022, pp. 490–499.

[23] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: Model and designs," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, 2020, pp. 62–75.

[24] M. Pant et al., "Routing entanglement in the quantum internet," *NPJ Quantum Inf.*, vol. 5, no. 1, p. 25, Mar. 2019.

[25] S. Pouryousef, N. K. Panigrahy, and D. Towsley, "A quantum overlay network for efficient entanglement distribution," 2022, *arXiv:2212.01694*.

[26] Y. Mao, Y. Liu, and Y. Yang, "Probability-aware qubit-to-processor mapping in distributed quantum computing," in *Proc. 1st Workshop Quantum Netw. Distrib. Quantum Comput.* New York, NY, USA: Association for Computing Machinery, Sep. 2023, pp. 51–56, doi: 10.1145/3610251.3610554.

[27] V. E. Beneš, "Heuristic remarks and mathematical problems regarding the theory of connecting systems," *Bell Syst. Tech. J.*, vol. 41, no. 4, pp. 1201–1247, Jul. 1962.

[28] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-24, no. 12, pp. 1145–1155, Dec. 1975.

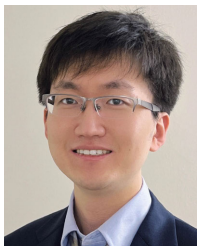
[29] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multiprocessor systems," in *Proc. 1st Annu. Symp. Comput. Archit.*, 1973, pp. 21–28.

[30] L.-M. Duan and C. Monroe, "Colloquium: Quantum networks with trapped ions," *Rev. Modern Phys.*, vol. 82, no. 2, pp. 1209–1224, Apr. 2010.
[31] M. Dong et al., "Programmable photonic integrated meshes for modular generation of optical entanglement links," *Npj Quantum Inf.*, vol. 9, no. 1, p. 42, Apr. 2023.
[32] S. Bartolucci et al., "Switch networks for photonic fusion-based quantum computing," 2021, *arXiv:2109.13760*.
[33] S. Niu and A. Todri-Sanial, "Multi-programming cross platform benchmarking for quantum computing hardware," 2022, *arXiv:2206.03144*.
[34] A. H. Burrell, D. J. Szwer, S. C. Webster, and D. M. Lucas, "Scalable simultaneous multiqubit readout with 99.99% single-shot fidelity," *Phys. Rev. A, Gen. Phys.*, vol. 81, no. 4, Apr. 2010, Art. no. 040302.
[35] J. P. Gaebler et al., "High-fidelity universal gate set for $^{9}Be^{+}$ ion qubits," *Phys. Rev. Lett.*, vol. 117, no. 6, 2016, Art. no. 060505.
[36] A. Barenco et al., "Elementary gates for quantum computation," *Phys. Rev. A, Gen. Phys.*, vol. 52, no. 5, p. 3457, 1995.
[37] S. Blinov, B. Wu, and C. Monroe, "Comparison of cloud-based ion trap and superconducting quantum computer architectures," *AVS Quantum Sci.*, vol. 3, no. 3, Sep. 2021, Art. no. 033801.
[38] S. Sheldon, "Quantum computing with noisy qubits," in *Proc. Frontiers Eng., Rep. Leading-Edge Eng. Symp.* Lexington, MA, USA, National Academies Press, 2019, pp. 13–17.
[39] G. Shu, N. Kurz, M. R. Dietrich, and B. B. Blinov, "Efficient fluorescence collection from trapped ions with an integrated spherical mirror," *Phys. Rev. A, Gen. Phys.*, vol. 81, no. 4, Apr. 2010, Art. no. 042321.
[40] C. Jones, D. Kim, M. T. Rakher, P. G. Kwiat, and T. D. Ladd, "Design and analysis of communication protocols for quantum repeater networks," *New J. Phys.*, vol. 18, no. 8, Aug. 2016, Art. no. 083015.
[41] A. VanDevender, Y. Colombe, J. Amini, D. Leibfried, and D. Wineland, "Efficient fiber optic detection of trapped ion fluorescence," *Phys. Rev. Lett.*, vol. 105, no. 2, Jul. 2010, Art. no. 023001.
[42] W. Dai, A. Rinaldi, and D. Towsley, "Entanglement swapping in quantum switches: Protocol design and stability analysis," 2021, *arXiv:2110.04116*.
[43] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge Univ. Press, 2010.
[44] P. Drmota et al., "Robust quantum memory in a trapped-ion quantum network node," *Phys. Rev. Lett.*, vol. 130, no. 9, Mar. 2023, Art. no. 090803.
[45] P. Gerbert and F. Rueß, "The next decade in quantum computing and how to play," Boston Consulting Group, Oct. 2018. Accessed: Feb. 23, 2025. [Online]. Available: https://bcg.com/publications/2018/next-decade-quantum-computing-how-play
[46] Various Authors. (2023). *Qiskit Textbook, Quantum Fourier Transform*. Github. [Online]. Available: https://github.com/Qiskit/textbook
[47] L. Burgholzer. (Jun. 2023). *MQT QMAP—A Tool for Quantum Circuit Compilation*. Accessed: Jul. 30, 2023. [Online]. Available: https://github.com/cda-tum/mqt-qmap
[48] K. F. Lee and G. S. Kanter, "Low-loss high-speed C-band fiber-optic switch suitable for quantum signals," *IEEE Photon. Technol. Lett.*, vol. 31, no. 9, pp. 705–708, May 1, 2019.

Yu Liu received the B.Eng. degree in telecommunication engineering from Xidian University, Xi'an, China, and the Ph.D. degree in computer engineering from Stony Brook University, Stony Brook, NY, USA. He is currently an Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests encompass edge computing and networks, low-earth orbit satellite networks, online algorithm design, network function virtualization, and distributed quantum computing.

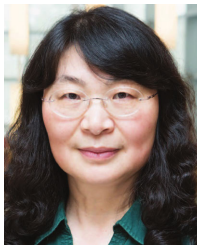
Yingling Mao received the B.S. degree in mathematics and applied mathematics from Zhiyuan College, Shanghai Jiao Tong University, Shanghai, China, in 2018. She is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Stony Brook University. Her research interests include network function virtualization, software-defined networks, cloud computing, and distributed quantum computing.

Xu Xu received the B.S. degree in physics from Nanjing University, Nanjing, China, in 2023. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Stony Brook University. His research interests include quantum networks, distributed quantum computing, and quantum communication.



Xiaojun Shang (Member, IEEE) received the B.Eng. degree in information science and electronic engineering from Zhejiang University, the M.S. degree in electronic engineering from Columbia University, and the Ph.D. degree in computer engineering from Stony Brook University. He is currently an Assistant Professor with the Department of Computer Science and Engineering, The University of Texas, Arlington. His research interests include the areas of mobile edge computing, software defined networking, and quantum computing.

Fan Ye (Fellow, IEEE) received the B.E. and M.S. degrees from Tsinghua University, Beijing, China, and the Ph.D. degree from the Computer Science Department, University of California at Los Angeles, Los Angeles, CA, USA. He is a Professor with the ECE Department, Stony Brook University, Stony Brook, NY, USA. He has published more than 60 peer-reviewed papers, that have received more than 8000 citations according to Google Scholar. He has 21 granted/pending U.S. and international patents/applications. His current research interests include mobile sensing platforms, systems and applications, the Internet of Things, indoor location sensing, wireless, and sensor networks. He received the IBM Research Division Award, five Invention Achievement Plateau Awards, and the Best Paper Award for International Conference on Parallel Computing in 2008. He was the Co-Chair of the Mobile Computing Professional Interests Community, IBM Watson, for two years.



Yuanyuan Yang (Life Fellow, IEEE) received the B.Eng. and M.S. degrees in computer science and engineering from Tsinghua University, Beijing, China, and the M.S.E. and Ph.D. degrees in computer science from Johns Hopkins University, Baltimore, MD, USA. She is a SUNY Distinguished Professor of computer engineering and computer science with Stony Brook University, NY, USA. She is on leave from the National Science Foundation, as the Program Director. Her research interests include edge computing, data center networks, cloud computing, and wireless networks. She has published more than 460 papers in major journals and refereed conference proceedings and holds seven U.S. patents in these areas. She has served as the general chair, the program chair, and the vice chair for several major conferences and a program committee member for several conferences. She is the Editor-in-Chief of IEEE TRANSACTIONS ON CLOUD COMPUTING and an Associate Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and *ACM Computing Surveys*. She has served as the Associate Editor-in-Chief for IEEE TRANSACTIONS ON CLOUD COMPUTING, the Associate Editor-in-Chief and an Associated Editor for IEEE TRANSACTIONS ON COMPUTERS, and an Associate Editor for IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS.