Entanglement Request Scheduling in Quantum Datacenter Networks

Francesco Vista, Stephen DiAdamo, Eneet Kaur, Hassan Shapourian, and Reza Nejabati

***Abstract*—In Distributed Quantum Computing (DQC), reliable entanglement distribution across a network of quantum computers is crucial. During the quantum program compilation process, the timing and locations of required entanglement resources can be determined. Leveraging this information, we use the properties of quantum network hardware—specifically, the entanglement generation and quantum switch operation rates—to optimize quantum job execution schedules. This work presents a system that, given a set of quantum programs, extracts an Entaglement Demand Schedule (EDS) and generates an execution plan, including resource allocation tables and switch control instructions. We outline the system architecture and benchmark its performance using collections of randomized quantum programs, and describe further steps to enhance our system.**

***Index Terms*—Quantum computing, distributed quantum computing, entanglement request scheduling, quantum networks, entanglement distribution.**

I. Introduction

Quantum computing addresses complex computational tasks by leveraging quantum-mechanical phenomena such as entanglement and superposition. However, quantum algorithms typically require a large number of quantum bits (qubits), and despite advancements, fabrication challenges and hardware noise limit current systems to sustaining only a few hundred qubits per machine. To overcome this limitation, Quantum Data Center (QDC) architecture has emerged as a promising solution for Distributed Quantum Computing (DQC) [1]–[3]. A QDC is a network of quantum computers, typically housed within a single location and interconnected via quantum channels. This setup significantly increases computational capacity by distributing workloads and sharing resources, enabling the execution of large-scale quantum algorithms through parallelization and scalability. For distributed quantum operations, entangled states must be shared between computing nodes [4]. In a QDC, this process involves generating entangled pairs of qubits, also known as Bell pairs, and distributing them across the network through quantum channels, which may include optical fibers, free-space communication links, or satellite-based systems [4], [5].

Although QDCs offer significant advantages, the complexity of coordinating the generation and distribution of entanglement poses challenges [6]. To manage the distribution of entanglements, networks need to extract an Entanglement Demand Schedule (EDS) based on distributed quantum algorithms [7], [8]. The EDS provides precise timing information for when entanglement is needed, determined at compile time, and takes into account the operational rates and constraints of the devices involved. Once extracted and by modeling the network, the EDS allows the network to generate control instructions for distributing entanglement, ensuring that resources such as quantum sources and switches are optimally configured to meet demand. Furthermore, since QDCs typically support multiple users, effective resource allocation algorithms are critical. In DQC, entanglement must be prepared ahead of time for non-local gates, and resource allocation impacts how much entanglement is needed. When demand exceeds network capacity, the timing of quantum operations may need adjustment, potentially introducing noise and errors, or deferring execution to a later time with sufficient resources.

We propose an algorithm that transforms monolithic quantum algorithms into resource allocations, execution schedules, and timing instructions for network components. It calculates total execution time for jobs on QDCs, using a star network topology with varying qubit counts. The model applies to both Noisy Intermediate-Scale Quantum (NISQ) systems and future logical qubits, operating above the physical layer with device success rates.

Our algorithm balances execution time, resource allocation, and fidelity, dynamically scheduling entanglement requests to maximize acceptance rates while minimizing noise, even under NISQ constraints. Its adaptability to diverse topologies and hardware ensures suitability for current and future quantum data centers, laying a foundation for scalable quantum computing and evolving with advances in hardware and network technologies.

The fulfillment of entanglement requests and scheduling in quantum networks has been studied in several works [9]–[12]. In [9], the authors propose a system for path selection and scheduling in multi-hop quantum networks, using entanglement rates and time slots specifying end nodes and entanglement arrival times. In contrast, our work extracts requests directly from quantum algorithms and, assuming a star network topology, eliminates complex routing by aligning scheduling with hardware parameters.

In [10], the scheduler input is the entanglement request arrival rate, where the consumers specify the desired rate at which the entanglement should be delivered. The scheduler then allocates resources to meet this fixed rate throughout the execution. However, our work addresses dynamic request intervals, with the aim of maximizing circuit fidelity by providing entanglement precisely when it is needed.

The study in [11] explores a multi-application quantum net-

Francesco Vista, Stephen DiAdamo, Eneet Kaur, Hassan Shapourian, and Reza Nejabati are with Cisco Research, San Jose, CA, USA. Francesco Vista is also with the Signal Processing and Communications (SIGCOM) Research Group at Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, 1855 Luxembourg City, Luxembourg.

work supporting Quantum Key Distribution (QKD) and DQC, addressing entanglement requests from source-destination pairs and using quantum memory for storage. It proposes scheduling and routing methods, evaluating request fulfillment. In contrast, our work focuses on DQC, automating entanglement request extraction from quantum circuits and manipulating them when needed. We benchmark fulfillment rates and execution runtimes across distributed networks.

In [12], the authors analyze entanglement distribution in a star network with a quantum switch, focusing on distribution rates using an analytical approach. However, their work does not address scheduling, relying on a max-weight scheduler that overlooks network-specific properties, unlike our approach.

Overall, this work is the first to address scheduling based on the operational rates of devices within the network to meet the demands of executing distributed quantum algorithms. We benchmark our scheduler and estimate the runtime of algorithms across various configurations of distributed quantum computers.

II. Quantum Data Center Computing

Near-term quantum computers require complex setups, including cooling systems and specialized enclosures for qubits. As a result, only a few entities can currently deploy quantum computers, similar to the early days of classical computing with mainframe systems. We are now entering an era where quantum computing clusters will be deployed for shared, multi-user access.

A quantum algorithm consists of machine-level instructions that manipulate qubits through operations known as gates. When these operations involve qubits located on different Quantum Processing Units (QPUs), non-local gates require entanglement distribution and classical communication to execute.

A quantum network connects multiple quantum computers via classical and quantum communication links. Executing a quantum algorithm in this distributed environment involves distributing qubits across the network and allocating resources. Transporting quantum information is challenging due to its fragile nature, making direct qubit transmission risky. Two protocols—Quantum Teleportation and Cat Entanglement/Disentanglement—address this challenge:

1) *Quantum Teleportation*: Transfers a qubit's state to another qubit in a different quantum computer using entangled pairs, classical communication, and local gate operations. The original state is destroyed after teleportation.
2) *Cat Entanglement/Disentanglement*: Uses entangled qubits to link qubits across quantum computers without moving the entire quantum state. The linked qubit serves as a control qubit for non-local gates, and a disentanglement operation follows, which destroys the link and sends phase information back.

Both protocols require pre-shared entanglement pairs. Teleportation physically moves qubit states, necessitating tracking, while cat-entanglement keeps qubits in their original positions, allowing for potentially multiple remote gate executions with a single linked copy.

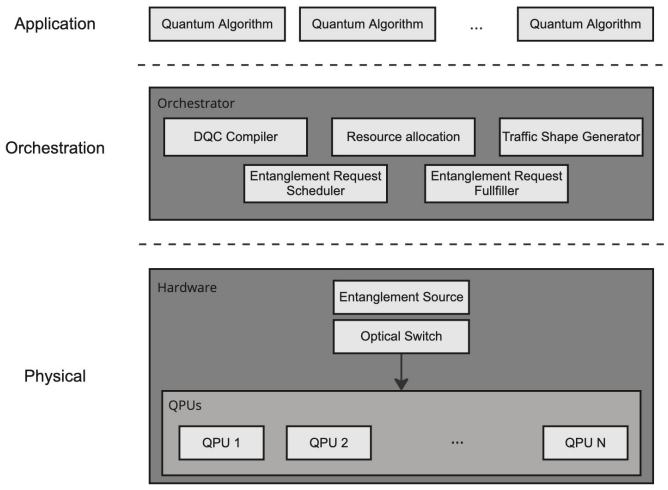

Fig. 1. A processing pipeline for executing multiple quantum algorithms over distributed quantum computers. Multiple quantum algorithms are sent to an orchestration layer, which involves several stages to prepare the algorithms for execution across a network. The orchestration layer generates instructions to operate the optical switch, enabling the distribution of entanglement to the appropriate locations for executing the distributed quantum algorithms.

Given this understanding, we design a network system that accommodates these requirements, ensuring that quantum algorithms are processed in a way that identifies when entanglement is needed during execution. The system would also assess whether the network hardware can provide the necessary entanglement and determine the optical switch configurations required to fulfill these demands. Fig. 1 illustrates a layered structure for a QDC designed to achieve this. We will explore these concepts in greater detail in the following subsections.

A. QDC Network Layers

In the context of network architectures that facilitate QDC, a comprehensive three-layer design has been developed from a holistic perspective. As illustrated in Fig. 1, this network architecture comprises three distinct logical layers.

Application layer: The highest layer is the QDC application layer, which consists of different quantum algorithms provided as monolithic circuit, that a generic user submits to the system for execution. It is important to note that the ability for a QDC network to accommodate multiple users simultaneously is contingent upon the availability of resources and the particular requirements of the users themselves.

Physical layer: The physical layer of the QDC network comprises a range of essential physical components necessary for DQC. It consists of a network of QPUs connected through optical fiber to a quantum source by a reconfigurable optical switch. The goal of this layer is to establish an entanglement link between QPUs in time to allow the execution of the users' circuit. In general, this can be a complex topology, but for the analysis in this work, we assume a star topology.

Orchestration layer: The orchestration layer manages the physical layer and acts as the interface between the application and the physical layers. Before executing a quantum algorithm, the orchestration layer performs two key functions: allocating

computing resources and compiling distributed quantum algorithms. These tasks are optimized together to improve network resources utilization. The resource allocation function assigns the algorithm to a cluster of QPUs based on the available qubits in each unit. A quantum compiler then translates the algorithm into a distributed quantum circuit, mapping the circuit's qubits to physical qubits and partitioning it into multiple segments. Additionally, the orchestration layer determines the timing of entanglement requests and evaluates how many requests the system can fulfill.

B. QDC Network Elements

From the above QDC network architecture, the corresponding network components, shown in the physical and orchestration layers in Fig. 1, are described as follows:

QPUs: Each quantum node is an independent QPU. These nodes are capable of initializing, storing, and manipulating quantum states. Each quantum node is also equipped with a finite number of quantum memory units and the necessary hardware to perform quantum operations, such as quantum gates. In a QPU, there are two types of physical qubits: data qubits, dedicated to executing quantum circuits, and communication qubits, employed for inter-QPU communication purposes.

Entanglement source: An entanglement source is an important device capable of generating entangled photons pairs. In this work, we build our system above the physical layer and focus on the scheduling and switching, under the assumption that the entanglement source can guarantee the emission of two entangled photons at a defined rate.

Optical Switch: A fast and low-loss optical switch, such as Micro-Electro-Mechanical System (MEMS), is a device able to switch between, or connect, multiple quantum nodes. It can be used to realize quantum networks of arbitrary topology at different distance scales. In this work, we consider a reconfigurable optical switch with 2 input interfaces and N output interfaces.

Centralized Controller: The centralized controller, or Orchestrator, manages the quantum cluster by receiving the monolithic circuit, generating its distributed version, and allocating resources within the QPU cluster. It creates the entanglement request schedule, controls entanglement pair generation and distribution, and distributes execution instructions to network devices.

III. Entanglement Scheduling and Distribution Framework

In this section, we present our entanglement scheduling and processing pipeline, illustrated in Fig. 2. This process happens in the centralized controller as a processing step to determine which jobs will take priority for the particular execution round. With regards to Fig. 1, this process happens in the orchestration layer. The process operates as follows:

Phase A: Incoming quantum algorithms, represented as monolithic circuits and denoted by the prefix J in Fig. 2, are initially stored in the quantum job buffer. This buffer serves as a holding area where jobs await resource allocation and scheduling.

Phase B: In this phase, the orchestration layer evaluates the current hardware status, including the availability of QPUs and qubits. Based on this assessment, it assigns QPUs to the algorithm. In this work, QPUs are selected based on availability to run the entire job, running instructions from a single job at a time, although one can consider multiple algorithms running on a single QPU. The monolithic circuits are then remapped to distributed circuits. It is worth noting that the controller should select the jobs that allows to maximize the use of the qubits per QPU used, while minimizing the entanglement requests needed. The jobs that have resources allocated and have been remapped are stored in the executing buffer.

Phase C: From each distributed circuit waiting in the executing buffer, the entanglement request extraction module extracts the specific timing of the entanglement request and the QPUs between which it must be established. Here, we refer to entanglement requests as the points in time in which entanglement is needed for the algorithm. Once the requests are extracted, they are then combined, sorted by time, and stored inside the request buffer.

Phase D: This phase evaluates entanglement requests against available hardware resources, such as entanglement sources and optical switches. This phase determines which requests can be fulfilled based on current resource constraints and capabilities. Requests are then sorted into two categories: *accepted* (those that the network can support) and *rejected* (those that cannot be fulfilled).

Phase E: Unfulfilled requests are deferred to the next scheduling iteration. If a request cannot be met at the exact time, the system explores alternative timings to maximize accepted requests while maintaining minimum fidelity. This may involve delaying subsequent requests or fulfilling the current request earlier. Each attempt results in acceptance or rejection, with retries limited to a predefined maximum. The approach balances noise from delays against overall job throughput.

Phases F and G: If some entanglement requests for a particular job remain unfulfilled after Phase E, that job is removed from the execution buffer. The system then selects another job from the quantum job buffer and repeats the process starting from Phase B. This iterative approach ensures that the system continually attempts to optimize resource usage and reduce the total execution time of all the quantum algorithms.

Phase H: Finally, the system generates the switch scheduling plan, detailing the sequence and timing of entanglement operations. Additionally, it provides an estimate of the total runtime required for the execution of the combined jobs, offering insights into the efficiency and performance of the scheduling framework.

This comprehensive framework ensures that quantum resources are efficiently allocated and managed, optimizing the execution of distributed quantum algorithms while addressing the inherent challenges of entanglement distribution. In the following subsections, we explain how we solve the various layers of this processing pipeline.

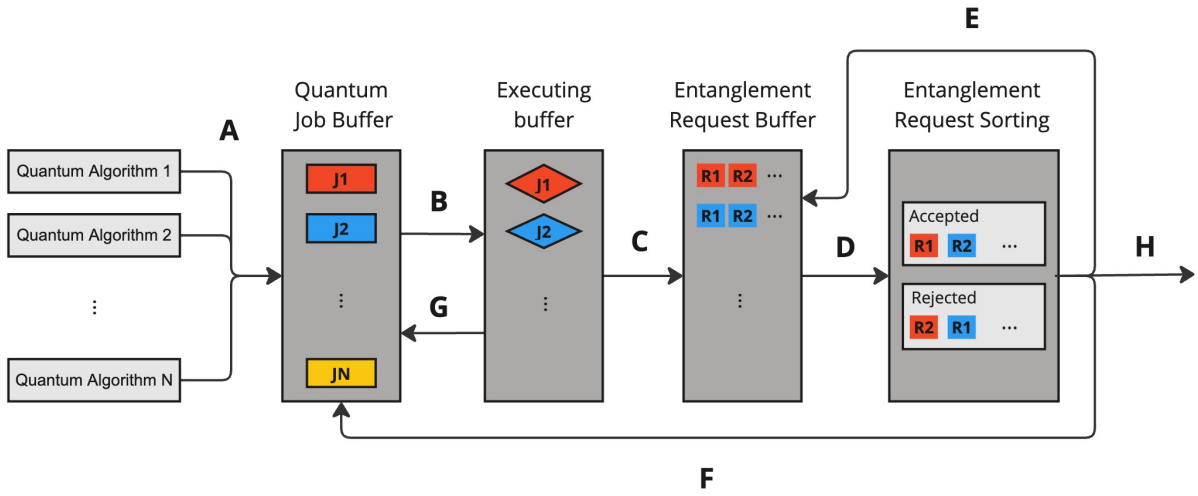

Fig. 2. Entanglement scheduling and distribution processing pipeline. The jobs begin as monolithic quantum algorithms, entering the queue in (rectangle) and get processed for execution and distributed (diamonds). The entanglement requests are extracted and sorted. A feedback loop attempts to optimize the accepted requests, or kicks a job out of the execution buffer if an agreeable number of requests cannot be met.

A. Request Extraction from Distributed Quantum Circuits

While existing literature often models end-to-end entanglement requests using stochastic processes [10], our approach introduces a deterministic, algorithm-driven method for generating entanglement requests directly from the submitted quantum algorithms. This ensures that the request generation is tailored to the specific computational requirements of the quantum algorithm, providing a more precise and predictable model.

The process begins after resources have been allocated to a quantum circuit, and the circuit has been remapped to accommodate distributed execution. During the remapping process, non-local gates, which require entanglement between qubits located on different QPUs, are identified. The timing of these gates directly informs the entanglement requests, as the execution of non-local gates is contingent on the availability of entanglement. Since quantum computers operate using precise, pulse-based operations, the execution times of these gates are predetermined by the system's timing characteristics. These timings are provided to the entanglement request extractor as input.

To formalize this process, we represent the remapped quantum circuit as a directed acyclic graph (DAG). In this representation, the nodes correspond to quantum gates, and the edges represent dependencies between gates. The DAG inherently identifies which operations can be executed simultaneously (i.e., within the same layer). By sweeping through these layers, we systematically analyze the gates to pinpoint any entanglement requirements introduced during the remapping process. When a non-local gate is encountered, its corresponding entanglement request is recorded, and the timing of the request is determined based on the execution time of the gate's layer, as dictated by the QPU gate times.

The key distinction of our model lies in its ability to generate entanglement requests directly from the input algorithms, in contrast to stochastic or probabilistic models that abstract away algorithm-specific details. This approach enables a fine-grained mapping between distributed quantum circuits and their entanglement requirements, ensuring that the generated schedule is both optimal and directly tied to the computational demands of the quantum algorithm. By leveraging the DAG representation, our method captures the interplay between circuit topology, gate timing, and resource allocation, offering a structured and deterministic mechanism for entanglement request generation.

B. Timing Aspects of the Network

To derive the switch scheduling plan, we need to consider the various timing aspects within the network architecture. In this work, we assume the entanglement source and switch are connected to the quantum hardware using a star topology, where an entanglement source generates Bell pairs at a rate f_{epr} Hz. What we consider here is an abstraction that deals only with successful events. These Bell pairs are transmitted to the appropriate QPUs via an optical switch, which requires a reconfiguration time t_{switch} to adjust its internal state. Our scheduler and control assume that the hardware can provide guaranteed entanglement at the given frequency. In the layer of the stack that we focus on here, we do not accommodate noise effects from the hardware.

To optimize the use of entanglement resources and fulfill as many requests as possible, we leverage the quantum memories within a computing node to receive Bell pairs in advance of when they are needed. To determine the maximum allowable storage time for these Bell pairs, we must consider the time-dependent decoherence of qubits in quantum memory, characterized by the memory relaxation time (T_1) and the dephasing time (T_2), as described in [13]. By accounting for the minimum fidelity required by the user, we can calculate the largest possible storage window t_{max} for Bell pairs while still meeting the fidelity criteria.

C. Request Scheduling and Fulfillment

The orchestration layer extracts the entanglement request schedule and directs the optical switch on when and where to forward entangled pairs. The scheduler adjusts requests to match job traffic patterns and QPU constraints while preserving their original order to maintain algorithm integrity. If a request fails, execution halts to avoid disrupting logic. The request fulfiller assesses feasibility based on quantum source and switch capabilities, providing feedback to the scheduler for optimizing switch operations.

For this process, we develop a greedy scheduling algorithm designed to efficiently allocate entanglement requests in a quantum network, ensuring that requests are fulfilled within their specified timing constraints. The details are as follows.

Initialization Phase: The algorithm begins by determining the entanglement generation events (eprEvents) within the execution time window based on the earliest and latest request times for a job and the entanglement source rate f_{epr}. Next, the switch is configured to match the destination nodes of the first request. Two lists, *accepted* and *rejected*, are initialized to store requests that are successfully scheduled and those that are not, respectively.

Processing Request Phase: The algorithm processes each request sequentially from the sorted list of entanglement requests. For each request, it checks if the specified QPUs match the current state of the switch. If they match, the algorithm invokes the *AcceptWithMatchSwitch* subroutine; if they do not match, it uses the *AcceptWithChangeSwitch* subroutine.

Acceptance Decision-Making: In the *AcceptWithMatchSwitch* subroutine, the algorithm determines a lower bound for the event time that satisfies the request. It then iterates through the list of eprEvents to find an event within the acceptable time window, that is, $t_{request} - t_{max} \leq t_{eprEvent} \leq t_{request}$. If such an event is found, the event time is assigned to the request, the event is removed from the eprEvents list, and the request is marked as accepted.

In the *AcceptWithChangeSwitch* subroutine, the algorithm first calculates the time t_{switch} required to switch the state of the optical switch. Then it checks if the request can be fulfilled after this switching time. A lower bound is determined based on the most recently accepted request and the timing constraints of the current request. The algorithm then iterates through the eprEvents to find a suitable event time that can accommodate the switch delay, i.e., $t_{request} - t_{max} - t_{switch} \leq t_{eprEvent} \leq t_{request} - t_{switch}$. If a suitable event is found, the request is updated with the new event time, the switch state is changed to match the required QPUs, and the request is marked as accepted.

Accepted requests are added to the *accepted* list, while rejected requests are added to the *rejected* list. This process continues until all requests have been processed. Finally, the algorithm returns the lists of accepted and rejected requests. The Greedy Scheduling algorithm ensures that entanglement requests are handled in their original order, respecting interleaved timing requirements. By incorporating the hardware constraints and capabilities of the quantum source and switch, the algorithm returns the scheduling plan in terms of trigger events for the optical switch.

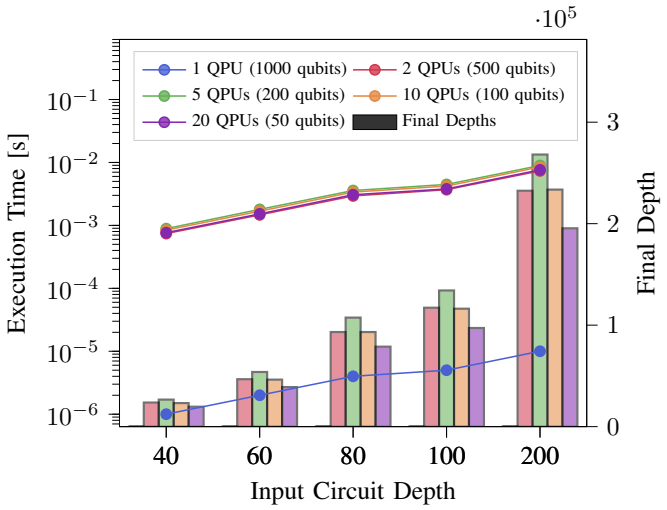

Fig. 3. The runtime estimate distributed quantum algorithms on the left axis (line plots) and the final circuit depth on the right axis (bar plots). The color of the lines and the color of the bars match to represent the associated configuration.

Centralized Controller and Network Telemetry: For effective scheduling, the centralized controller requires detailed network state and hardware information. Standard telemetry systems monitor metrics such as device connectivity, available quantum resources (e.g., qubit states), entanglement fidelity, and switch configurations. Network devices periodically update a centralized telemetry database, providing the controller with an up-to-date view of the topology, device capabilities, and ongoing entanglement processes.

Role of the *AcceptWithChangeSwitch* Subroutine: Telemetry data predicts the feasibility of fulfilling requests based on current and future network states. The *AcceptWithChangeSwitch* subroutine evaluates proposed entanglement configurations, simulating switch reconfigurations, and accounting for delays. It ensures sufficient resources are available, determines if a request can be accepted or adjusted, and respects interleaved timing. This subroutine, combined with the controller's detailed network knowledge, optimizes resource allocation and maximizes fulfilled requests.

IV. Numerical Results and Discussion

A series of simulations were performed to assess the feasibility of the proposed entanglement scheduling and distribution processing pipeline. The first set of simulations investigate the impact of various job parameters and network setting, such as the depth of the single monolithic circuit and the number of QPUs, on the execution time and the number of accepted requests. The second part evaluates the benefits of running multiple jobs simultaneously, comparing a parallel approach, where jobs are executed concurrently, to a sequential approach, where jobs are processed one at a time, with a particular focus on reducing total execution time.

To define our quantum data center network, we considered the following setup unless otherwise specified: (i) 1, 2, 5, 10,

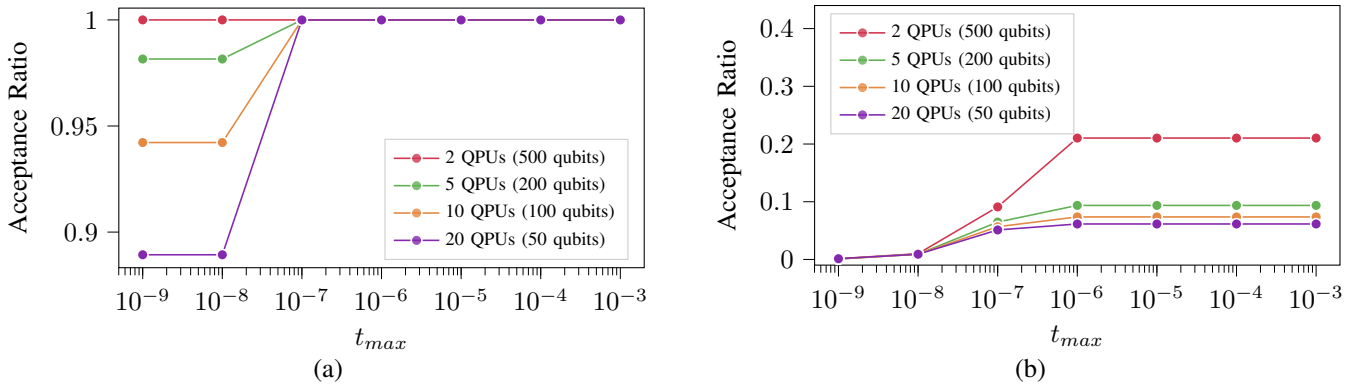

Fig. 4. Results for an entanglement source with (a) $f_{epr} = 1$ GHz and (b) $f_{epr} = 1$ MHz. Both figures show the ratio of accepted entanglement requests for various topologies as a function of the memory storage time t_{max}.

20 QPUs, each with 1000, 500, 200, 100, and 50 processing qubits, respectively, and an infinite number of communication qubits with finite coherence time; (ii) a continuous entanglement source producing successful entangled pairs at a rate f_{epr} of 1 GHz or 1 MHz; and (iii) a reconfigurable 2xM optical switch.

For generating entanglement requests, we set the durations of single-qubit and two-qubit gates to 5 ns and 50 ns, respectively, based on a superconducting qubit architecture [1]. The switch reconfiguration time t_{switch} is 2 ns for a single output line change and 10 ns for a double output line change [14]. For the quantum memory model, we consider the decoherence times T_1 and T_2 to be 0.5 ms [15]. Furthermore, to ensure a minimum fidelity of 0.8, and based on the values for T_1 and T_2, the maximum storage time t_{max} is calculated to be 10^{-3} s, as detailed in [13].

A. Scenario 1: Single Job Input

In this scenario, we consider a single random quantum job, i.e., $J = 1$, requiring 1000 qubits, executed across different QPU configurations.

Fig. 3 shows the execution time and final depth after converting the circuit to a distributed format, varying the depth of the input quantum job. As expected, both metrics increase with the original circuit depth due to the growing number of gates. Notably, single QPUs outperform data center scenarios, as additional gates are introduced when non-local gates are performed in a distributed setup. However, increasing the number of QPUs reduces both the depth and execution time, as more gates can be executed in parallel.

Fig. 4 shows the ratio of accepted requests to the total number of requests under different storage time windows and source frequency f_{epr}, with a circuit depth of 200. Specifically, in Fig. 4 (a) with a $f_{epr} = 1$ GHz, it can be observed that at lower storage times, a gap in the acceptance ratio emerges with increasing the number of QPUs. This occurs because the number of requests increases with more QPUs, necessitating the storage of a larger number of entangled pairs. However, as time storage increases, i.e., starting from $t_{max} = 10^{-7}$, the acceptance ratio improves, and performance between different QPU configurations becomes nearly identical.

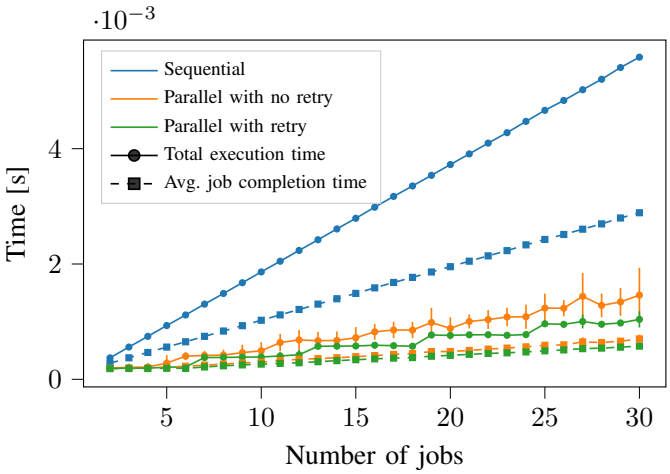

Fig. 5. A comparison of runtime estimates and job completion time for executing multiple jobs over cluster of quantum computers using a sequential approach vs. parallelized with a greedy entanglement scheduler with and without scheduling retries.

To further corroborate these results, we evaluated the number of communication qubits needed to execute the job. With low-quality memories, i.e., storage time below 10^{-5} s, the number of communication qubits remains low across all QPU configurations. As storage time increases, the number of stored qubits rises significantly, peaking at around 8000 qubits for 20 QPUs and 2000 qubits for 2 QPUs. This indicates that better memory quality allows more qubits to be stored in advance, improving acceptance rates.

To compare how emission frequency affects the schedule, we slow the entanglement source to $f_{epr} = 1$ MHz and show the same results in Fig. 4 (b). As observed, a slower entanglement source significantly impacts the fulfillment of requests, leading to minimal improvements across configurations, with a maximum acceptance rate of 0.21% for 2 QPUs starting around $t_{max} = 10^{-7}$. With slower emission, fewer qubits can be sent ahead of time, leading to a buffer of 1 communication qubit.

B. Scenario 2: Multiple Jobs Input

To better understand the benefits of executing multiple jobs in parallel, we compute the total execution time for running J jobs, where J ranges from 2 to 30. Each job requires 10 qubits with a depth of 500. The underlying architecture consists of 30 QPUs, each with 5 qubits of memory, and a storage window of 10^{-3} seconds. The results are obtained through Monte Carlo simulation, with 100 iterations performed for each number of jobs.

In the sequential approach, jobs are executed one after the other. In contrast, the parallel approach proposed in this work allows jobs to be executed concurrently. In the retry approach, if some requests cannot be fulfilled, the first pending request and subsequent requests are delayed by 10^{-9} s until an available entanglement pair is found, or a maximum of 10 attempts are reached for the same initial request.

Fig. 5 shows total execution time and average job completion time for varying numbers of jobs. The sequential approach exhibits a linear increase in both metrics as jobs are executed one by one. In contrast, the parallel and parallel-with-retry approaches achieve significantly lower execution times by enabling concurrent job processing. The retry mechanism further reduces waiting times by resolving pending requests, demonstrating the efficiency of parallel execution in minimizing delays.

V. Conclusion and Outlook

This paper has presented a robust framework for scheduling and distributing entanglement requests across QPUs, considering quantum hardware constraints. Our simulation campaign demonstrated the feasibility and effectiveness of the proposed approach in optimizing both execution time and acceptance ratios under varying conditions.

Future enhancements could include incorporating redundant requests to mitigate noise and loss, increasing the success rate of entanglement generation. Furthermore, techniques like entanglement distillation could be integrated into the framework to enhance the quality and fidelity of the entangled states, ensuring more reliable quantum operations.

In addition, the development and implementation of more advanced scheduling algorithms, such as dynamic programming, reinforcement learning, or optimization-based methods, will be investigated. These approaches could enable more informed decisions when selecting the optimal set of jobs to execute concurrently, balancing the trade-offs between throughput, noise, and overall execution time.

In summary, the proposed entanglement scheduling framework effectively addresses the challenges of quantum resource management, achieving high acceptance ratios and efficient execution times. By considering hardware constraints and introducing more advanced job selection strategies, this framework lays a strong foundation for future improvements in quantum computing systems.

References

[1] S. Bahrani, R. Wang, R. Oliveira, R. Nejabati, and D. Simeonidou, "Analysing the effect of quantum network interconnect on the performance of distributed quantum computing," in *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, 2023, pp. 1–3.

[2] M. Caleffi, M. Amoretti, D. Ferrari, J. Illiano, A. Manzalini, and A. S. Cacciapuoti, "Distributed quantum computing: a survey," *Computer Networks*, p. 110672, 2024.

[3] J. Liu and L. Jiang, "Quantum data center: Perspectives," *IEEE Network*, vol. 38, no. 5, pp. 160–166, 2024.

[4] Z. Li, K. Xue, J. Li, L. Chen, R. Li, Z. Wang, N. Yu, D. S. L. Wei, Q. Sun, and J. Lu, "Entanglement-assisted quantum networks: Mechanics, enabling technologies, challenges, and research directions," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2133–2189, 2023.

[5] A. Chang, Y. Wan, G. Xue, and A. Sen, "Entanglement distribution in satellite-based dynamic quantum networks," *IEEE Network*, vol. 38, no. 1, pp. 79–86, 2024.

[6] L. Chen, K. Xue, J. Li, Z. Li, R. Li, N. Yu, Q. Sun, and J. Lu, "Redp: Reliable entanglement distribution protocol design for large-scale quantum networks," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 7, pp. 1723–1737, 2024.

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

[8] R. Parekh, A. Ricciardi, A. Darwish, and S. DiAdamo, "Quantum algorithms and simulation for parallel and distributed quantum computing," in *2021 IEEE/ACM Second International Workshop on Quantum Computing Software (QCS)*. IEEE, 2021, pp. 9–19.

[9] C. Cicconetti, M. Conti, and A. Passarella, "Request scheduling in quantum networks," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 2–17, 2021.

[10] S. Gauthier, G. Vardoyan, and S. Wehner, "An architecture for control of entanglement generation switches in quantum networks," *IEEE Transactions on Quantum Engineering*, vol. 4, no. 01, pp. 1–17, 2023.

[11] H. Gu, R. Yu, Z. Li, X. Wang, and F. Zhou, "Esdi: Entanglement scheduling and distribution in the quantum internet," in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2023, pp. 1–10.

[12] N. K. Panigrahy, T. Vasantam, D. Towsley, and L. Tassiulas, "On the capacity region of a quantum switch with entanglement purification," in *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE, 2023, pp. 1–10.

[13] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. de Oliveira Filho, M. Papendrecht, J. Rabbie, F. Rozpędek, M. Skrzypczyk *et al.*, "Netsquid, a network simulator for quantum information using discrete events," *Communications Physics*, vol. 4, no. 1, p. 164, 2021.

[14] X. Xue and N. Calabretta, "Nanosecond optical switching and control system for data center networks," *Nature communications*, vol. 13, no. 1, p. 2257, 2022.

[15] S. Ganjam, Y. Wang, Y. Lu, A. Banerjee, C. U. Lei, L. Krayzman, K. Kisslinger, C. Zhou, R. Li, Y. Jia *et al.*, "Surpassing millisecond coherence in on chip superconducting quantum memories by optimizing materials and circuit design," *Nature Communications*, vol. 15, no. 1, p. 3687, 2024.

Francesco Vista received the Ph.D. in electrical and information engineering from Politecnico di Bari, Bari, Italy, in January 2024. He spent a year as a quantum researcher intern at Cisco Quantum Lab. Currently, he is a research and development specialist in quantum communication at the University of Luxembourg.

Stephen DiAdamo received a Hon. B.Sc. in computer science at the University of Toronto in 2014 and a M.Sc. in mathematics in 2018 and Dr.-Ing in electrical engineering from the Technical University of Munich in 2023. He was a quantum research scientist at Cisco Quantum Lab and now works as a technical founder at Qoro Quantum.

Eneet Kaur received her Ph.D. in 2020 at Louisiana State University with a focus on quantum information and communication. She then spent a year and a half at the Institute for Quantum Computing, delving into quantum repeater designs, followed by a year at the College of Optical Sciences, University of Arizona. She is currently a quantum researcher in the Quantum Research group at Cisco.

Hassan Shapourian received a 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.

Reza Nejabati is the Head of Quantum Research and Quantum Lab at Cisco, focused on advancing quantum networking. Before joining Cisco, he served as a Chair Professor of Networks and Head of the High-Performance Network Group in the Department of Electrical and Electronic Engineering at the University of Bristol, UK. His research focus and expertise lies in the realms of Quantum and Classical network infrastructure for the Future Internet.