This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2024.Doi Number*

# Resource Management for Distributed Quantum Computing in Quantum Data Centers

**Zebo Yang[1], Member, IEEE, Chenliang Tian[2], Student Member, IEEE, and Raj Jain[2], Life Fellow, IEEE, Ramana Kompella[3], Member, IEEE, Reza Nejabati[3], Member, IEEE, Mounir Hamdi[4], Fellow, IEEE, Aiman Erbad[5], Senior Member, IEEE, Hassan Shapourian[6], Member, IEEE, and Lei Yang[7], Member, IEEE**

[1]Department of Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL, USA
[2]Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO, USA
[3]Cisco Research, San Jose, CA, USA
[4]College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar
[5]College of Engineering, Qatar University, Doha, Qatar
[6]Zyphra, CA, USA
[7]Department of Information Sciences and Technology at George Mason University, Fairfax, VA, USA

Corresponding author: Zebo Yang (e-mail: yangz@fau.edu).

This work was supported in part by the Qatar Research, Development, and Innovation (QRDI)Academic Research Grant #ARG02-0415-240191, Cisco University Research Grant #98690499, and NSF/CSSI #2513431. The statements made here are solely the responsibility of the authors.

**ABSTRACT** In recent years, advances in quantum computing have been driven by substantial improvements in both the number and quality of qubits. As the field progresses, there is growing interest in interconnecting quantum systems to enable scalable computation through Distributed Quantum Computing (DQC) architectures. Consider a distributed quantum application executed across multiple Quantum Processing Units (QPUs) within a quantum data center, where remote gates require establishing entanglement between different QPUs. The creation of such end-to-end entanglement can lead to network congestion and resource contention. To address these challenges, we propose a resource management framework that maximizes fidelity-guaranteed throughput while satisfying dependency constraints. We first formulate the problem as a Mixed-Integer Linear Programming (MILP) model to provide a performance benchmark. Building on this, we develop efficient approximate scheduling algorithms that achieve performance comparable to the optimization solver. Although a trade-off exists between execution time and network throughput, simulation results demonstrate that one of the proposed strategies, Weighted Group Least Resource First (WGLRF), closely approximates the solver's performance across most scenarios. These findings suggest that the lightweight strategy is sufficient for current DQC settings, offering a practical solution for managing remote-gate resource contention in distributed quantum circuits and improving overall system performance.

**INDEX TERMS** Distributed Quantum Computing, Quantum Data Centers, Quantum Networks, Resource Management, Scheduling, Topology.

## I. INTRODUCTION

Recent advances in quantum computing, marked by the introduction of increasingly powerful quantum machines and advancing communication experiments, signify substantial progress in the field [1], [2], [3], [4]. As quantum computing continues to evolve, there is a growing demand to interconnect these systems, especially within a data center connected by optical switches (analogous to traditional circuit-switched networks). This can be achieved through Distributed Quantum Computing (DQC) [5], [6], [7], [8], in which individual quantum processors, each with limited capacity, work together to address computational tasks that exceed the capabilities of a single device. Unlike classical distributed computing, where adding more processors leads to a linear increase in computing power, the integration of additional quantum processors can result in an exponential enhancement of quantum computational capacity [9].

Quantum computers can be interconnected using a hybrid approach combining quantum and classical techniques [10], [11], [12], [13]. Hybrid quantum-classical networks often rely on a central controller to handle network tasks like scheduling and routing, a practice that extends to DQC. However, scaling quantum resources within data centers presents unique



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



challenges due to the inherent complexities of quantum technology, including decoherence, probabilistic operations, and unreliable entanglement generation [14]. Addressing these challenges calls for innovative strategies specifically designed for DQC, aligned with the shift from monolithic to distributed architectures. Enhancing performance necessitates unique resource management and optimization techniques suited to the quantum domain.

In the current Noisy Intermediate-Scale Quantum (NISQ) era [15], executing a complex quantum algorithm in DQC involves partitioning the algorithm into smaller components (i.e., dividing the quantum circuit into sub-circuits) and allocating them to specific Quantum Processing Units (QPUs). Each sub-circuit comprises a set of quantum gates, some of which can be executed locally within a QPU, while others require resources and collaboration across QPUs, i.e., inter-processor operations, specifically referred to as remote gates. A primary goal in quantum algorithm partitioning is to reduce the number of remote gates, as their execution is significantly more resource-intensive compared to local gates. However, achieving this is highly challenging and inherently constrained [16], [17]. Therefore, minimizing the overhead associated with remote gate execution, such as employing resource management to enable their parallel execution, is equally important. Resource management strategies from classical networks cannot be directly applied due to the above differences [18].

A key method for executing remote gates between two QPUs is by leveraging a pre-established entangled pair shared between them [9]. Thus, an overlay network for entanglement distribution is crucial for enabling remote operations in DQC [19], [20]. This overlay network facilitates the distribution of end-to-end (E2E) entanglement between arbitrary nodes in the network. The end nodes then utilize this E2E entanglement as a communication resource to perform remote operations. For entanglement distribution to function, each node in the network must communicate to coordinate the implementation of local quantum operations, also known as Local Operations and Classical Communication (LOCC) [21]. Once paths between end nodes are established, intermediate nodes (i.e., quantum repeaters) execute entanglement swapping and purification [22], [23] to produce E2E entanglement of sufficient quality. The key challenge lies in selecting paths for concurrent remote gates, which require specific QPUs for execution, to achieve optimal performance. This involves maximizing the availability of resources, e.g., E2E entangled pairs with desired quality, for a batch of remote gates within a given time frame.

Consider a quantum data center with multiple QPUs distributed across various racks tasked with executing one or more complex quantum circuits in a distributed manner. Within a distributed quantum application, numerous requests may arise for resources to execute remote gates across distributed QPUs, each requiring E2E entanglement between QPUs. Ideally, we want to maximize the concurrent execution of remote gates. However, as the number of requests increases, congestion and resource contention become inevitable. For instance, multiple remote gates may rely on a shared link, but the resources available between the QPUs connected by that link may be insufficient to meet all demands. In this work, the evaluation primarily considers requests generated from a single partitioned circuit, although the proposed request-driven framework is applicable to both single-tenant and multi-tenant settings.

We propose a resource management framework for quantum data centers that prioritizes gate dependencies and maximizes quality-assured resources for remote gates. The problem is first formulated as a Mixed-Integer Linear Programming (MILP) optimization model, serving as a benchmark. We then introduce a set of efficient scheduling algorithms, such as prioritizing requests with minimal resource demands. While there is a tradeoff between execution time and request satisfaction, simulations show that the proposed strategies closely approximate the solver's results in most scenarios.

This demonstrates that simple algorithms are sufficient for current DQC settings. Unlike sophisticated methods such as reinforcement learning or optimization solvers used in existing works, a straightforward strategy can achieve comparable performance with significantly faster execution times. Even in harsh conditions, the results are acceptable. That is, an efficient strategy can offer a more practical solution for managing the potentially large number of remote gates in a quantum circuit. These efficiency improvements can greatly enhance overall performance, especially for large quantum computations involving thousands or even millions of sub-circuits. Saving time on processing one batch of remote gates, a set of sub-circuits, can translate into substantial time savings over the course of the entire computation.

The contributions of this paper can be summarized as:

1. We propose a resource management framework for quantum data centers that abstractly models the components of a partitioned quantum circuit executed in a DQC environment.
2. We formulate the resource management problem as an MILP model, which serves as a reference for performance evaluation.
3. We introduce a set of efficient algorithms that deliver performance comparable to MILP while achieving significantly faster execution.
4. We show that lightweight scheduling strategies can achieve strong performance with substantially lower complexity compared to sophisticated methods.

The remainder of the paper is organized as follows: Section II provides an overview of the preliminaries related to DQC and quantum data centers, along with a discussion of related work. Section III presents the typical topologies identified for quantum data centers. Section IV introduces the proposed resource management framework for quantum data centers using partitioned quantum circuits. In Section V, we formulate the MILP problem and present a set of scheduling algorithms. Section VI evaluates the proposed methods and presents simulation results. Finally, Section VII concludes the paper.

## II. BACKGROUND AND RELATED WORK

This section presents the foundational concepts related to entanglement distribution, DQC, and quantum data centers, along with an overview of related work.



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



### *A. Entanglement Swapping and Purification*

As discussed earlier, a key approach for executing remote gates between two QPUs involves leveraging pre-established entangled qubit pairs between intermediate nodes on the path between them. Once the paths are established, intermediate nodes perform entanglement swapping and purification to create E2E entanglement of sufficient quality (fidelity).

Entanglement swapping is a quantum operation that enables the creation of remote entanglement between two distant end nodes that are not directly connected by a quantum link. This operation is performed by an intermediate node, typically known as a quantum repeater. As shown in Fig. 1(a), end nodes A and B aim to establish E2E entanglement, but no direct quantum link exists between them. Instead, both nodes share direct links with an intermediate (repeater) node. Each node generates direct-link entanglement with the repeater. The repeater, possessing one side of each direct-link entangled pair, then performs entanglement swapping on its sides of the two entangled pairs. If the operation is successful, an E2E entangled pair is formed between nodes A and B, as depicted by the blue dashed wavy line in Fig. 1(a).

Entanglement purification is a process designed to enhance entanglement quality, quantified by fidelity, a measure ranging from 0 to 1, where 1 indicates a perfect entangled state. This operation leverages multiple entangled pairs to distill several lower-fidelity pairs into a single higher-fidelity entangled pair. As shown in Fig. 1(b), entanglement purification can proceed through iterative rounds to improve fidelity. For example, in the first round, multiple entangled pairs with an initial fidelity of 0.8 are combined, resulting in purified pairs with an improved fidelity of 0.94 (further details on how this value is derived can be found in Section V). A subsequent purification round further increases the fidelity to 0.98, producing entangled pairs of significantly higher quality. This iterative approach ensures the creation of entangled pairs suitable for tasks requiring high-quality E2E entanglement, such as remote gate execution.

For simplicity, this research considers the generation of maximally entangled pairs (Bell states [24]) between adjacent nodes and the implementation of Bell state measurements [25] at repeater nodes to facilitate entanglement swapping. Each attempt at generation or swapping may succeed or fail, resulting in probabilistic and unpredictable routing paths. We assume a memory-based entanglement distribution setting in which direct-link Bell pairs can be temporarily buffered at neighboring nodes and consumed for entanglement swapping and purification. The model abstracts the number of currently available buffered Bell pairs on a link as the link capacity during a scheduling interval.

### *B. Entanglement Distribution in DQC*

In DQC and, more broadly, in quantum networks, qubits within a QPU are classified as communication or data qubits [26]. Communication qubits are specialized qubits within a QPU reserved for facilitating inter-QPU operations, primarily by enabling the generation and distribution of entangled states necessary for networking purposes or remote operations. Data qubits are qubits within a QPU dedicated to storing and processing quantum information, enabling the execution of computational tasks and operations within a quantum circuit.

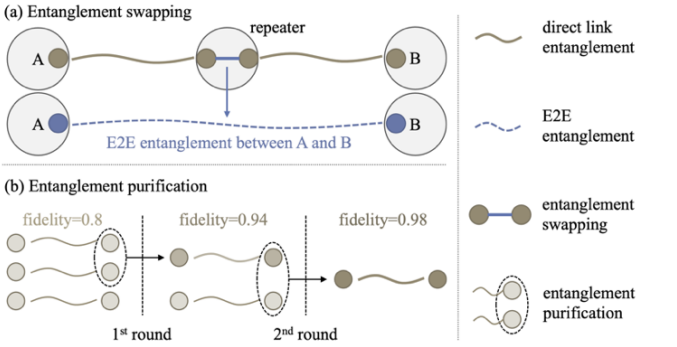



**Fig. 1.** Examples of (a) entanglement swapping and (b) entanglement purification.

In DQC, E2E entanglement enables the execution of remote gates through methods such as TeleData and TeleGate [27], enabling the transfer of qubits prior to gate execution or the direct execution of gates across remote QPUs. With TeleData, quantum information stored in a data qubit at one QPU, such as $|\varphi\rangle$ stored in $q_0$ at QPU 1 in Fig. 2(a), is teleported to a communication qubit at another QPU, such as $q_2$ at QPU 2. This teleportation utilizes the E2E entanglement between QPUs, specifically the Bell pair $|q_1 q_2\rangle$ in Fig. 2(a). Once $|\varphi\rangle$ is teleported to $q_2$, any operations previously requiring interaction between $q_0$ in QPU 1 and $q_2$ in QPU 2 can be performed locally on QPU 2. In Fig. 2(a) example, a remote CNOT gate is executed with the data qubit $q_0$ in QPU 1 as the control qubit and the data qubit $q_3$ in QPU 2 as the target qubit.

In contrast, TeleGate enables direct gate operations between data qubits in different QPUs without teleportation, provided E2E entanglement exists between the QPUs. Let us consider the same example of a remote CNOT gate with TeleData, where $q_0$ in QPU 1, serves as the control qubit and $q_3$ in QPU 2 acts as the target qubit. As shown in Fig. 2(b), this remote gate can be executed by performing two local CNOT gates: one between the data and communication qubit in each QPU, followed by a conditional operation on the data qubit based on the measurement outcome of the remote communication qubit.

Each method has its own advantages and limitations. TeleData is beneficial when a quantum state must be transferred to another QPU for subsequent local processing and execution. However, in distributed and multi-hop environments, repeated state relocation through teleportation may introduce additional communication and resource overhead. TeleGate, on the other hand, enables direct remote gate execution once E2E entanglement is established, avoiding explicit data-qubit migration between QPUs. However, TeleGate may impose stricter requirements on synchronization, gate fidelity, and remote operation coordination. Therefore, the trade-offs between TeleData and TeleGate depend on the circuit structure, hardware architecture, and distributed execution workflow.

In any case, both TeleData and TeleGate rely on end-to-end entanglement to operate effectively. This work focuses on the allocation and scheduling of such E2E entanglement resources for either approach. The selection between TeleData and



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



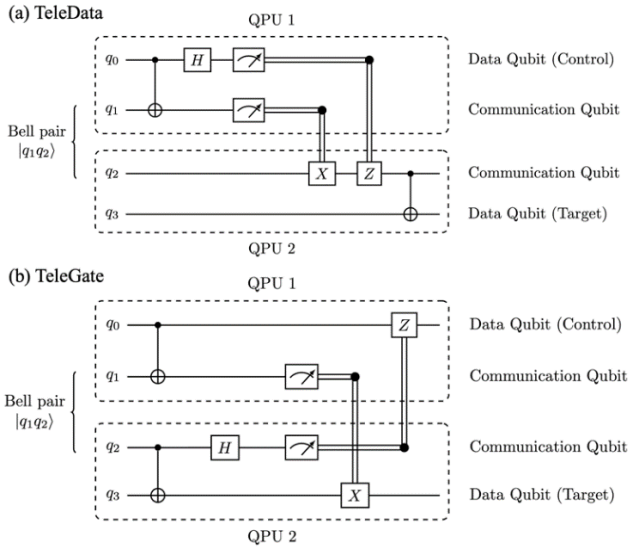



**Fig. 2.** Executing (a) TeleData or (b) TeleGate for a remote CNOT gate between QPU 1 and QPU 2.

TeleGate is assumed to be handled by upstream compilation and execution layers based on hardware and application requirements.

In a quantum data center, as the demand for executing remote gates increases, an overlay network for E2E entanglement distribution becomes essential. To facilitate this, each QPU must allocate at least one communication qubit to generate direct-link entangled pairs. Networking protocols then utilize these pairs to establish E2E entanglement between end nodes, serving as a fundamental communication primitive. Previous research on entanglement distribution often assumed a single E2E entanglement request at any given time, requiring only one path between two nodes [14], [19], [28], [29]. However, quantum data centers typically handle multiple simultaneous requests, as quantum circuits often involve numerous remote gates, each dependent on E2E entanglement. Some gates can be executed in an arbitrary order, while others must be performed sequentially, as they depend on the outcomes of preceding gates [30]. Additionally, certain gates are restricted to specific QPUs, whereas others can be executed on any QPU. These scenarios necessitate managing the allocation of entangled qubits between neighboring nodes, leading to variations in link capacities and emphasizing the importance of fair and efficient resource allocation [31], [32], [33].

### *C. Quantum Circuit Partitioning in DQC*

In a quantum data center, a complex circuit is partitioned into multiple sub-circuits for distributed execution. This subsection outlines the partitioning approach considered in this paper.

A perfectly distributable quantum circuit is one that can be divided into independent sub-circuits that either do not interact or interact only minimally with one another. In such cases, each sub-circuit can be assigned to a QPU, allowing each QPU to contribute independently to the overall computation without incurring communication overhead. However, this is rarely the case. In a typical quantum circuit, such as the Quantum Fourier Transform (QFT), each qubit tends to interact strongly with all

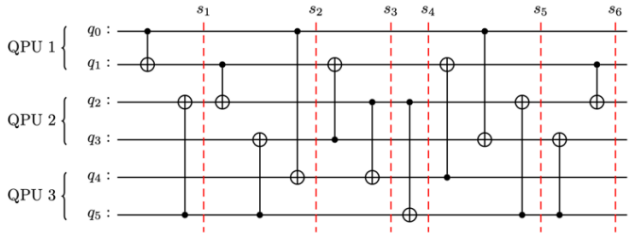



**Fig. 3.** An example partitioned quantum circuit.

other qubits [34]. Distributing such a circuit requires the use of a quantum compiler to determine the optimal partitioning, aiming to minimize the number of gates applied to qubits stored on different devices [35], [36], [37], [38], [39].

In this paper, we focus on partitioning a quantum circuit such that the execution order of gates within a sub-circuit is flexible, while the execution order of the sub-circuits themselves must be preserved. As shown in Fig. 3, an example quantum circuit is divided into six sub-circuits (i.e., $s_1$ to $s_6$). Each sub-circuit contains multiple quantum gates, including remote operations (e.g., CNOT gates) between different QPUs. The CNOT gates within a sub-circuit can be executed in any order because no single data qubit is shared by more than one gate within the sub-circuit. That is, during circuit partitioning, a dependency graph is constructed to identify dependency-resolved remote gates that can be executed concurrently. Remote gates without overlapping data qubits may be grouped into the same scheduling batch, providing flexibility for resource allocation and request scheduling. Each remote gate requires a certain number of E2E entangled pairs, with higher quantities preferred to meet fidelity requirements. This setup enables the management of batches of E2E entangled pair requests, either from groups of remote gates within a single sub-circuit or spanning multiple sequential sub-circuits with dependency relationships. The goal is to maximize the number of satisfied requests within a given time interval.

Note the assumption that remote gates within a sub-circuit can be executed in arbitrary order represents a simplified setting intended to isolate the management problem associated with remote-gate scheduling and entanglement allocation. In more general quantum circuits, intra-sub-circuit dependencies may exist due to shared data qubits or gate ordering constraints. In such cases, the proposed framework can naturally incorporate these dependencies by extending the dependency set and precedence constraints in the MILP formulation and refining the dependency graph used by the scheduling algorithms.

### *D. Related Work*

Research in entanglement routing, resource management for quantum networks, and DQC has gained significant attention in recent years due to its critical role in enabling scalable quantum systems and the ongoing advances in hardware.

Entanglement routing is a key aspect of quantum networks, focusing on the efficient distribution of entangled states across multiple network nodes. Routing strategies have been proposed to maximize the availability of E2E entangled pairs while considering the probabilistic nature of quantum link generation. For example, works such as [11], [12], [19], [22], [40], and [41]



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



introduce key components, architectures, and analyses for the quantum internet. Additionally, other studies focus on dynamic and adaptive routing protocols to establish E2E entanglement under resource constraints [14], [42], [43]. Some of these methods incorporate classical networking techniques, such as multipath and graph-based routing [28], [44], [45], [46]. These methods are refined to accommodate the unique constraints of quantum systems, including the imperative to maintain fidelity and mitigate the deleterious effects of decoherence. Despite these advancements, current techniques often assume idealized settings with limited scalability, leaving room for exploration in real-world quantum network configurations. Many existing studies simplify the problem by assuming the presence of only a single user request at any given time.

Efficient resource management is crucial for supporting distributed quantum tasks such as entanglement generation, teleportation, and remote gate execution, particularly in cases involving simultaneous user requests. Works such as [47], [48], [49], [50], and [51] have developed optimization models to allocate quantum resources effectively, considering factors like link capacity and fidelity requirements. Additionally, reinforcement learning has been explored in studies like [52], [53], [54], and [55] to address the dynamic nature of quantum resources. Despite these advances, practical implementations are constrained by the inherently short-lived nature of states and the necessity for precise synchronization across nodes.

DQC relies on partitioning large quantum circuits across multiple QPUs to overcome the qubit limitations of individual processors. Early work on DQC, such as [56], [57], [58], [59], [60], and [61], introduced foundational approaches for quantum circuit partitioning and remote execution using teleportation and entanglement swapping. More recent studies, including [62] and [63], have addressed optimizing communication overhead by developing strategies that reduce the number of remote gates and enhance the fidelity of shared entanglement. In addition, there are studies that focus on deadline scheduling in DQC [64]. While these efforts have laid a strong foundation, challenges remain in ensuring efficient coordination between QPUs and minimizing delays in large-scale quantum systems.

In particular, Bahrani et al [48] studied resource management and circuit scheduling in DQC interconnect networks, focusing on circuit-level problems such as QPU allocation, circuit assignment, and execution scheduling across heterogeneous DQC hardware. In contrast, this paper focuses specifically on dependency-aware entanglement-request scheduling and resource allocation after circuit partitioning and remote-gate extraction. Our framework abstracts remote operations as requests with dependency and fidelity constraints and studies lightweight request scheduling under dynamically available entanglement resources.

While significant progress has been made in these areas, to the best of our knowledge, no existing resource management framework specifically targets quantum data centers by abstractly modeling the components of a partitioned quantum circuit executed in a DQC environment. Furthermore, there is limited research focused on developing effective and efficient algorithms for resource management in DQC or quantum networks. Given the current scale of DQC architectures, particularly in quantum data centers, simple algorithms could suffice in many cases. However, most existing works emphasize sophisticated methods such as reinforcement learning or optimization solvers, which may compromise practicality due to their complexity, offering only marginal performance improvements. Additionally, few studies address the trade-offs between execution time and fidelity in resource-constrained environments. This paper addresses these gaps. This work focuses on the scheduling and allocation of entanglement resources after circuit partitioning and abstracts lower-level aspects such as compilation strategies, protocol selection, and physical entanglement-control mechanisms. Many prior approaches focus on different problem formulations, including adaptive routing, dynamic traffic control, or learning-based quantum network optimization, which are complementary to the request-driven scheduling abstraction considered in this work.

## III. TOPOLOGIES

In this section, we introduce potential network topologies for connecting QPUs in quantum data centers. A straightforward approach is to adopt topologies inspired by those used in early classical data centers [65], [66], and analyzed in the context of quantum networks [67], [68], [69], [70]. One example is the widely discussed entanglement switch [71], [72], designed for dynamically switchable entanglement networks, which, in its early stages, is expected to resemble a Clos network, a multistage circuit-switching network [73]. By adding stages, a Clos network reduces the number of cross points needed to construct a large switch, e.g., a hub. In this context, the distribution of E2E entanglement is analogous to circuit switching in classical networks, where a dedicated communication path is established between end nodes for the duration of the connection. Based on these principles, we explore two types of initial topologies for quantum data centers: DCell and BCube [74].

DCell and BCube were both originally developed to address the limitations of traditional hierarchical data center networks, offering scalability, fault tolerance, and manageability. These topologies are well-suited for quantum data centers, where efficient interconnectivity between QPUs is critical for DQC. DCell is defined recursively: a DCell at one level (referred to as Cell 0) acts as a building block for higher levels (Cell 1, Cell 2, etc.). For example, the topology shown in Fig. 4(a) is a 5-cell DCell topology. Note that Quantum Hubs (i.e., Q-Hubs) in a DCell mainly serve as internal optical switches within a cell while facilitating intra-cell switching. BCube, on the other hand, employs a multi-level recursive design inspired by the hypercube. In this topology, QPUs are interconnected through multiple layers of switches, creating a structure that supports robust connection paths and efficient fault recovery. As shown in Fig. 4(b), a 2-level BCube topology showcases the layered interconnections between nodes, enabling greater network capacity and manageability compared to traditional designs.

In both topologies, a central controller oversees network management, handling tasks such as routing and scheduling for entanglement distribution. DCell and BCube topologies are



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



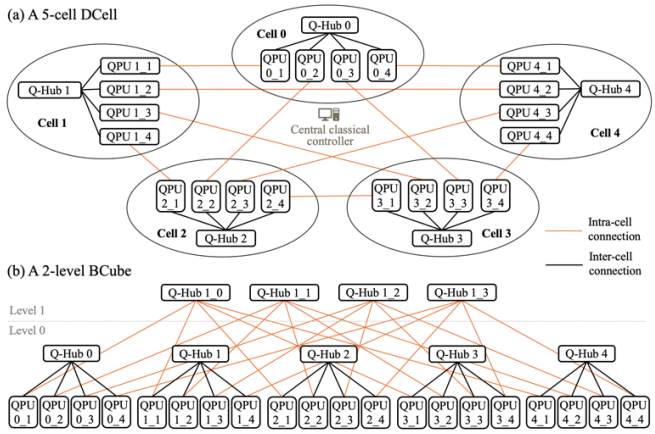



**Fig. 4.** DCell and BCube topologies for quantum data centers.

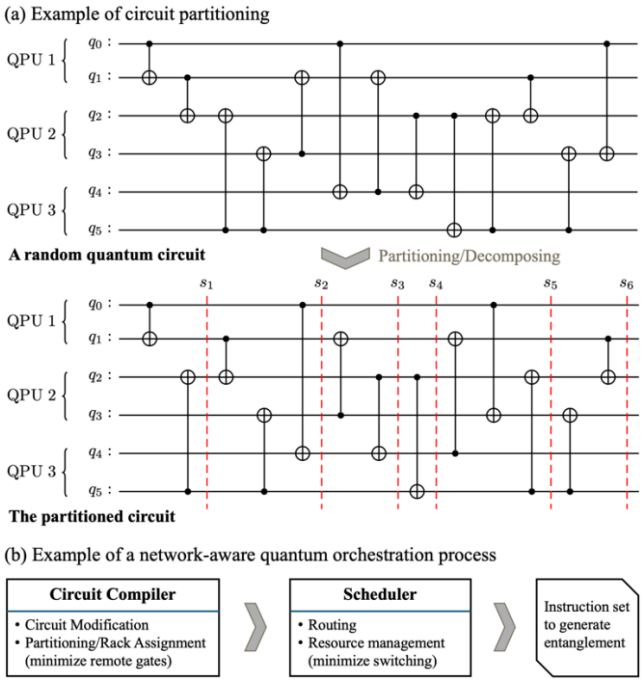



**Fig. 5.** Example of (a) circuit partition and (b) network-aware orchestration (this paper focusing on scheduler).

ideal candidates for the inter-rack connections. Nonetheless, we envision a hybrid approach in near-term quantum data centers: using DCell and BCube for inter-rack connectivity and employing simpler topologies, such as star or tree configurations, for intra-rack connections. In this paper, we focus on the inter-rack topologies DCell and BCube due to their high connectivity, rich path diversity, and multiple alternative routing paths, which are desirable properties for entanglement distribution and remote-gate execution in DQC environments. In contrast, more traditional topologies (e.g., fat-tree) generally exhibit more hierarchical structures and different contention characteristics. A broader comparison across topology designs is left for future work, which can also explore the integration of intra-rack connections to complement these designs.

We refer to the topologies formed by physical quantum channels, such as DCell and BCube, as *physical topologies*. To enable the distribution of E2E entanglement, a pre-establishment phase is required, during which each node generates a direct-link entangled pair with its neighbors, commonly known as an *entanglement link*. In contrast to the physical topology, we define the topology formed by entanglement links as the *instant topology*, reflecting its dynamic nature. Ideally, the instant topology would match the physical topology if direct-link entanglement generation were fully successful. However, in practice, the instant topology is often a subset of the physical topology, with link attributes reflecting varying entanglement link fidelities and capacities. The concept of the instant topology will be frequently referenced throughout this paper. Each scheduling interval operates on a newly generated instant topology and associated buffered entanglement resources. To simplify the model, unused Bell pairs and previously established E2E entanglement are not assumed to persist across scheduling intervals. Entanglement resources must be consumed before decoherence, which is abstracted through coherence-time constraint.

## IV. RESOURCE MANAGEMENT

Quantum data centers face unique constraints that do not apply to classical data centers. For instance, although a qubit can be transmitted to a remote node by encoding it into a photon's degree of freedom and sending it through a fiber link or free space, any loss of the photon due to attenuation or decoherence makes the information it carries irretrievable, as it cannot be copied or retransmitted [75]. Entanglement generation is probabilistic, resulting in dynamic connections and constantly shifting network topologies. Furthermore, entangled states are highly unstable, with lifetimes ranging from milliseconds to seconds [76]. These challenges necessitate the development of adaptive routing and scheduling algorithms to ensure efficient and reliable operation [13], [77], [78], [79], [80], [81], [82].

Therefore, a specialized resource management approach that accounts for quantum data centers' unique quantum properties, components, and architectures is crucial. In this context, we foresee two primary types of resource contention: one arising from the execution of dense sets of remote two-qubit gates within a quantum circuit (i.e., single-tenancy), and another stemming from running multiple quantum circuits concurrently on the same network (i.e., multi-tenancy). To effectively address resource contention in these scenarios, the approach must maximize the number of satisfied requests, referred to as network throughput, at any given time while ensuring fairness by proper request sequencing and satisfying request priority. Each request represents a remote operation requiring E2E entanglement resources, independent of the specific remote-gate implementation protocol. In this paper, the evaluation focuses on requests generated from a single partitioned circuit to study the core resource management behavior under controlled conditions. However, the proposed request-driven framework itself is not inherently restricted to single-tenancy and can also support requests originating from multiple concurrent workloads.

In a single-tenancy quantum data center, a quantum circuit is partitioned by a circuit compiler such that the order of remote gates within a sub-circuit is arbitrary, while the order of sub-circuits remains essential, as illustrated in Fig. 5(a). The



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



partitioned sub-circuits are then processed by a scheduler to generate an instruction set for the generation of E2E entanglement, as shown in Fig. 5(b). Multiple QPUs may be located in different network cells within the quantum data center. At any given time, the scheduler handles numerous sub-circuits containing remote gates that request E2E entangled pairs between QPUs. Many of these requests may rely on the same link to establish the required entangled pairs, but the available resources on each link are limited. The proposed framework operates at the resource-management layer after circuit partitioning and remote-gate extraction. Based on these configurations, the resource management process in a quantum data center is defined as follows.

The symbols and notations used in this paper are listed in Table I. As indicated, $G(V, E)$ represents the physical topology of a quantum data center, where each node $v \in V$ corresponds to a QPU, and each edge $e \in E$ represents a link connecting two QPUs. Each link $e$ has a maximum capacity of $C_e$, representing the maximum number of entanglement generations it can support between adjacent nodes. At any given time, the active link capacity, denoted as $c_e \leq C_e$, refers to the number of entangled pairs that have been successfully generated on the link $e$. Each direct-link entangled pair initially has a fidelity $f_e$. Note that $G(V, E)$, with zero entanglement edges removed, i.e., $c_e = 0$, , is called the instant topology.

At a given time, there exists a batch of requests from a group of remote gates, $R$, requiring E2E entangled pairs between distant QPUs for execution in a specific sequence. The batch of requests considered during a scheduling interval is determined by the upstream orchestration/runtime layer after dependency analysis. Specifically, only dependency-resolved remote-gate requests eligible for execution under the current entanglement-resource window are included in the current batch.

These remote gates can be from one or more sub-circuits within a single quantum circuit or across multiple circuits. Each remote gate request $r$ is characterized by two required QPUs, a duration $d_r$ to complete, a weight $w_r$, a maximum acceptable path length $h_r^{max}$ (i.e., the maximum number of hops), a minimum fidelity $f_r$ and a dependency set, $D_r$, containing the requests that need to be finished before it. Multiple paths are identified for each request $r$, denoted by $\mathrm{P}_r$, with each path $\rho$ having a hop length $h_r^{\rho}$. For simplicity, it is assumed that all entangled pairs share a uniform coherence time $t^{co}$, meaning all requests must be completed within this timeframe.

The management process for requests from remote gates in a quantum data center is summarized as follows:

1. Each node generates entangled pairs with its neighbors, forming an instant topology $G(V, E)$ in which each link's capacity is determined by its entanglement generation outcome. Each direct-link entanglement within a link is generated with a probability of $p$.
2. The network receives a batch of requests, $R$, from a group of sub-circuits, which contain remote gates.
3. A set of paths, $\mathrm{P}_r$, are identified for each request $r$. The number of paths $|\mathrm{P}_r|$ for each request should be limited to $|\mathrm{P}_r|^{max}$ to ensure fairness.
4. The best path selection and start time for each request are determined.
5. Go to 1 and repeat the process for the next batch of requests.

The challenge in this process involves selecting the optimal paths and arranging the correct sequence of requests in step 4.

TABLE I. SYMBOL TABLE

| Notation | Description |
|---|---|
| $G(V, E)$ | The network topology in a quantum data center, with nodes represented by set $V$ and links by set $E$. |
| $c_e$ | The number of currently available buffered Bell pairs on link $e \in E$ during a scheduling interval. |
| $f_e$ | Fidelity of a direct-link entanglement generated by link $e$. |
| $R$ | Set of requests from a group of remote gates. |
| $s_r$ | Start time for request $r \in R$. |
| $d_r$ | Simplified request duration associated with request $r$. |
| $w_r$ | Relative priority/importance weight associated with request $r$. |
| $D_r$ | Dependency set for request $r$. The request $r$ can only begin after all the requests in $D_r$ have been completed. |
| $h_r^{max}$ | Maximum number of hops for each path associated with request $r$ to satisfy the minimum entanglement rate for request $r$. |
| $f_r^{min}$ | Fidelity threshold for request $r$. |
| $\mathrm{P}_r$ | A set of paths that connect QPUs required for the request $r$. The paths are arranged in ascending order by their length. |
| $f_r^{\rho}$ | Fidelity that can be achieved by using the path $\rho \in \mathrm{P}_r$ for request $r$. |
| $f_e^{\rho}$ | Distilled link fidelity on link $e$ for path $\rho$. |
| $h_r^{\rho}$ | Length of path $\rho$ (number of hops) for request $r$. |
| $c_r^{\rho}$ | The capacity required for purification and meeting the fidelity requirement of request $r$ over path $\rho$. |
| $\xi_r^{\rho}$ | Entanglement rate for request $r$ along path $\rho$. |
| $p$ | Direct-link entanglement generation success probability. |
| $q$ | Entanglement swapping success probability. |
| $t^{co}$ | The coherence time of the Bell pairs in a quantum data center, which serves as the deadline for completing all requests in $R$. |

## V. OPTIMAL AND SIMPLER STRATEGIES

As previously discussed, quantum data centers face unique constraints that are not present in classical data centers,



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



requiring specialized strategies. In this section, we define the problem and present the proposed models and algorithms for resource management in DQC within quantum data centers.

### *A. Optimization by Mixed Integer Linear Programming*

**Given** 1) a set of requests for E2E entanglement, each with an associated weight, fidelity requirement, a dependency set, and duration, 2) a set of candidate paths for each request, where each path has a certain length, fidelity level, and requires a specific number of purified Bell pairs to meet fidelity targets, together with 3) a network graph with limited link capacities, and a coherence time that bounds how long any request can take to complete, the problem is **to determine** 1) which path should be selected for each request, and 2) when each request should start, **such that** the total weighted number of successfully served requests (called the weighted throughput) is maximized, while ensuring 1) no link exceeds its capacity, 2) selected paths meet fidelity requirements, 3) dependencies are respected, 4) paths are not too long, and 5) all requests finish in time.

With that, we define the allocation MILP problem as follows:

$$\text{maximize} \sum_{\forall r \in R} \sum_{\forall \rho \in \mathrm{P}_r} x_r^\rho \cdot w_r \cdot \xi_r^\rho \tag{1}$$

subject to

$$\sum_{\forall r \in R} \sum_{\forall \rho \in \mathrm{P}_r : e \in \rho} x_r^\rho \cdot c_r^\rho \le c_e, \forall e \in E \tag{2}$$

$$s_i + d_i \le s_r, \forall r \in R, \forall i \in D_r \tag{3}$$

$$h_r^\rho \le h_r^{max}, \forall \rho \in \mathrm{P}_r, \forall r \in R \tag{4}$$

$$f_r^\rho \ge f_r^{min}, \forall \rho \in \mathrm{P}_r, \forall r \in R \tag{5}$$

$$\max_{\forall r \in R}(s_r + d_r) \le t^{co} \tag{6}$$

The objective of the optimization problem in (1) is to maximize the weighted network throughput for a given set of requests, which need to be satisfied within one coherence time. The request weight $w_r$ represents the relative importance or priority of a remote-gate request in the scheduling process. Depending on the application scenario, it may reflect factors such as communication demand, execution priority, or the significance of the corresponding circuit operation. Moreover, the coherence time here is used as a resource-validity and scheduling-feasibility bound rather than a fixed cycle. Requests must complete before decoherence renders the generated entanglement resources unusable. The weighted throughput ensures that requests with higher weights are executed with choices of shorter path or more paths (i.e., higher rate for higher weight) within the constraints. The decision variables are $\{x_r^\rho, s_r\}$ where $x_r^\rho \in \{0,1\}$ denotes the selection of the path $\rho$ for request $r$ (with 1 indicating $\rho$ is selected, otherwise, 0), and $s_r$ denotes the start time of the request $r$, which ensures the predefined request precedence. The primary decision variable is $x_r^\rho$, which determines whether request $r$ is scheduled. The variable $s_r$ is additionally introduced to enforce scheduling feasibility. $w_r$ indicates the weight of the request $r$. $\xi_r^\rho$ denotes the resulting E2E entanglement rate for the request $r$ when using the path $\rho$ and can be computed by $\xi_r^\rho = q^{h_r^\rho - 1}$ (i.e., the success probability of a sequence of swapping operations). Note that the direct-link entanglement generation success probability $p$ is implicitly reflected in the generated instant topology and the resulting available link capacities. The objective formulation explicitly models the E2E entanglement swapping success probability.

The constraint in (2) ensures the selected paths together do not exceed the capacity of any link in any path. $c_r^\rho$ denotes the capacity path $\rho$ required for purification to meet the fidelity requirement of the request $r$ (i.e., each link in that path $\rho$ has a need to have available capacity greater than $c_r^\rho$).

Before computing $c_r^\rho$, we need to know the minimum fidelity each link in the path $\rho$ needs to have, to achieve $f_r^{min}$, given the number of swapping needed. That is, we need to consider fidelity changes after a number of swaps (fidelity as a function of path length). For a path with a length of $h_r^\rho$ and distilled link fidelity for the path $\rho$, $f_e^\rho$, the resulting fidelity of the E2E Bell pair path $\rho$ can achieve (after undergoing $l_r^\rho - 1$ nested swap operations) can be expressed as [83]:

$$f_r^\rho = \frac{1}{4} + \frac{3}{4}\left(\frac{\phi(4\eta^2 - 1)}{3}\right)^{l_r^\rho - 2}\left(\frac{4f_e^\rho - 1}{3}\right)^{l_r^\rho - 1} \tag{7}$$

where $\phi$ is the two-qubit gate fidelity and $\eta$ is the measurement fidelity of the swapping operation. To compute $f_e^\rho$, we need to consider the purification relationship between resulting fidelity and direct-link Bell pair fidelity $f_e$. Without loss of generality, by considering bit flip errors only, the resulting fidelity after purification using two Bell pairs can be computed by [42]:

$$F(f_1, f_2) = \frac{f_1 f_2}{f_1 f_2 + (1 - f_1)(1 - f_2)} \tag{8}$$

If we consider the uniform fidelity of Bell pairs on the same edge, then, $f_1 = f_2$, and the formula becomes $F(f) = \frac{f^2}{2f^2 - 2f + 1}$. In principle, this process can be applied recursively to achieve arbitrarily high fidelities by consuming a number of Bell pairs. Using a recursive approach can determine the round of purifications (or the number of Bell pairs) needed to achieve the target fidelity, which thus computes $c_r^\rho$. For effective purposes, an iterative approximation can be applied. More details can be found in Appendix A.

Let us assume we use the same number of direct-link Bell pairs in each link of the path. $\rho$. That is, $c_r^\rho = n + 1$ where $n$ is the round of recursive purification conducted. Then, we have the same distilled link fidelity on each link for the path $\rho$, i.e., $f_e^\rho$. Then, with (8), we have:

$$f_e^\rho = \begin{cases} F_n = F_{n-1}(F_{n-2}, f_e), & n \ge 2 \\ F_1 = F(f_e, f_e), & n = 1 \\ F_0 = f_e, & n = 0 \end{cases} \tag{9}$$

where $F_{n-1}$ is a recursive form based on (8) when n>=2. $F_1$ is the same as (8) and $F_0 = f_e$. Let $f_r^\rho = f_r^{min}$ and solve for $n$ using (7) and (9), we get $c_r^\rho = n + 1$. Note that the current formulation abstracts storage-related decoherence effects through $t^{co}$ constraint and does not explicitly model memory occupancy evolution or fidelity degradation during storage. Incorporating persistent entanglement storage, memory-aware



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



scheduling, and cross-batch entanglement reuse is an important direction for future work.

The constraint in (3) ensures that requests must respect predefined sequences and dependencies. If request $i$ precedes request $r$ (i.e., $i \in D_r$), the completion time of the request $i$ (i.e., $s_i + d_i$) must be less than or equal to the start time of the request $r$, $s_r$. The constraint in (4) ensures the practicality of each path by setting a maximum path length for each path associated with the request $r$ to satisfy the minimum entanglement rate for the request $r$. The constraint in (5) ensures the fidelity achieved by using the path $\rho$ for request $r$ is greater than the fidelity threshold for the request $r$. The constraint in (6) ensures the latest completion time of all requests is less than the coherence time of the Bell pairs in a quantum data center (i.e., the deadline for completing all requests of the current batch).

Note that in the current formulation, the request duration $d_r$ is modeled as a fixed abstract parameter to simplify the scheduling problem and isolate the impact of resource allocation and scheduling. While realistic execution latency may depend on path length, swapping operations, purification rounds, and classical coordination overhead, these effects are partially reflected through fidelity, path-length, and resource-consumption constraints rather than explicit path-dependent service times. An extension can be incorporated by defining $d_r$ as a function of the selected path and associated operations.

Furthermore, it is worth noting that if the batch of requests is time-sensitive, where we want to minimize the maximum completion time (makespan) while maximizing throughput, the objective function can be changed to reflect the dual objectives (note that the makespan in this work should be interpreted as an abstract scheduling-feasibility bound rather than a detailed latency-accurate execution model). These two objectives often conflict: maximizing throughput could increase execution time due to resource saturation, and minimizing execution time might reduce the opportunities to maximize throughput. Thus, if makespan matters so much that throughput can be compromised (or balanced), it can be modeled as a dual-objective optimization problem that simultaneously maximizes throughput and minimizes execution time. The second objective can be expressed as

$$\text{minimize } O_2 = \text{minimize} \max_{\forall r \in R}(s_r + d_r) \tag{10}$$

Given the dual objectives, we can model them as a weighted sum objective:

$$\text{maximize } \alpha O_1 - \beta O_2 \tag{11}$$

where $O_1$ is the first objective function $O_1 = \sum_{\forall r \in R} \sum_{\forall \rho \in \mathrm{P}_r} x_r^\rho \cdot w_r \cdot \xi_r^\rho$ in (1). $\alpha$ and $\beta$ are weights indicating the importance of maximizing throughput and minimizing execution time, respectively.

Alternatively, we can use a $\varepsilon$-constraint model where one objective is optimized while the other is included as a constraint to be satisfied:

$$\text{maximize } O_1 \;\; s.t. \; O_2 \le \varepsilon \tag{12}$$

or vice versa, where $\varepsilon$ is a threshold that you are willing to accept for the constrained objective.

Nonetheless, in the rest of this paper, for simplicity, we consider throughput the sole objective as shown in (1) and include global makespan as a constraint to be satisfied. The threshold for the constrained makespan is that it is always less than $t^{co}$, which makes the allocation feasible.

### *B. Resource Optimization with Non-overlapping Constraint*

In certain scenarios within a quantum data center, it is essential to ensure that no two requests are scheduled to overlap in time on the same machine. For example, consider a scenario where a quantum data center is tasked with processing several remote gates that require the utilization of a specialized QPU. Each request needs to execute quantum algorithms that involve complex calculations such as factoring large numbers, simulating molecular interactions, or optimizing financial portfolios. These tasks are computationally intensive and sensitive to the precise control of quantum states, necessitating dedicated access to the QPU resources.

If two requests, $i$ and $r$, are assigned to the same QPUs and their execution times overlap, it could lead to quantum state interference, erroneous computation results, or degradation in system performance. For instance, if the remote gates of request $i$ involves preparing and measuring quantum states that are susceptible to decoherence over time, and gates in request $r$ concurrently starts an operation that unintentionally impacts these states due to shared resource contention, both requests might fail to achieve their goals. To manage this, we can utilize a non-overlapping constraint expressed as below, which can be added to (1)'s constraint set when necessary:

$$\begin{gathered} s_i + d_i \le s_r \;\; or \;\; s_r + d_r \le s_i, \forall i, r \in R, i \ne r \\ if \; \{a_i, b_i\} \cap \{a_r, b_r\} \ne \emptyset \end{gathered} \tag{13}$$

where $a_i$ and $b_i$ are the source node and the destination node of the request $i$, respectively. This constraint ensures that for each pair of requests assigned to the same QPUs, one of the following conditions must hold: either the completion time of the first request must be less than or equal to the start time of the second request, or the completion time of the second request must be less than or equal to the start time of the first request. The "if" condition checks if there is any common node between the sets of source and destination nodes of requests $i$ and $r$. This captures any overlap among the source and destination nodes of the two requests.

To model the logical operations in (13) so as to implement this constraint effectively in a MILP model, we can use a big-M model to linearize the disjunction [84]. We can introduce a binary variable indicating whether the request $i$ is scheduled before the request $r$, $y_{ir} \in \{0,1\}$. It equals 1 when the request $i$ is in the dependency set of the request $r$. Then, the constraints can be:

$$\begin{gathered} s_i + d_i \le s_r + M(1 - y_{ir}), s_r + d_r \le s_i + M y_{ir} \\ \forall i, r \in R, i \ne r \end{gathered} \tag{14}$$

where M is a sufficiently large number that makes the constraint non-binding when not required.

### *C. Scheduling by Simpler Strategies*

MILP may be appropriate if the primary goal is to determine an optimal allocation of paths and resources for a static set of requests and constraints without factoring in the time required to compute the solution. It is particularly effective when relationships can be linearized, and the environment remains stable. However, in a quantum data center, the instant topology



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



**Algorithm 1.** Weighted Group Least Resource First (WGLRF)

**Input:** Graph $G$, Requests $R$ and the number of requests per group $n$

**Output:** Assignment of paths to requests with start times

```
assignment ← empty dictionary
# Sort requests by weighted topological order
sorted_requests ← weighted_topological_sort_requests(R)
for i from 0 to len(sorted_requests) step n do
    group_requests ← sorted_requests[i:i + n]
    for each request r in group_requests do
        earliest_start ← get_earliest_start_time(assignment)
        min_resource_usage ← infinite
        for each path ρ in r['paths'] do
    resource_usage ← get_required_bell_pairs(G[u][v]['fidelity'],
                r['min_fidelity'], len(ρ))
            # Check path feasibility and path optimality
            if G[u][v]['capacity'] ≥ resource_usage and
                resource_usage < min_resource_usage then
                # Assign path to request with the earliest start time
                append (r, ρ, earliest_start) to assignment
                G[u][v]['capacity'] −= resource_usage
return assignment
```

changes dynamically with each coherence time, and incoming request batches are inherently dynamic. More critically, in the current NISQ era, the coherence time is very short, necessitating fast solutions to address the current instant topology before it changes. To address these challenges, we propose a set of efficient, simple scheduling algorithms as follows.

We first propose the Least Resource First (LRF) scheduling algorithm, which prioritizes paths that consume the least resources. It selects paths that leave the maximum resources available post-allocation while respecting the dependency relationships between requests. Specifically, it evaluates multiple paths for each request to identify the one that minimizes resource usage, thereby reducing the impact on the overall system capacity. LRF prevents over-commitment to resource-intensive requests, reducing the risk of scheduling bottlenecks. Its solution space is relatively small, making it exceptionally efficient.

To enhance performance by expanding the solution space, we extend LRF to propose the Group LRF (GLRF) algorithm. GLRF processes a group of requests (e.g., two requests per group) simultaneously, selecting paths collectively that leave the most resources free after allocation. This broader exploration of the solution space provides better optimization compared to processing requests individually with LRF. For instance, when evaluating solutions for a batch containing $k$ requests, GLRF considers $C(k,2)$ possible combinations, while LRF only evaluates $k$. Here, $C(k,2)$ represents the "given k choose 2" combinatorial. By assessing the collective impact of multiple requests on resource allocation, GLRF identifies more effective solutions. Prior to path selection, requests are sorted based on their precedence sets using a directed acyclic graph (DAG) by topological sorting.

Considering the importance of weighted throughput (not just throughput), we further introduce the Weighted GLRF (WGLRF) algorithm, which incorporates request weights into GLRF. WGLRF prioritizes groups of requests with higher total weights, where group weight is the sum of the weights of all requests in the group. This improves performance by emphasizing high-priority requests. Like GLRF, WGLRF explores group-based solutions but adds a weight-based prioritization step. The three algorithms, LRF, GLRF, and WGLRF, are computationally efficient, with WGLRF closely approximating optimization results in many scenarios.

Since WGLRF encompasses the functionality of both GLRF and LRF, we present only the pseudocode for WGLRF in Algorithm 1. For simplicity, the pseudocode shows the assignment of a single path to each request. However, it can be easily adapted to assign multiple paths to a single request when additional link capacity is available after all requests in the batch have been assigned one path. Weighted topological sorting is used to order non-dependent vertices by weight during topological sorting [85].

WGLRF naturally promotes a more balanced distribution of load across system resources, preventing overuse of any particular link by focusing on minimizing resource usage. In multi-request or multi-tenant environments, this ensures that no single request monopolizes resources, improving overall fairness and request satisfaction. The WGLRF strategy is conceptually straightforward and can be implemented using simple algorithms to evaluate resource usage for each path.

Note that, in all methods, paths are evaluated to ensure they meet the fidelity requirements of a request before being assigned. Paths that cannot achieve the required fidelity through entanglement purification are excluded from consideration. Additionally, requests with unresolved dependencies are temporarily deferred until their dependencies are resolved, and requests that cannot be satisfied within the current timeframe are carried over to the next batch.

## VI. EVALUATION

To evaluate the performance, we use the simple First Come First Served (FCFS) algorithm as a lower-bound benchmark and the CPLEX MILP solver [86] for optimization problem (1) as an upper bound. FCFS processes requests based on their dependencies and weights, starting with those that have none and progressing to those with increasing levels of dependency.

### *A. Time Complexity*

Branch-and-bound is widely used in integer programming to solve MILP problems, including CPLEX. The method systematically divides the problem (branching) and uses solution bounds of subproblems (bounding) to gradually narrow down the search for the optimal solution. In our case, each branching point (or decision) in the solution space can potentially result in the generation of two new subproblems or branches. This bifurcation occurs because each variable at a decision point can lead to two distinct paths: one where the variable is included in the solution and one where it is not. As a result, the number of subproblems, or nodes in the decision tree, can double with each additional variable. This exponential growth in the number of nodes that need to be evaluated reflects the inherent complexity of the problem.

Given that $n$ potential paths are identified for the requests in a



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



batch, with each path requiring a binary decision (branch), the total number of possible combinations of these decisions is $2^n$. This corresponds to the total number of leaf nodes in a fully expanded binary tree, with each node representing a subproblem that may need to be resolved in the worst case. Consequently, the time complexity of a branch-and-bound-based solver is $O(2^n)$. The complexity of handling dependency constraints is negligible compared to $2^n$ and thus is omitted. Let $k$ represent the number of requests and $m$ denote the constant number of potential paths per request. Since the total number of paths $n = k * m$, the complexity can be represented as $O(2^k)$.

In FCFS, requests are processed sequentially based on the order of arrival and the resolution of their dependencies. This can be achieved using topological sorting, which has a time complexity of $O(k + \delta)$, where $\delta$ represents the number of directed edges in the dependency graph. Each edge is a precedence constraint, meaning that one request must be completed before another can begin.

LRF not only processes requests based on their dependency resolution but also evaluates which path for each request minimizes resource usage. This involves a comparative analysis of all paths per request to determine the one that leaves the most resources free. It involves determining the least-resource path among all paths of a request, together with dependency checks. This process sequentially evaluates each path until a valid one is found. Assuming a uniform distribution of paths across requests, the complexity is $O(k * m + \delta)$, thus $O(k + \delta)$.

In GLRF, requests are grouped based on criteria like minimum dependencies. For each group, GLRF evaluates multiple paths for each request in the group to determine which combination of paths across all requests in the group minimizes the total resource usage. First of all, we still need topological sorting, which is $O(k + \delta)$. Then, for fixed group sizes, resource evaluation for each group has a complexity related to the combinations of requests within each group. For each group, $C(k, g) = O(k^g)$ combinations may be considered, where $g$ is the number of requests within the group. Including the sorting and grouping, the complexity can be represented as $O(k + \delta + k^g * m * k/g)$. Since $m$ and $g$ are constant, the complexity is $O(k + \delta + k^{g+1})$ and thus $O(k^{g+1})$.

WGLRF is similar to GLRF, but adds a preliminary step of weighting groups, which might involve additional complexity depending on how weights are calculated and how groups are formed. This can add an $O(klogk)$ component for sorting by weight before topological sorting. Thus, the complexity of WGLRF is $O(klogk + k^{g+1})$. Since $O(klogk)$ grows slower than $O(k^{g+1})$ for any $g \geq 1$, it can be simplified as $O(k^{g+1})$.

In short, as expected, the optimization solver's complexity is exponential for the branch-and-bound method to explore the solution set. All other algorithms are polynomial.

### *B. Simulation Settings*

To evaluate the performance of the proposed simple approaches in the aforementioned configuration, we conduct simulations using NetworkX [87] to model the dynamic network graph. A uniform-distribution-based random number generator is used to generate random circuits that produce weighted requests with random dependencies. For simplicity, the direct-link entanglement generation success probability is set to a constant value of 0.8, introducing variability in link capacities during the simulation. Note that we use simulations as current quantum hardware lacks the scale and reliability for dynamic network experiments; real-device emulations remain future work, with similar trends anticipated.

We simulate the 5-cell DCell and 2-level BCube topologies separately to assess the methods' compatibility with different network structures. Both topologies are configured with the same number of end nodes (QPUs), complemented by quantum hubs positioned to facilitate connectivity. Remote gate pairs are generated randomly among QPUs, with switches and hubs excluded as end nodes. The requests generated from these remote gates correspond to E2E entangled pairs required between the selected QPU pairs. Network graphs are constructed with edge attributes, including maximum capacity and initial fidelity of the direct-link entanglement, while vertices are categorized as either QPUs or switches.

A random partitioned quantum circuit, such as the one shown at the bottom of Fig. 5(a), is generated to form the basis of the simulation. Requests are grouped into batches corresponding to the partitioned circuit: gates in a sub-circuit on the right depend on the completion of all gates in sub-circuits to its left. Each gate generates a request, which includes two randomly selected QPUs requiring an E2E entangled pair with a minimum fidelity. For each request, five potential paths are identified using a $k$-shortest path algorithm [88]. The proposed algorithms are then implemented to determine the optimal combination of paths for all (or as many as possible) requests in a batch, considering the link capacity constraints and fidelity requirements.

The simulation iterates over a range of request sets, with each set varying by the number of requests, their complexity, and interdependencies. Instead of merely counting the total number of requests processed, we evaluate the weighted throughput, which measures the "value" of requests successfully scheduled and completed, reflecting the priority and importance of requests handled by the scheduler. Additionally, fairness in scheduling is assessed using Jain's Index [89], a widely used fairness metric that measures how evenly resources and scheduling opportunities are distributed among requests. Jain's Index is calculated as the square of the sum of allocated resources divided by the number of requests multiplied by the sum of the squares of the resources allocated to each request. A higher Jain's Index indicates a more equitable allocation of resources, which is especially important in scenarios where requests vary significantly in priority and requirements.

### *C. Simulation Results*

Fig. 6 presents the performance of all the discussed schedulers across multiple dimensions in both DCell and BCube topologies. Details are in the subsequent paragraphs. Note that for all simulations, when a factor is not explicitly mentioned or is not a variable on the x-axis, its default value is as follows: the link capacity is set to 20, meaning there are 20 entangled pairs available between two adjacent nodes; the minimum fidelity requirement for a request is randomly generated between 0.5 and 1; the initial fidelity of a direct-link entangled pair is 0.9; the maximum number of paths selected per request is 3; the maximum number of hops for a path is 10; and the number of requests per batch is 20. The default



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



parameter values were selected to represent practical operating conditions for quantum networking and DQC scenarios while maintaining meaningful scheduling and resource-contention tradeoffs. The selected ranges were also guided by commonly used assumptions in prior quantum networking and entanglement-routing literature.

As shown in Fig. 6(a) and (e), for randomly generated circuits on DCell and BCube, respectively, the optimization solver consistently achieves the best performance. However, the weighted throughput achieved by WGLRF closely approximates that of the optimization solver, regardless of the number of requests in a batch. As anticipated, GLRF performs slightly worse than WGLRF but outperforms LRF. All these algorithms demonstrate better performance compared to FCFS.

As shown in Fig. 6(b) and (f), the performance degrades significantly as the average minimum fidelity requirement for requests increases. This decline is due to fidelity loss during entanglement swapping, which uses more entangled pairs to meet the fidelity requirement, reducing available resources for other requests. Despite this, WGLRF continues to closely approximate the optimization solver in terms of weighted throughput, and the performance ranking remains unchanged.

Furthermore, as shown in Fig. 6(c) and (g), while WGLRF cannot reach the solver performance when the link capacity is low, its performance increasingly approaches that of the solver as the link capacity grows. The performance ranking for the methods remains consistent. Then, as shown in Fig. 6(d) and (h), WGLRF is slightly inferior to the optimization solver at low initial fidelity for direct-link entanglement, but its performance becomes nearly optimal as the initial link fidelity improves.

Moreover, the results for requests decomposed from multiple real quantum circuits from QASMBench [90], as shown in Fig. 6(i–p). The evaluated benchmark circuits include dnn (2 qubits, 226 gates, 42 CNOT gates), error_correctiond3 (5 qubits, 114 gates, 49 CNOT gates), hhl (7 qubits, 689 gates, 196 CNOT gates), ising (10 qubits, 480 gates, 90 CNOT gates), and pea (5 qubits, 98 gates, 42 CNOT gates). These circuits provide varying levels of circuit size and remote-gate complexity for evaluating the proposed framework. Larger circuits and higher CNOT counts generally generate more remote-gate requests and increase resource contention during distributed execution. The results obtained using real benchmark circuits closely follow the trends observed with generated circuits. In particular, WGLRF typically remains within approximately 5-10% of the optimization baseline across most evaluated scenarios, while exhibiting similar qualitative behavior under varying conditions for both DCell and BCube topologies.

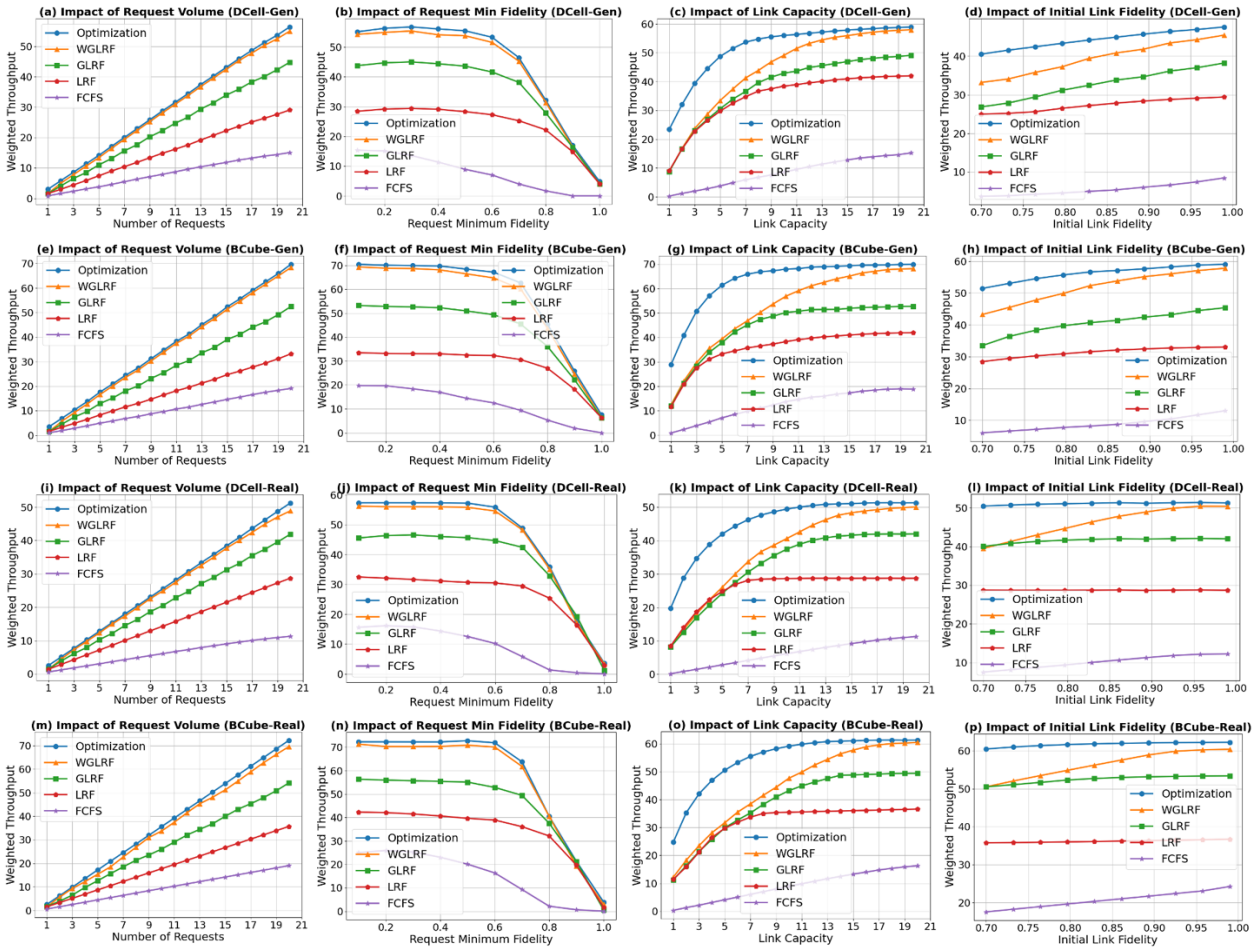



**Fig. 6.** Simulation results of all methods in DCell and BCube with generated and real quantum circuits.



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



We further conducted simulations to evaluate the fairness and runtime of the schedulers. As shown in Fig. 7(a), all algorithms exhibit similar fairness, which is expected due to the presence of dependency relationships among requests. WGLRF and GLRF exhibit slightly improved fairness as the number of requests increases. In Fig. 7(b), the execution times of WGLRF, GLRF, LRF, and FCFS are significantly lower than those of the optimization solver. Among these, the runtime increases slightly from FCFS and LRF to WGLRF. Fig. 7(c) and 7(d) present results for requests decomposed from the same real circuit set as above, which are highly similar to those in Fig. 7(a) and 7(b).

Overall, WGLRF closely approximates the results of the optimization solver when link capacity is relatively sufficient. As link capacity increases, the performance of WGLRF progressively aligns with that of the solver. Across varying minimum fidelity requirements, WGLRF remains close to optimal. At low initial link fidelity, WGLRF performs slightly below optimal, but it approaches optimal performance as the link fidelity increases. All algorithms demonstrate similar fairness, influenced by the dependency relationships. Additionally, all proposed schedulers (i.e., WGLRF, GLRF, LRF) execute significantly faster than the optimization solver, with WGLRF maintaining performance comparable to the optimization solver in most scenarios. Moreover, our simulations indicate that performance in BCube generally surpasses that in DCell.

These results demonstrate that, despite the tradeoff between execution time and weighted throughput, lightweight approaches achieve strong performance. Compared to optimization solvers, these simple solvers provide a practical solution for managing the large number of remote gates. Efficiency gains in processing batches of requests can substantially enhance overall performance in DQC. Consider a circuit containing thousands of remote gates, such efficiency gains per batch become critical for executing the entire circuit efficiently.

## VII. CONCLUSION

With the limited computational power of a single quantum computer, DQC in quantum data centers has strong potential in areas such as cybersecurity, sensing, and pharmaceuticals. However, quantum data centers may contain many QPUs executing numerous remote gates, each requiring entanglement between distinct QPUs with specific fidelity constraints. To address congestion and resource contention during E2E entanglement distribution, we propose a resource management framework tailored for quantum data centers. The problem is formulated as an MILP optimization model to serve as an upper-bound benchmark. We further introduce efficient scheduling algorithms that, in many scenarios, perform comparably to the optimization solver. Although there is a tradeoff between execution time and network throughput, simulations demonstrate that the proposed simple methods closely approximate the solver's results in most cases. The results suggest that lightweight heuristic scheduling approaches can achieve performance close to the optimization baseline under the evaluated DQC resource-management settings while maintaining substantially lower computational complexity. These observations indicate that lightweight heuristics represent practical scheduling alternatives for the considered request-driven DQC scheduling model.

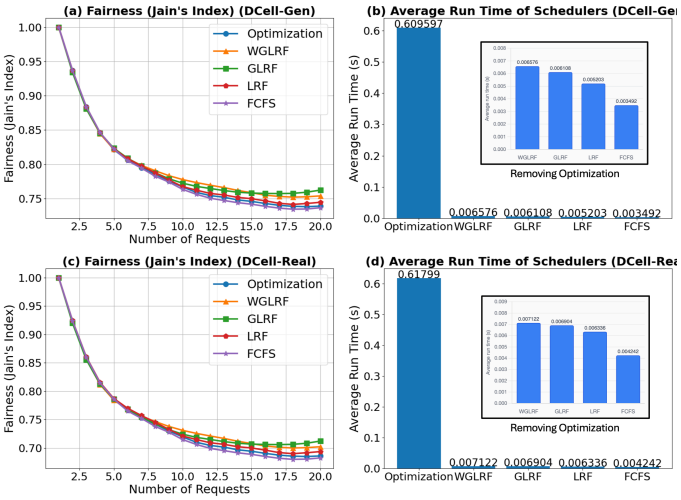



**Fig. 7.** Simulation results of fairness and execution time.

## APPENDIX A

Here, we develop an iterative approximation for $n$, the number of rounds of purification required to achieve a target fidelity $f_i$, by examining the behavior of the fidelity purification function based on (8), assuming the same initial Bell pair fidelity, represented as follows:

$$F_{n+1} = \frac{{F_n}^2}{{F_n}^2 + (1-F_n)^2} \tag{A1}$$

(A1) can be simplified under the assumption that $F_n$ is close to 1, which is usually the case in practical scenarios where high fidelity is needed, and initial fidelities are also reasonably high.

When $F_n$ is close to 1, $(1-F_n)$ is small. Let us denote $x_n = 1 - F_n$. The update formula then simplifies to:

$$\begin{aligned} x_{n+1} = 1 - F_{n+1} &= 1 - \frac{{F_n}^2}{{F_n}^2 + (1-F_n)^2} \\ &= \frac{(1-F_n)^2}{{F_n}^2 + (1-F_n)^2} \approx (1-F_n)^2 \end{aligned} \tag{A2}$$

Thus:

$$x_{n+1} \approx x_n^2 \tag{A3}$$

Given (A3), we can see that the error rate decreases quadratically with each round. If we start with $x_0 = 1 - F_0$, after $n$ rounds, $x_n$ will be approximately:

$$x_n \approx (1-F_0)^{2^n} \tag{A4}$$

To achieve a target fidelity $f_i$, we need:

$$1 - f_i \le (1-F_0)^{2^n} \tag{A5}$$

Taking logarithms on both sides, we get:

$$\log(1-f_i) \le 2^n \log(1-F_0) \tag{A6}$$

Thus:

$$n \ge \frac{\log(\log(1-f_i)/\log(1-F_0))}{log2} \tag{A7}$$



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



## REFERENCES

[1] H. Neven, "Meet Willow, our state-of-the-art quantum chip," Google. Accessed: Aug. 29, 2025. [Online]. Available: https://blog.google/technology/research/google-willow-quantum-chip/

[2] J. Gambetta, "The hardware and software for the era of quantum utility is here." Accessed: Aug. 29, 2025. [Online]. Available: https://www.ibm.com/quantum/blog/quantum-roadmap-2033

[3] Z. Yang, M. Zolanvari, and R. Jain, "A Survey of Important Issues in Quantum Computing and Communications," *IEEE Commun. Surv. Tutor.*, vol. 25, no. 2, pp. 1059–1094, 2023, doi: 10.1109/COMST.2023.3254481.

[4] M. Pompili *et al.*, "Experimental demonstration of entanglement delivery using a quantum network stack," *Npj Quantum Inf.*, vol. 8, no. 1, pp. 1–10, Oct. 2022, doi: 10.1038/s41534-022-00631-2.

[5] R. Nejabati *et al.*, "Quantum Data Center Interconnect - A Cisco Vision," Cisco Research. Accessed: Aug. 29, 2025. [Online]. Available: https://research-strapi-s3.s3.us-east-2.amazonaws.com/QN_4_QDC_2024_Release3_2_7c0273901f.pdf

[6] IBM, "First IBM Quantum Data Center in Europe Opens; Will Include IBM's Most Performant Quantum Systems," IBM Newsroom. Accessed: Aug. 29, 2025. [Online]. Available: https://newsroom.ibm.com/2024-10-01-first-ibm-quantum-data-center-in-europe-opens-will-include-ibms-most-performant-quantum-systems

[7] D. Cuomo, M. Caleffi, and A. S. Cacciapuoti, "Towards a distributed quantum computing ecosystem," *IET Quantum Commun.*, vol. 1, no. 1, pp. 3–8, 2020, doi: 10.1049/iet-qtc.2020.0002.

[8] M. Caleffi, A. S. Cacciapuoti, and G. Bianchi, "Quantum internet: from communication to distributed computing!," in *Proceedings of the 5th ACM International Conference on Nanoscale Computing and Communication*, in NANOCOM '18. New York, NY, USA: Association for Computing Machinery, Sep. 2018, pp. 1–4. doi: 10.1145/3233188.3233224.

[9] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, "Distributed quantum computing: A survey," *Comput. Netw.*, vol. 254, p. 110672, Dec. 2024, doi: 10.1016/j.comnet.2024.110672.

[10] Z. Li *et al.*, "Entanglement-assisted quantum networks: Mechanics, enabling technologies, challenges, and research directions," *IEEE Commun. Surv. Tutor.*, 2023.

[11] H. J. Kimble, "The quantum internet," *Nature*, vol. 453, no. 7198, pp. 1023–1030, Jun. 2008, doi: 10.1038/nature07127.

[12] H. J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, "Quantum repeaters: the role of imperfect local operations in quantum communication," *Phys. Rev. Lett.*, vol. 81, no. 26, pp. 5932–5935, Dec. 1998, doi: 10.1103/PhysRevLett.81.5932.

[13] J. Illiano, M. Caleffi, A. Manzalini, and A. S. Cacciapuoti, "Quantum Internet protocol stack: A comprehensive survey," *Comput. Netw.*, vol. 213, p. 109092, Aug. 2022, doi: 10.1016/j.comnet.2022.109092.

[14] S. Shi and C. Qian, "Concurrent entanglement routing for quantum networks: model and designs," in Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM) 2020. New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 62–75. doi: 10.1145/3387514.3405853.

[15] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018, doi: 10.22331/q-2018-08-06-79.

[16] P. Andrés-Martínez and C. Heunen, "Automated distribution of quantum circuits via hypergraph partitioning," *Phys. Rev. A*, vol. 100, no. 3, p. 032308, Sep. 2019, doi: 10.1103/PhysRevA.100.032308.

[17] R. G Sundaram, H. Gupta, and C. R. Ramakrishnan, "Efficient Distribution of Quantum Circuits," in *35th International Symposium on Distributed Computing (DISC 2021)*, S. Gilbert, Ed., in Leibniz International Proceedings in Informatics (LIPIcs), vol. 209. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021, p. 41:1-41:20. doi: 10.4230/LIPIcs.DISC.2021.41.

[18] R. Beals *et al.*, "Efficient distributed quantum computing," *Proc. R. Soc. Math. Phys. Eng. Sci.*, vol. 469, no. 2153, p. 20120686, May 2013, doi: 10.1098/rspa.2012.0686.

[19] M. Pant *et al.*, "Routing entanglement in the quantum internet," *Npj Quantum Inf.*, vol. 5, no. 1, pp. 1–9, Mar. 2019, doi: 10.1038/s41534-019-0139-x.

[20] Z. Yang *et al.*, "Layer-Wise Security Framework and Analysis for the Quantum Internet," *IEEE J. Sel. Areas Commun.*, Jan. 2025, doi: 10.48550/arXiv.2501.06989.

[21] C. H. Bennett, H. J. Bernstein, S. Popescu, and B. Schumacher, "Concentrating partial entanglement by local operations," *Phys. Rev. A*, vol. 53, no. 4, pp. 2046–2052, Apr. 1996, doi: 10.1103/PhysRevA.53.2046.

[22] P. C. Humphreys *et al.*, "Deterministic delivery of remote entanglement on a quantum network," *Nature*, vol. 558, no. 7709, pp. 268–273, Jun. 2018, doi: 10.1038/s41586-018-0200-5.

[23] C. Li, T. Li, Y. X. Liu, and P. Cappellaro, "Effective routing design for remote entanglement generation on quantum networks," *Npj Quantum Inf.*, vol. 7, no. 1, pp. 1–12, Jan. 2021, doi: 10.1038/s41534-020-00344-4.

[24] D. Sych and G. Leuchs, "A complete basis of generalized Bell states," *New J. Phys.*, vol. 11, no. 1, p. 013006, Jan. 2009, doi: 10.1088/1367-2630/11/1/013006.

[25] J.-W. Pan, D. Bouwmeester, H. Weinfurter, and A. Zeilinger, "Experimental entanglement swapping: entangling photons that never interacted," *Phys. Rev. Lett.*, vol. 80, no. 18, pp. 3891–3894, May 1998, doi: 10.1103/PhysRevLett.80.3891.

[26] W. Kozlowski *et al.*, "Architectural Principles for a Quantum Internet," Internet Engineering Task Force, Request for Comments RFC 9340, Mar. 2023. doi: 10.17487/RFC9340.

[27] R. Van Meter, K. Nemoto, W. J. Munro, and K. M. Itoh, "Distributed Arithmetic on a Quantum Multicomputer," *SIGARCH Comput Arch. News*, vol. 34, no. 2, pp. 354–365, May 2006, doi: 10.1145/1150019.1136517.

[28] Z. Yang, A. Ghubaish, R. Jain, H. Shapourian, and A. Shabani, "Asynchronous entanglement routing for the quantum internet," *AVS Quantum Sci.*, vol. 6, no. 1, p. 013801, Jan. 2024, doi: 10.1116/5.0172819.

[29] C. Tian *et al.*, "Asynchronous Routing for Multipartite Entanglement in Quantum Networks," in *2026 IEEE 16th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 2026, pp. 0533–0541. doi: 10.1109/CCWC67433.2026.11393739.

[30] R. Mengoni *et al.*, "Efficient Gate Reordering for Distributed Quantum Compiling in Data Centers," *EPJ Quantum Technol.*, Apr. 2026, doi: 10.1140/epjqt/s40507-026-00513-y.

[31] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comput. Netw. ISDN Syst.*, vol. 17, no. 1, pp. 1–14, Jun. 1989, doi: 10.1016/0169-7552(89)90019-6.

[32] G. Vardoyan and S. Wehner, "Quantum Network Utility Maximization," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Sep. 2023, pp. 1238–1248. doi: 10.1109/QCE57702.2023.00140.

[33] S. Pouryousef, H. Shapourian, A. Shabani, and D. Towsley, "Quantum Network Planning for Utility Maximization," in *Proceedings of the 1st Workshop on Quantum Networks and Distributed Quantum Computing*, in QuNet '23. New York, NY,



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



USA: Association for Computing Machinery, Sep. 2023, pp. 13–18. doi: 10.1145/3610251.3610553.

[34] D. Coppersmith, "An approximate Fourier transform useful in quantum factoring," Jan. 16, 2002, *arXiv*: arXiv:quant-ph/0201067. doi: 10.48550/arXiv.quant-ph/0201067.

[35] N. M. P. Neumann, R. van Houte, and T. Attema, "Imperfect Distributed Quantum Phase Estimation," in *Computational Science – ICCS 2020*, V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, Eds., Cham: Springer International Publishing, 2020, pp. 605–615. doi: 10.1007/978-3-030-50433-5_46.

[36] S. DiAdamo, M. Ghibaudi, and J. Cruise, "Distributed quantum computing and network control for accelerated vqe," *IEEE Trans. Quantum Eng.*, vol. 2, pp. 1–21, 2021.

[37] D. Cuomo *et al.*, "Optimized Compiler for Distributed Quantum Computing," *ACM Trans. Quantum Comput.*, vol. 4, no. 2, p. 15:1-15:29, Feb. 2023, doi: 10.1145/3579367.

[38] D. Ferrari, S. Carretta, and M. Amoretti, "A Modular Quantum Compilation Framework for Distributed Quantum Computing," *IEEE Trans. Quantum Eng.*, vol. 4, pp. 1–13, 2023, doi: 10.1109/TQE.2023.3303935.

[39] M. Bandini, D. Ferrari, S. Carretta, and M. Amoretti, "Optimized Compilation for Distributed Quantum Computing," arXiv.org. Accessed: May 24, 2026. [Online]. Available: https://arxiv.org/abs/2602.24062v1

[40] S. Yu, L. Zhang, E. Kissel, W. Wu, and I. Monga, "An extensible control plane software architecture for quantum networking research," in *Proceedings of the 2nd Workshop on Quantum Networks and Distributed Quantum Computing*, in QuNet '25. New York, NY, USA: Association for Computing Machinery, Sep. 2025, pp. 35–41. doi: 10.1145/3749096.3750033.

[41] P. Andres-Martinez *et al.*, "Distributing circuits over heterogeneous, modular quantum computing network architectures," *Quantum Sci. Technol.*, vol. 9, no. 4, p. 045021, Aug. 2024, doi: 10.1088/2058-9565/ad6734.

[42] J. Li *et al.*, "Fidelity-Guaranteed Entanglement Routing in Quantum Networks," *IEEE Trans. Commun.*, vol. 70, no. 10, pp. 6748–6763, Oct. 2022, doi: 10.1109/TCOMM.2022.3200115.

[43] L. Yang, Y. Zhao, H. Xu, L. Huang, and C. Qiao, "Dynamic Entanglement Routing Based on Stream Processing for Quantum Networks," *IEEEACM Trans. Netw.*, pp. 1–16, 2024, doi: 10.1109/TNET.2024.3485919.

[44] Z. Yang, A. Ghubaish, R. Jain, R. Kompella, and H. Shapourian, "Multi-Tree Quantum Routing in Realistic Topologies," *IEEE Commun. Mag.*, vol. 62, no. 10, pp. 78–85, Oct. 2024, doi: 10.1109/MCOM.006.2300851.

[45] R. Negrin, N. Dirnegger, W. Munizzi, J. Talukdar, and P. Narang, "Efficient Multiparty Entanglement Distribution with DODAG-X Protocol," Aug. 13, 2024, *arXiv*: arXiv:2408.07118. doi: 10.48550/arXiv.2408.07118.

[46] E. Schoute, L. Mancinska, T. Islam, I. Kerenidis, and S. Wehner, "Shortcuts to quantum network routing," Oct. 17, 2016, *arXiv*: arXiv:1610.05238. doi: 10.48550/arXiv.1610.05238.

[47] S. Bahrani, R. Wang, J. Parra-Ullauri, R. D. Oliveira, R. Nejabati, and D. Simeonidou, "Joint Network and Computing Resource Optimisation in Distributed Quantum Computing," in *2024 Optical Fiber Communications Conference and Exhibition (OFC)*, Mar. 2024, pp. 1–3.

[48] S. Bahrani, R. D. Oliveira, J. M. Parra-Ullauri, R. Wang, and D. Simeonidou, "Resource Management and Circuit Scheduling for Distributed Quantum Computing Interconnect Networks," Oct. 14, 2024, *arXiv*: arXiv:2409.12675. doi: 10.48550/arXiv.2409.12675.

[49] S. Pouryousef *et al.*, "Resource Management in Quantum Virtual Private Networks," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Sep. 2023, pp. 393–394. doi: 10.1109/QCE57702.2023.10298.

[50] S. Pouryousef, H. Shapourian, A. Shabani, R. Kompella, and D. Towsley, "Resource Placement for Rate and Fidelity Maximization in Quantum Networks," *IEEE Trans. Quantum Eng.*, vol. 5, pp. 1–16, 2024, doi: 10.1109/TQE.2024.3432390.

[51] H. Zhong *et al.*, "UNIQ: Communication-Efficient Distributed Quantum Computing via Unified Nonlinear Integer Programming," Nov. 29, 2025, *arXiv*: arXiv:2512.00401. doi: 10.48550/arXiv.2512.00401.

[52] L. Le, T. N. Nguyen, A. Lee, and B. Dumba, "Entanglement Routing For Quantum Networks: A Deep Reinforcement Learning Approach," in *ICC 2022 - IEEE International Conference on Communications*, May 2022, pp. 395–400. doi: 10.1109/ICC45855.2022.9839240.

[53] P. Sharma, S. Gupta, V. Bhatia, and S. Prakash, "Deep reinforcement learning-based routing and resource assignment in quantum key distribution-secured optical networks," *IET Quantum Commun.*, vol. 4, no. 3, pp. 136–145, 2023, doi: 10.1049/qtc2.12063.

[54] G. Ni, L. Ho, and H. Claussen, "Entanglement Request Scheduling in Quantum Networks Using Deep Q-Network," in *ICC 2025 - IEEE International Conference on Communications*, Jun. 2025, pp. 4603–4608. doi: 10.1109/ICC52391.2025.11161055.

[55] W. Luo, J. Zhao, T. Zhan, and Q. Guan, "Adaptive Job Scheduling in Quantum Clouds Using Reinforcement Learning," in *Proceedings of the 54th International Conference on Parallel Processing*, in ICPP '25. New York, NY, USA: Association for Computing Machinery, Dec. 2025, pp. 658–667. doi: 10.1145/3754598.3754641.

[56] J. I. Cirac, A. K. Ekert, S. F. Huelga, and C. Macchiavello, "Distributed quantum computation over noisy channels," *Phys. Rev. A*, vol. 59, no. 6, pp. 4249–4254, Jun. 1999, doi: 10.1103/PhysRevA.59.4249.

[57] A. Y. Kitaev, "Quantum computations: algorithms and error correction," *Russ. Math. Surv.*, vol. 52, no. 6, p. 1191, Dec. 1997, doi: 10.1070/RM1997v052n06ABEH002155.

[58] H. Buhrman and H. Röhrig, "Distributed Quantum Computing," in *Mathematical Foundations of Computer Science 2003*, B. Rovan and P. Vojtáš, Eds., Berlin, Heidelberg: Springer, 2003, pp. 1–20. doi: 10.1007/978-3-540-45138-9_1.

[59] K.-C. Chen, F. Burt, N. K. Panigrahy, and K. K. Leung, "Adaptive Resource Orchestration for Distributed Quantum Computing Systems", Accessed: Mar. 31, 2026. [Online]. Available: https://www.computer.org/csdl/magazine/ic/2026/01/11339455/2ddLDxzrchW

[60] N. K. Panigrahy *et al.*, "A Framework for Distributed Resource Allocation in Quantum Networks," Oct. 10, 2025, *arXiv*: arXiv:2510.09371. doi: 10.48550/arXiv.2510.09371.

[61] U. Bacher *et al.*, "Quantum resources in resource management systems," Nov. 14, 2025, *arXiv*: arXiv:2506.10052. doi: 10.48550/arXiv.2506.10052.

[62] D. Ferrari, A. S. Cacciapuoti, M. Amoretti, and M. Caleffi, "Compiler Design for Distributed Quantum Computing," *IEEE Trans. Quantum Eng.*, vol. 2, pp. 1–20, 2021, doi: 10.1109/TQE.2021.3053921.

[63] N. M. P. Neumann and R. S. Wezeman, "Distributed Quantum Machine Learning," vol. 1585, 2022, pp. 281–293. doi: 10.1007/978-3-031-06668-9_20.

[64] N. Dehaini, C. Chahoud, and M. Chehimi, "Deadline-Aware Scheduling of Distributed Quantum Circuits in Near-Term Quantum Cloud," Dec. 04, 2025, *arXiv*: arXiv:2512.06157. doi: 10.48550/arXiv.2512.06157.

[65] A. Singla, P. B. Godfrey, and A. Kolla, "High Throughput Data Center Topology Design," presented at the 11th USENIX



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173



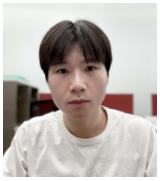
Symposium on Networked Systems Design and Implementation (NSDI 14), 2014, pp. 29–41.
[66] B. Lebiednik, A. Mangal, and N. Tiwari, "A Survey and Evaluation of Data Center Network Topologies," May 05, 2016, *arXiv*: arXiv:1605.01701. doi: 10.48550/arXiv.1605.01701.
[67] Y. Mao, Y. Liu, X. Shang, and Y. Yang, "Network Topology Design for Distributed Quantum Computing," in *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*, Jul. 2024, pp. 1213–1223. doi: 10.1109/ICDCS60910.2024.00115.
[68] R. Yu, R. Dutta, and J. Liu, "On Topology Design for the Quantum Internet," *IEEE Netw.*, vol. 36, no. 5, pp. 64–70, Sep. 2022, doi: 10.1109/MNET.001.2200170.
[69] T. R. Beauchamp, H. Jirovská, S. Gauthier, and S. Wehner, "A Modular Quantum Network Architecture for Integrating Network Scheduling with Local Program Execution," *IEEE Trans. Quantum Eng.*, pp. 1–31, 2025, doi: 10.1109/TQE.2025.3624658.
[70] C. Delle Donne *et al.*, "An operating system for executing applications on quantum network nodes," *Nature*, vol. 639, no. 8054, pp. 321–328, Mar. 2025, doi: 10.1038/s41586-025-08704-w.
[71] G. Vardoyan, S. Guha, P. Nain, and D. Towsley, "On the Stochastic Analysis of a Quantum Entanglement Switch," *SIGMETRICS Perform Eval Rev*, vol. 47, no. 2, pp. 27–29, Dec. 2019, doi: 10.1145/3374888.3374899.
[72] M. A. Hall, J. B. Altepeter, and P. Kumar, "Ultrafast Switching of Photonic Entanglement," *Phys. Rev. Lett.*, vol. 106, no. 5, p. 053901, Feb. 2011, doi: 10.1103/PhysRevLett.106.053901.
[73] C. Clos, "A Study of Non-Blocking Switching Networks," *Bell Syst. Tech. J.*, vol. 32, no. 2, pp. 406–424, 1953, doi: 10.1002/j.1538-7305.1953.tb01433.x.
[74] K. Bilal *et al.*, "Quantitative comparisons of the state-of-the-art data center architectures," *Concurr. Comput. Pract. Exp.*, vol. 25, no. 12, pp. 1771–1783, 2013, doi: 10.1002/cpe.2963.
[75] W. K. Wootters and W. H. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, Art. no. 5886, Oct. 1982, doi: 10.1038/299802a0.
[76] O. Milul *et al.*, "Superconducting Cavity Qubit with Tens of Milliseconds Single-Photon Coherence Time," *PRX Quantum*, vol. 4, no. 3, p. 030336, Sep. 2023, doi: 10.1103/PRXQuantum.4.030336.
[77] L. Lin *et al.*, "HWDSQP: A Historical Weighted and Dynamic Scheduling Quantum Protocol to Enhance Communication Reliability," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 8, pp. 2810–2824, Aug. 2025, doi: 10.1109/JSAC.2025.3568051.
[78] Z. Yang *et al.*, "Effective Scheduling for Quantum Data Centers," in *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Aug. 2025, pp. 454–455. doi: 10.1109/QCE65121.2025.10392.
[79] R. J. Hayek, J. Chung, and R. Kettimuthu, "A Review of Software for Designing and Operating Quantum Networks," *Adv. Quantum Technol.*, vol. 9, no. 2, p. e00808, 2026, doi: 10.1002/qute.202500808.
[80] D. Ferrari, M. Bandini, and M. Amoretti, "Execution Management of Distributed Quantum Computing Jobs," in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Sep. 2024, pp. 150–154. doi: 10.1109/QCE60285.2024.10269.
[81] T. R. Beauchamp, H. Jirovská, S. Gauthier, and S. Wehner, "A Modular Quantum Network Architecture for Integrating Network Scheduling with Local Program Execution," *IEEE Trans. Quantum Eng.*, pp. 1–31, 2025, doi: 10.1109/TQE.2025.3624658.
[82] G. Ni, D. Ferrari, L. Ho, and M. Amoretti, "Advanced Scheduling Strategies for Distributed Quantum Computing Jobs," arXiv.org. Accessed: May 24, 2026. [Online]. Available: https://arxiv.org/abs/2602.24152v2
[83] W. Dür, H.-J. Briegel, J. I. Cirac, and P. Zoller, "Quantum repeaters based on entanglement purification," *Phys. Rev. A*, vol. 59, no. 1, pp. 169–181, Jan. 1999, doi: 10.1103/PhysRevA.59.169.
[84] M. Cococcioni and L. Fiaschi, "The Big-M method with the numerical infinite M," *Optim. Lett.*, vol. 15, no. 7, pp. 2455–2468, Oct. 2021, doi: 10.1007/s11590-020-01644-6.
[85] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, fourth edition*. MIT Press, 2022.
[86] IBM, "OPL and MILP - IBM ILOG CPLEX Optimization Studio." Accessed: Aug. 29, 2025. [Online]. Available: https://www.ibm.com/docs/en/icos/22.1.1?topic=problem-opl-milp
[87] "NetworkX — NetworkX documentation." Accessed: Dec. 30, 2024. [Online]. Available: https://networkx.org/
[88] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Q. Appl. Math.*, vol. 27, no. 4, pp. 526–530, 1970, doi: 10.1090/qam/253822.
[89] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination," *East. Res. Lab. Digit. Equip. Corp. Hudson MA*, vol. 21, 1984.
[90] A. Li, S. Stein, S. Krishnamoorthy, and J. Ang, "QASMBench: A Low-Level Quantum Benchmark Suite for NISQ Evaluation and Simulation," *ACM Trans. Quantum Comput.*, vol. 4, no. 2, p. 10:1-10:26, Feb. 2023, doi: 10.1145/3550488.



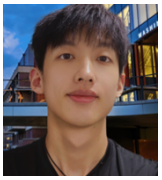
This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2026.3709173





**Zebo Yang** (Member, IEEE) received his M.E. in Computer Engineering from Waseda University, Japan, in 2019, and his Ph.D. in Computer Science from Washington University in St. Louis, USA, in 2025. He is currently an Assistant Professor in the Department of Electrical Engineering and Computer Science at Florida Atlantic University, USA. From 2011 to 2017, he held technical roles at Tencent, Baidu, and DJI. Between 2019 and 2025, he served as a Research Assistant at Washington University. His research interests include quantum computing, quantum networks, security, and machine learning.



**Chenliang Tian** (Student Member, IEEE) received his B.S. in Computing from Macao Polytechnic University, Macao, China, in 2018. He is currently a Ph.D. student in Computer Science at Washington University in St. Louis. Since 2025, he has been a Research Assistant at Washington University in St. Louis. His research interests include quantum computing and networking.



**Raj Jain** (Life Fellow, IEEE) received a B.E. in electrical engineering from APS University, India, and an M.E. from the Indian Institute of Science. He obtained his Ph.D. in Applied Maths (computer science) from Harvard University. He is the Barbara J. and Jerome R. Cox, Jr., Professor at Washington University in St. Louis. Dr. Jain co-founded Nayna Networks, Inc., and has held significant roles in academia and industry. He has received the 2018 James B. Eads Award and the 2017 ACM SIGCOMM Life-Time Achievement Award. He is known as one of the most cited authors in computer science and has authored "The Art of Computer Systems Performance Analysis." He is a Fellow of IEEE, ACM, and AAAS.



**Ramana Kompella** (Member, IEEE) received a B.Tech. from the Indian Institute of Technology, Bombay (1999), an M.S. from Stanford University (2001), and a Ph.D. from the University of California, San Diego (2007), all in computer science and engineering. He serves as a Distinguished Engineer and the Head of Research in Cisco's Emerging Tech and Incubation group, leading university research collaborations. His research focuses on data centers, cloud performance optimization, and router algorithmics. Dr. Kompella has been an ACM member since 2007, receiving the NSF CAREER Award in 2011, and has received Best Paper awards at conferences like ACM SOCC.



**Reza Nejabati** (Member, IEEE) is the Head of Quantum Research and Quantum Lab at Cisco, focused on advancing quantum networking. Previously, he was Chair Professor of Networks and Head of the High-Performance Network Group at the University of Bristol, UK. He is also a visiting professor and Cisco Chair at Curtin University's Cisco Center for Intent-Based Networking in Australia. Reza established globally recognized research in autonomous and quantum networks, co-founded Zeetta Networks Ltd, and received the IEEE Charles Kao Award in 2016 for contributions to 5G, smart cities, quantum communication, and future Internet experimentation.



**Mounir Hamdi** (Fellow, IEEE) received his B.S. in Electrical Engineering from the University of Louisiana in 1985, and his M.S. and Ph.D. from the University of Pittsburgh in 1987 and 1991, respectively. He is currently the Founding Dean of the College of Science and Engineering at Hamad Bin Khalifa University (HBKU). Previously, he was a Chair Professor and Founding Member of HKUST, where he also served as Head of the Department of Computer Science and Engineering. He was a Visiting Professor at Stanford University and EPFL (1999–2000). His research focuses on high-speed wired and wireless networking.



**Aiman Erbad** (Senior Member, IEEE) received a B.Sc. in computer engineering from the University of Washington (2004), a master's degree in embedded systems and robotics from the University of Essex (2005), and a Ph.D. in computer science from the University of British Columbia (2012). He is a Professor of Computer Engineering and the VP of Research and Graduate Studies at Qatar University. His research areas include cloud and edge computing, edge intelligence, network intelligence, IoT, and Blockchains. Dr. Erbad is also an editor and program chair for various journals and conferences and a Senior Member of ACM.



**Hassan Shapourian** (Member, IEEE) received an M.S. in electrical engineering from Princeton University (2013). He has a Ph.D. in Theoretical Physics from the University of Chicago (2019) and worked as a postdoctoral researcher at MIT/Harvard and Microsoft Station Q. He is a Senior Quantum Researcher at Cisco, leading projects on quantum information processing and hardware physics. He is a recipient of the Microsoft Research Postdoctoral Fellowship, the Simons Postdoctoral Fellowship, the Kavli Institute for Theoretical Physics (KITP) Graduate Fellowship, the Biruni Graduate Student Research Award, the John Bardeen Award (by UIUC), and the Princeton University Fellowship.



**Lei Yang** (Member, IEEE) is currently an Assistant Professor in the Department of Information Sciences and Technology at George Mason University. Lei's primary research interests lie in the joint area of Hardware/Software Co-Exploration for Neural Network Architectures, Embedded Systems, and High-Performance Computing, and currently works on quantum performance modeling. She has published more than 50 research articles in premier international conferences and journals, including DAC, CODES+ISSS, ASP-DAC, ICCD, HPCC, RTCSA, IEEE Transactions (TC, TPDS, TVLSI, TCAD), and ACM Transactions (TECS).



This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/