Designing an Emulation Platform for Realistic Experimentation on Quantum Networks

Blanca Lopez [1,2,*], **Ivan Vidal** [2], **Francisco Valera** [2], **Diego R. Lopez** [3]
[1]IMDEA Networks Institute, Avda. del Mar Mediterráneo, 22, 28918 Leganés, Madrid, Spain
[2]Telematic Engineering Department, Universidad Carlos III de Madrid, Avda. Universidad, 30, 28911 Leganés, Madrid, Spain
[3]Telefónica, Ronda de la Comunicación, Fuencarral-El Pardo, 28050 Madrid, Spain
[*]Correspondence: blanca.lopez@imdea.org (B.L.)

ABSTRACT

The realization of a Quantum Internet is expected to transform communication networks by enabling intrinsically secure information exchange and distributed quantum applications. However, the experimental study of large-scale quantum networks remains limited by the high cost and operational constraints of quantum hardware. To address this challenge, this paper presents the design of a platform conceived to support research and experimentation in the field of quantum networking. The proposed system provides an emulation environment that mirrors the behavior of a real quantum network, enabling researchers to test protocols and mechanisms through a set of quantum operation instructions accessible via an Application Programming Interface (API). Emulated quantum nodes can be deployed across physically distributed locations, using physical or virtual machines or virtualization containers, allowing experiments to reproduce realistic network conditions and inter-node interactions. This approach enables applications to be executed over a persistent, configurable, and interactive environment that integrates both quantum and classical communication layers. To validate the feasibility of the proposed design, a prototype has been built and used to demonstrate its operation through a multi-hop quantum teleportation proof of concept. The results confirm the viability and flexibility of the architecture, laying the foundation for a distributed framework that facilitates the development and evaluation of quantum network architectures, control mechanisms, and experimental use cases.

1 INTRODUCTION

The emergence of a Quantum Internet is expected to mark a new technological revolution, substantially transforming our current communication networks and endowing them with unprecedented capabilities, such as the application of teleportation mechanisms or the intrinsic security guaranteed by quantum cryptography techniques [1]. However, despite these promising expectations, the realization of a fully functional Quantum Internet integrated with current networking infrastructures remains a distant goal.

At present, most of the quantum technologies are at best in an intermediate stage of development. This consequently means that they remain largely inaccessible to the broader research community, due to

their high cost and the strict requirements of quantum hardware operating conditions. This limitation becomes particularly critical in the context of quantum networking research, where experimental validation often demands large-scale testbeds. Networks composed of only a few quantum devices are insufficient to explore realistic scenarios, considering essential aspects such as routing, resource management, and inter-domain coordination. Addressing these challenges requires environments replicating networks of dozens, or even hundreds, of interconnected devices, thereby approximating the complexity and dynamics of an actual quantum network deployment at scale. Furthermore, these environments must be flexible enough to support new mechanisms and protocols under research, as well as new applications of the Quantum Internet, in order to verify their viability and functionality under realistic conditions.

To address these limitations and facilitate investigation in the field of quantum networking without the need to deploy physical quantum hardware, several simulation frameworks have been developed (e.g., [2–6]). These platforms provide valuable tools for modeling quantum systems, testing communication protocols, and analyzing theoretical performance under controlled conditions. However, they operate as centralized simulations executed on a single machine, which imposes intrinsic constraints. A simulation is by design a predefined and static process, one that must be configured and launched in advance. Once initiated, its behavior cannot be modified on the fly to reflect changing conditions or user actions. This rigidity limits experimentation with adaptive scenarios. In addition, these platforms often exhibit limited interoperability with external systems, making it difficult to integrate widely used classical network management or orchestration tools as control layers, as they operate outside the simulator closed software environment and cannot interact with the simulations. Consequently, while these tools are indispensable for early-stage experimentation, they offer only a partial representation of the operational and architectural challenges inherent to realistic, large-scale quantum networks.

In this context, we introduce our platform, a distributed and interactive environment currently under continuous active development, designed not just to simulate, but to mimic the behavior of quantum communication networks under realistic operational conditions. The design of the platform establishes the principles and architecture required for the creation of a fully distributed emulation environment that faithfully reproduces the temporal and structural dynamics of a quantum network. Unlike conventional simulations that follow a fixed sequence of predefined events and terminate upon completion, emulation provides a persistent, continuously running environment that remains active after deployment. This environment is built using classical distributed equipment, such as physical and virtual machines or virtualization containers [7]. By enabling continuous and interactive experimentation, the emulation approach brings us closer to the concept of a Quantum Network Digital Twin [8]. Users can interact with the devices that form the emulated quantum network in real time, configuring topologies, adding or removing nodes, triggering entanglement operations, or running control protocols dynamically, thereby enabling adaptive experimentation beyond static, predetermined scenarios.

The platform presented in this paper has been designed with interoperability as a central guiding principle. It can integrate classical networks and orchestration tools, such as SDN controllers or monitoring frameworks, as external managing layers, thereby bridging the gap between quantum and classical network management paradigms and enabling research on different Quantum Internet architectures. The prototype introduced in this work establishes the foundation for a flexible, modular, and extensible research platform aimed at testing, validating, and analyzing quantum network protocols and architectures under realistic and interactive conditions.

This paper continues as follows: Section 2 reviews the state of the art in quantum network modeling; Section 3 discusses the design of the platform, while Section 4 present the first prototype; Section 5 proposes and presents a proof of concept; and lastly, Section 6 gathers the conclusions of this works and identifies future work on this line.

2 STATE OF THE ART

Since the potential of quantum objects in the field of communications was first theorized, a growing number of platforms have been developed to model, simulate, and analyze quantum communication networks. Each of these platforms focuses on different aspects of network design, protocol evaluation, and hardware abstraction. Among the best known are NetSquid [2], SeQUeNCe [3], QuISP [4], QuNetSim [5], and SimulaQron [6]. These tools have significantly contributed to advancing quantum networking research by enabling the study of entanglement distribution, quantum repeaters, and Quantum Key Distribution (QKD) protocols under diverse experimental and theoretical conditions. Most of them rely on discrete-event simulation engines that reproduce the stochastic behavior of quantum and classical processes, allowing researchers to test network topologies, timing constraints, and routing strategies in controlled, scenario-based experiments.

NetSquid [2] is one of the most detailed and physically grounded open-access simulation environments. It provides high-fidelity modeling of quantum states, decoherence processes, photon loss, and hardware imperfections, making it especially suitable for the design and evaluation of physical-layer components and quantum repeater architectures. Similarly, SeQUeNCe [3] also follows a discrete-event scheme and emphasizes scalability and modularity. It includes both quantum and classical message layers, enabling end-to-end simulation of network protocols over large topologies with configurable timing, noise, and resource management policies.

In contrast, QuISP [4] focuses on the development and testing of quantum repeater networks at a systems level. It provides a modular and extensible architecture for studying entanglement generation, purification, and swapping strategies in complex topologies. It enables the analysis of throughput and latency trade-offs across multiple repeater chains.

At a higher level of abstraction, SimulaQron [6] and QuNetSim [5] provide accessible environments for

network-layer protocol development and testing. SimulaQron offers a virtual quantum network that allows multiple applications to communicate through simulated quantum channels, using socket-based connections. QuNetSim, on the other hand, extends this approach by providing a more structured network stack, incorporating distinct layers for quantum and classical communication.

Despite their different goals and contributions, all these platforms share a common trait. They operate as predefined simulations that must be configured before execution and proceed according to a fixed event schedule. While some of them, such as SimulaQron or QuNetSim, allow limited runtime interaction through external Application Programming Interfaces (APIs), the simulated network itself remains bounded by the simulation timeline and terminates once all scheduled events have been processed. Furthermore, their closed execution models limit interoperability with external software systems, preventing integration with standard network management or orchestration tools, widely used in real-world deployments, or even with application processes willing to use the quantum network for whatever purpose. None of these frameworks supports a persistent, continuously running network environment that can be modified or managed in real time using external control layers.

In contrast, our platform adopts an emulation-oriented approach inspired by the design principles of Quditto [9], a framework focused on the emulation of QKD networks. Quditto has demonstrated the potential of creating emulated QKD network environments capable of interacting with external application processes or management planes. Building upon this vision, our platform extends the concept toward general-purpose quantum network emulation. Instead of reproducing events within a single process, it deploys multiple, independently running nodes that communicate through message exchange, mirroring the structure of a real distributed network. This design allows continuous operation, dynamic reconfiguration, and real-time control interaction, thus enabling experimentation with adaptive and hybrid quantum-classical networking scenarios that fall beyond the scope of existing simulators.

3 GENERAL DESIGN OF THE PLATFORM

Building upon the limitations identified in existing simulation frameworks, the proposed platform introduces a distributed architecture designed to reproduce the operational dynamics of a quantum communication network. Rather than executing a predefined sequence of events on a single simulation engine, our platform builds an environment composed of emulated, independent, continuously running elements that interact through a message-based communication layer. This approach allows real-time experimentation, adaptive control, and integration with classical orchestration and management tools.

Beyond bridging the gap between simulation and operational deployment, the design of the platform pursues additional objectives. First, it aims to deliver realistic results in terms of modeling the quantum behavior of the network, providing a reliable approximation of entanglement dynamics and quantum operations. Second, it seeks to be a widely accessible platform, creating a user-friendly environment that

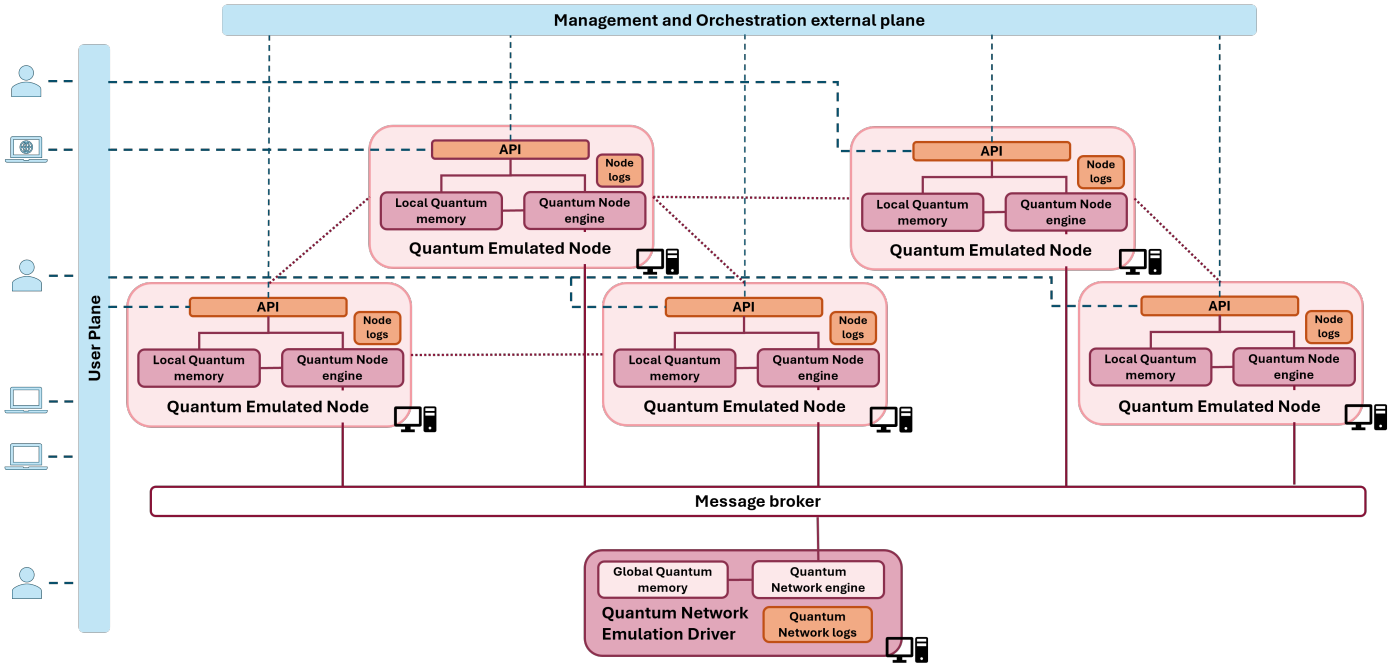

Figure 1: Architecture of the proposed platform, showing distributed Quantum Emulated Nodes connected through a message broker and coordinated by a Quantum Network Emulation Driver, with external orchestration and user interaction enabled via APIs.

enables any interested party, regardless of their programming knowledge, to conduct experiments on it. Finally, the platform design emphasizes flexibility and extensibility, ensuring that its architecture can evolve alongside emerging quantum technologies and networking paradigms without requiring fundamental redesign.

Figure 1 illustrates the overall system architecture. The figure depicts two different types of elements within the platform, the Quantum Emulated Nodes and the Quantum Network Emulation Driver, both interconnected through a message broker that coordinates communication and synchronization across the system. The diagram also presents potential external users and a management and orchestration plane, which can interact with individual nodes via their exposed APIs to request operations, configure, monitor, or dynamically modify the network.

Following this general review of the system architecture, we now examine in more detail the two main classes of components that constitute the platform. On the one side, the Quantum Emulated Nodes represent the distributed elements of the network, each operating autonomously and capable of performing local quantum operations. On the other side, the Quantum Network Emulation Driver acts as a coordination layer that manages shared and global resources, ensures coherence among distributed quantum states, and enables higher-level control of network behavior.

3.1 Quantum Emulated Nodes

The Quantum Emulated Nodes are the fundamental building blocks of the emulated network, as are the nodes in a real network. Each node operates as a self-contained process that can be deployed on a phys-

ical machine, virtual machine, or containerized environment. This flexibility allows the emulator to scale across distributed infrastructures, reproducing realistic network conditions such as communication delays, node heterogeneity, and asynchronous event handling.

As can be seen in Figure 1, each Quantum Emulated Node comprises three main components:

- The **Local Quantum Memory** is responsible for storing the quantum states associated with the node, including local qubits and those entangled with remote nodes. It supports state creation, manipulation, and retrieval, providing the foundation for emulating qubit generation and quantum gate operations.
- The **Quantum Node Engine** executes local quantum operations, such as local qubit creations, gate application, or measurements. It manages the internal logic of the node and interacts with the local quantum memory and with both external nodes and the Quantum Network Emulation Driver through the message-based communication layer.
- The **API** exposes a set of endpoints for external access. Through this interface, users or external controllers can interact with each node to query state information, inject control commands, or trigger specific quantum operations in real time. The API also enables the remote execution of external programs or applications over the emulated network, allowing higher-level software processes, such as protocol testing or application-layer services, to operate directly on top of the emulation environment.

Additionally, each node maintains a Node Log registry that records all local events, control messages, and measurement outcomes. These logs facilitate debugging, performance evaluation, and analysis. The distributed execution of nodes, coupled with asynchronous communication via the message broker, enables dynamic coordination and realistic representation of network-scale quantum behavior.

3.2 Quantum Network Emulation Driver

The Quantum Network Emulation Driver serves as the coordination layer that manages global resources and ensures coherence across the distributed nodes. While each node operates autonomously, the driver maintains a system-wide view of the network, orchestrating the creation, tracking, and routing of entangled states among multiple endpoints.

This module is also composed of three main subsystems:

- The **Global Quantum Memory** maintains a record of all quantum states, with a special focus on those that form entangled distributed collections. It ensures consistency and synchronization when quantum operations are executed.

- The **Quantum Network Engine** is responsible for coordinating high-level network functionalities such as entanglement creation and routing, purification, and resource allocation. To this end, it interacts with the global quantum memory and with all the nodes in the network through the message broker. A quantum communications modeling tool is used to derive realistic performance parameters, which are then applied to the operations carried out by users on the quantum memories and links.
- The **Quantum Network Logs** aggregate monitoring information and performance metrics from all nodes. These logs provide a unified view of network activity and could be exported to external visualization and analysis tools.

3.3 Additional components

In addition to the main functional elements, the platform includes a Message Broker as an auxiliary component that ensures efficient interconnection. It provides asynchronous and scalable message exchange among nodes and between nodes and the Emulation Driver, decoupling message transmission from reception. This allows nodes to operate independently in time, so even a recently instantiated node can retrieve previously published messages relevant to its operation. It abstracts internal message transport, enabling messages to be exchanged using standardized protocols such as MQTT [10] or AMQP [11], which are lightweight publish-subscribe and message-queuing protocols widely used for communication in distributed systems. This abstraction allows for flexible deployment configurations, including local setups for small-scale experiments and distributed clusters for larger topologies. The message broker also supports message queuing, topic-based routing, and event synchronization, which are essential for maintaining consistency in the emulated quantum network state.

Beyond internal communication, the API offered by each emulated node allows direct interaction from external entities, such as SDN controllers (e.g., the one defined in the ETSI GS QKD 015 standard [12] for QKD networks), experiment managers, or visualization tools. Through these APIs, orchestration systems can perform operations including network configuration, entanglement initiation, or performance monitoring. This design enables the platform to operate as a hybrid experimentation environment, where classical control systems can dynamically manage quantum processes, mirroring the architecture expected in real-world quantum Internet deployments. The inclusion of these integration interfaces also facilitates interoperability with external frameworks for data collection and analysis, enhancing observability and experiment traceability.

4 PROTOTYPE IMPLEMENTATION

A prototype of the proposed platform has been built to validate the design and assess the feasibility of the emulation approach. This prototype, developed to verify the core architectural concepts introduced in Section 3, focuses on demonstrating an end-to-end workflow, from topology definition to entanglement

generation, qubit manipulation, and state persistence; using a single coordination entity that mediates all network operations. The overall scheme of the built prototype is shown in Figure 2, which shows how multiple Quantum Emulated Nodes communicate with a single Quantum Network Emulation Driver. This implementation provides a foundation for distributed extensions while already supporting realistic quantum-state modeling, temporal evolution, and flexible network configuration.

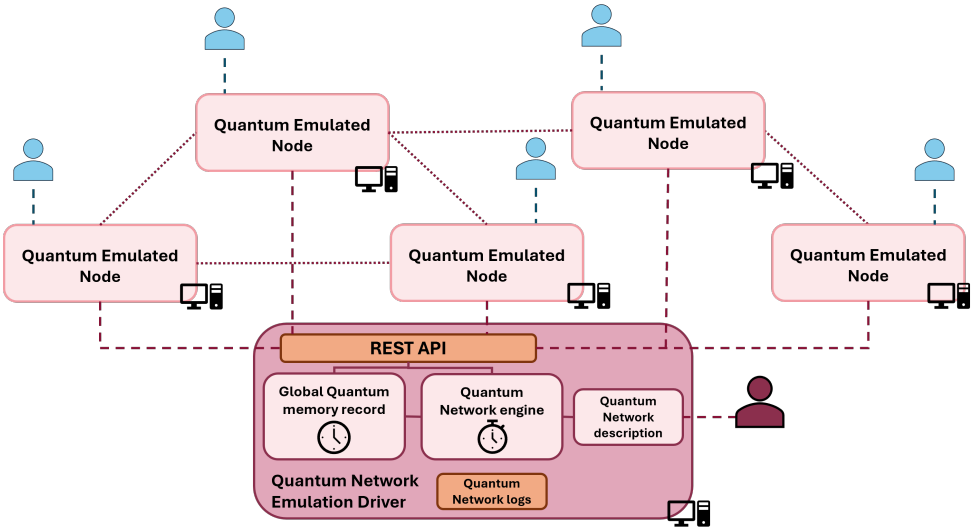

Figure 2: Prototype of the platform. Multiple Quantum Emulated Nodes interact with a central Quantum Network Emulation Driver via a specific API. The driver maintains the global quantum memory record, manages and coordinates quantum operations, and models temporal evolution and decoherence to reproduce realistic quantum network behavior.

4.1 Validation prototype

The built prototype adopts a centralized architecture, in which the Quantum Network Emulation Driver coordinates all network activities and maintains a global view of the network state. The Driver is responsible for creating, managing, and updating quantum states, as well as handling requests from the Quantum Emulated Nodes.

Each Quantum Emulated Node operates as an independent process that interacts with the driver through a REST-based API. Nodes use the corresponding REST requests to perform specific operations, such as creating a local qubit, generating entanglement with another node, performing quantum measurements, or applying gate operations on their qubits. Upon receiving a request, the Quantum Network Emulation Driver executes the corresponding operation and returns relevant results to the requesting node, including updated quantum states in the form of reduced density matrices, measurement outcomes, or positions in the emulated nodes quantum memory where the qubits are stored.

4 Network configuration

The network topology can be fully defined by the user in a JSON configuration file. This file specifies the set of nodes, their neighbors, and the properties of the emulated quantum and classical links connecting them. This structure allows complete flexibility: researchers can emulate arbitrary topologies by editing

the JSON file without modifying the source code. Each link can include parameters such as length, attenuation, depolarization rate, and fidelity of the entangled pair source; which enables realistic customization of physical-layer conditions.

4 Quantum modeling

To accurately represent quantum behavior, the Quantum Network Engine integrates the NetSquid framework as its modeling backend. NetSquid provides a detailed simulation of quantum state behavior, including entanglement creation and quantum memory modeling. The platform modular integration allows flexibility: the underlying quantum modeling component can be replaced or extended with other frameworks in future versions without altering the emulator higher-level logic. The Quantum Network Emulation Driver also implements native ancillary functions to address, for example, the decoherence of quantum states saved on the emulated memories over time.

Operations currently supported include: (i) local qubit creation, representing initialization of a quantum memory position; (ii) entanglement generation between any two connected nodes, with state fidelity and photon loss modeled according to the channel parameters; (iii) entanglement swapping, allowing indirect entanglement between distant nodes via intermediate ones; (iv) quantum gate application, e.g. X, Y, Z, H, or CNOT, on one or multiple qubits; (v) qubit measurement in the X, Y, or Z basis, with automatic update of all correlated states; (vi) qubit state retrieval, providing the reduced density matrix of one or more qubits, including those entangled across nodes.

All these operations modify a persistent representation of the global quantum state, allowing the emulator to reproduce the logical behavior of a quantum communication network.

4 API and orchestration

The Quantum Network Emulation Engine exposes a REST API, allowing users or external controllers and orchestration systems to interact with the platform to configure the network, trigger operations, or collect results, either through direct requests or programmatically. The interface provides access to the main functionalities of the platform, network initialization, qubit manipulation, entanglement management, and measurement.

All API calls are synchronous, i.e., operations are executed to completion before a response is returned, and results are provided as JSON objects, enabling straightforward integration with external orchestration systems, visualization dashboards, and automated experiment pipelines.

4 Temporal behavior and decoherence modeling

The Quantum Network Emulation Driver applies a time-aware operation model, as emphasized with the clock elements in Figure 2, which is crucial to faithfully emulate quantum behavior. The driver keeps track of the real elapsed time between operations and adjusts the quantum states accordingly. This implies that fidelity and entanglement quality degrade naturally with time. In practice, this means that if an

operation is delayed, either by the user interaction or processing time, the stored states will lose fidelity in proportion to the elapsed interval.

In addition, completing quantum operations, such as entanglement generation or swapping, introduces a deliberate temporal delay, accounting for physical propagation or operation time. With these two mechanisms, the platform achieves a time-consistent emulation in which both decoherence and operation durations reflect the behavior of practical quantum networks.

4 State persistence

The platform maintains a persistent record of the entire quantum network state, including the nodes local quantum memories and all shared entangled states. Each operation requested by the nodes and performed by the Network Emulation Engine updates the Global Memory record file, enabling experiments to be paused, resumed, or partially restored at any time. This mechanism also stores timestamps for every quantum state update, which are essential for applying time-dependent decoherence in subsequent operations.

Functions in the engine allow saving and restoring complete or partial networks from the record, and inspecting fidelity and density matrices of stored quantum states. This approach ensures consistency across all operations and provides traceability for analysis.

5 VALIDATION OF DESIGN

To test and validate the capabilities of our design, we designed a multi-hop teleportation experiment deployed across different virtual machines hosted at the NEXTONIC laboratory [13]. The experiment involved entanglement generation and exchange, local qubit creation, and quantum gate execution. The objective is to teleport a quantum state between two non-adjacent nodes within the emulated topology by dynamically establishing long-distance entanglement through intermediate relays, thus verifying that the system correctly models quantum operations and state propagation across a non-trivial topology.

The network topology used for this proof of concept consists of six nodes ($N_0, ..., N_5$) interconnected through a set of emulated quantum and classical channels, as illustrated in Figure 3. Each quantum channel includes an entanglement source located at its midpoint, responsible for generating entangled photon pairs that are distributed to the connected nodes. The network structure, described entirely through a JSON configuration file (see Figure 4 for a snippet), supports multiple possible paths between N_0, the sender in our experiment, and N_5, the receiver.

The network and its links are characterized by physical parameters that impact its performance and temporal behavior. In our case, we set channel attenuation and depolarization rate to 0.15 dB/km and 0.5, respectively. Additionally, we utilized sources with a fidelity of 0.98 and a frequency of 1 MHz[1].

[1] Not all link parameters are shown in Figure 4 due to size constraints.

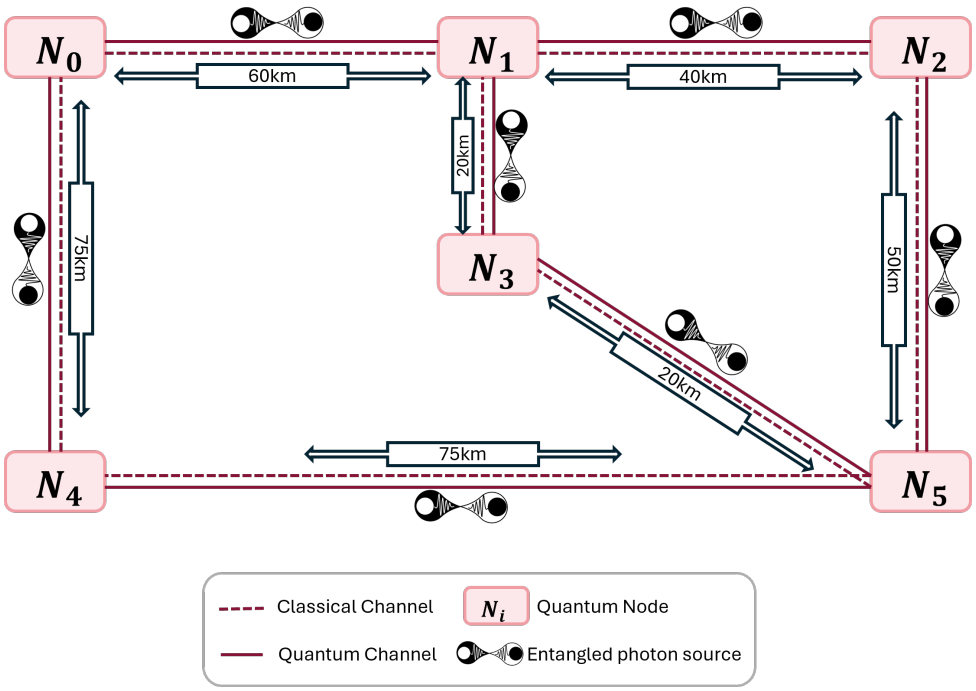

Figure 3: Network topology used for the teleportation proof of concept. The six-node network includes multiple quantum and classical channels, enabling different teleportation routes between N_0 and N_5.

```
{
  "nodes": {
    "N0": {"neighbors": ["N1", "N4"]},
    "N1": {"neighbors": ["N0", "N2", "N3"]},
    "N2": {"neighbors": ["N1", "N5"]},
    "N3": {"neighbors": ["N1", "N5"]},
    "N4": {"neighbors": ["N0", "N5"]},
    "N5": {"neighbors": ["N2", "N3", "N4"]}
  },
  "channels": [
    {"type": "quantum", "endpoints": ["N0", "N1"], "length_km": 60, "fidelity": 0.98, "
        source_frequency_MHz": 1, "depolar_rate": 0.5, "attenuation_db_per_km": 0.15},
    {"type": "quantum", "endpoints": ["N0", "N4"], "length_km": 75, (...)},
    {"type": "quantum", "endpoints": ["N1", "N2"], "length_km": 40, (...)},

                              (...)

    {"type": "classical", "endpoints": ["N0", "N1"], "length_km": 60}
    {"type": "classical", "endpoints": ["N0", "N4"], "length_km": 75}
    {"type": "classical", "endpoints": ["N1", "N2"], "length_km": 40},
                              (...)
  ]
}
```

Figure 4: Snippet from the JSON network configuration file for the proof of concept.

5.1 Teleportation procedure

The teleportation workflow follows the standard quantum teleportation protocol executed across a multi-hop quantum path.

1. **Entanglement distribution:** Bell pairs are generated between adjacent nodes along the chosen route. Each entanglement attempt is emulated using the link attenuation and depolarization parameters along with its length, which directly affect the fidelity of the achieved shared state.
2. **Entanglement swapping:** Intermediate nodes perform Bell-state measurements to connect local entanglements into a single end-to-end pair. After each swap, the Quantum Network emulation driver must update the global quantum memory and recalculate the combined density matrix of the resulting long-distance state.
3. **Local qubit preparation and measurement:** The sender, in our case N_0, prepares a qubit to teleport its state, ψ, and applies a CNOT gate between this qubit and its half of the entangled end-to-end pair, followed by a Hadamard transformation. Both qubits are then measured, producing two classical bits, b_1 and b_2
4. **Classical communication and correction:** The classical bits are communicated over classical channels to the receiver, N_5, which applies Pauli corrections to its qubits, completing the teleportation process.
5. **State verification:** In our experiment, the final state at N_5 is retrieved, and the probability $P_\psi = \psi\rho_{out}\psi$ is calculated as an indicator of teleportation success and fidelity.

As we discussed in previous sections, our platform is time-aware. Before performing any operation, the driver computes the real elapsed time since each state last update and applies decoherence accordingly using amplitude damping and dephasing parameters. Additionally, after each operation, the driver waits for the simulated duration corresponding to photon propagation and local processing before returning the results. In the proof-of-concept experiment, the quantum memory parameters were configured to provide relatively long coherence times, allowing entangled states to persist during the entire sequence of operations without employing purification or error-correction protocols. These parameters can, however, be freely adjusted by the user to explore different physical regimes and evaluate the impact of memory quality and decoherence on network performance.

5.2 Routes and Results

Three distinct routes were evaluated in the teleportation proof-of-concept experiment. Each path differs either in total length or in the number of intermediate nodes, providing a diverse set of conditions for entanglement distribution, swapping, and state fidelity evolution. Table 1 lists for each path, the nodes

involved, the total distance, the number of entanglement-swapping processes required, the final probability of the teleported state being measured in the correct state $|\psi\rangle$ (P_ψ), and the total wall-clock time elapsed during the entire teleportation process.

Path	Distance (km)	Swaps	Final P_ψ	Total time (s)
N_0–N_1–N_2–N_5	150	2	0.734	26.11
N_0–N_4–N_5	150	1	0.736	31.36
N_0–N_1–N_3–N_5	100	2	0.793	24.15

Table 1: Summary of the teleportation experiments showing the nodes involved, total distance, number of swaps, final P_ψ, and total wall-clock time.

Figure 5 illustrates the temporal progression of the experiment, showing the complete event sequence from network initialization to final-state retrieval. Each point corresponds to a recorded event (network setup, entanglement creation, swapping, teleportation operations, and final-state measurement).

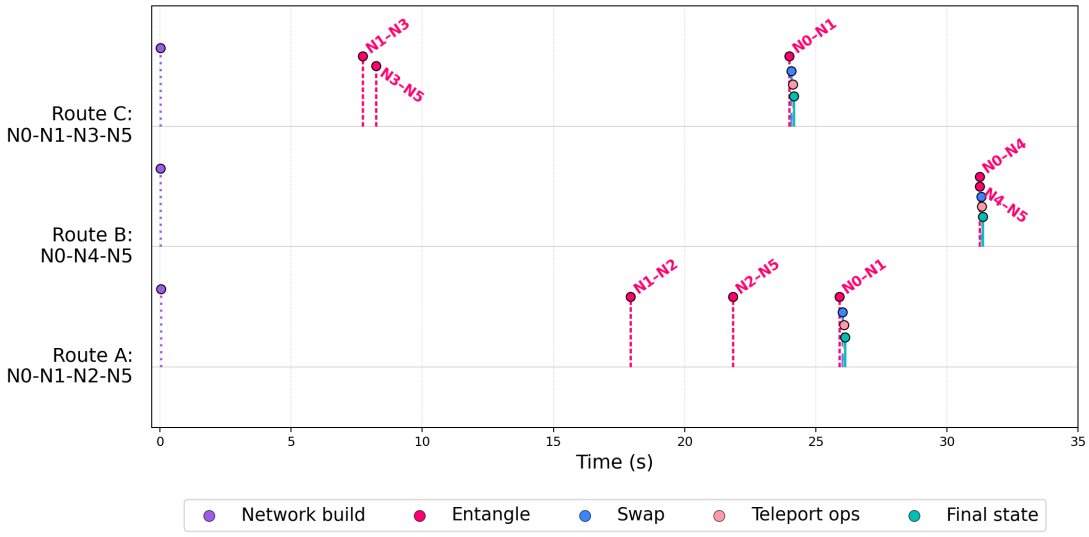

Figure 5: Temporal evolution of the teleportation experiments across the different routes. Each color indicates a different operation type.

Intermediate fidelities collected after each entanglement generation and swapping step are summarized below. These values, together with the computed end-to-end fidelities, confirm the expected physical behavior of the network. In the present configuration, each link-level entangled pair is generated over a channel that introduces attenuation and depolarization, causing its fidelity to decrease with the physical length of the link.

An additional, route-dependent effect arises from the quantum memory model. When entangled photons arrive at different times, the earlier pair must wait in memory until the remaining links finish sharing their pairs. During this waiting period, the stored state decoheres. This effect is negligible when link lengths (and therefore arrival times) are similar, but becomes significant when the path contains both short and long segments.

- **Route A (N_0–N_1–N_2–N_5)**: $F_{N_0-N_1} = 0.806$, $F_{N_1-N_2} = 0.864$, $F_{N_2-N_5} = 0.834$; after two swapping operations at N_1 and N_2, the end-to-end fidelity was $F_{N_0-N_5} = 0.575$. The teleported state yielded a final $P_\psi = 0.7336$.
- **Route B (N_0–N_4–N_5)**: $F_{N_0-N_4} = 0.765$, $F_{N_4-N_5} = 0.765$; following one swap at N_4, the resulting fidelity was $F_{N_0-N_5} = 0.603$, corresponding to $P_\psi = 0.7359$.
- **Route C (N_0–N_1–N_3–N_5)**: $F_{N_0-N_1} = 0.806$, $F_{N_1-N_3} = 0.929$, $F_{N_3-N_5} = 0.929$; after two swaps at N_1 and N_3, the end-to-end fidelity was $F_{N_0-N_5} = 0.617$, with a final $P_\psi = 0.7929$.

Overall, Route C provides the best performance, consistent with its shorter optical distance of 100 km. Routes A and B, which both span 150 km, show comparable teleportation accuracy despite differing designs. Route B slightly outperforms Route A because its two long links have matched lengths, causing the Bell pairs to be generated nearly simultaneously. This minimizes memory dwell time and prevents additional decoherence before the swap. In contrast, Route A includes links of unequal lengths, meaning some entangled photons ($N1 - N2$ and $N2 - N5$) must wait in memory while others are still being generated ($N0 - N1$), leading to extra decoherence that degrades the final swapped state.

Longer and asymmetric routes lead to increased decoherence and reduced teleportation quality, while shorter paths and balanced link lengths maintain stronger correlations. These findings demonstrate that the results obtained are coherent with the expected physical behavior and confirm the flexibility of the proposed design, which enables the integration of quantum communication modeling within a distributed emulation framework.

6 LESSONS LEARNED AND FUTURE DIRECTIONS

This paper presented the design of a platform conceived to bridge the gap between simulation-based quantum network models and realistic, interactive emulation environments. The proposed architecture enables the deployment of persistent, distributed, and configurable quantum network environments that can be interacted with in real time, thereby reproducing the operational behavior of an actual quantum communication system.

To demonstrate the feasibility of the proposed design, a prototype has been developed. This prototype validates the practicality of the architectural principles by enabling time-aware, configurable emulation of quantum operations such as entanglement generation, swapping, and qubit manipulation. A teleportation experiment across multiple routes was conducted as a proof of concept.

Considerations about platform implementation

Building on the results obtained with the prototype and the proof of concept, future development will focus on evolving the system into a complete distributed emulation framework, following the design prin-

ciples introduced in this paper. This evolution will involve introducing a clear separation of responsibilities between local and global operations within the system.

On the one hand, local operations such as local qubit creation, quantum gate execution, or measurements, will be performed directly by each Quantum Emulated Node. In this configuration, the Quantum Network Emulation Driver will act primarily as a coordinator, updating the global quantum memory to maintain overall coherence across the network. When an operation involves a qubit that is part of an entangled pair, any modification applied at one node must be reflected both in the corresponding density matrix stored in the global memory and in the reduced density matrix of its entangled partner, ensuring physical consistency across distributed states. This mechanism will enable each node to operate with autonomy while preserving a globally synchronized view of the quantum state.

On the other hand, global operations such as entanglement distribution and routing, will remain under the supervision of the Quantum Network Emulation Driver. These tasks require coordination among multiple nodes, the management of shared quantum resources, and the consistent propagation of quantum and classical control messages. The driver will thus serve as a high-level orchestrator, ensuring that distributed processes remain synchronized, conflicts among concurrent operations are avoided, and the resulting quantum states remain valid across all participating network elements. To this aim, an asynchronous communication layer incorporating the message broker features described in Section 3 will replace the current requestresponse model between the nodes and the Driver.

ACKNOWLEDGMENT

This work has been partially funded by the project 6GINSPIRE PID2022-137329OB-C42, funded by MCIN/AEI/10.13039/501100011033/. This publication is part of the I+D+I project MADQuantum-CM, financed by the European Union NextGeneration-EU, Madrid Government and by the PRTR. This work has also been partially supported by the European Union's Quantum Flagship project Quantum Security Networks Partnership (QSNP), under grant 101114043.

REFERENCES

[1] A. S. Cacciapuoti, M. Caleffi, F. Tafuri, F. S. Cataliotti, S. Gherardini and G. Bianchi, "Quantum Internet: Networking Challenges in Distributed Quantum Computing," in IEEE Network, vol. 34, no. 1, pp. 137-143, January/February 2020, doi: 10.1109/MNET.001.1900092.

[2] Coopmans, T.; Knegjens, R.; Dahlberg, A.; Maier, D.; Nijsten, L.; de Oliveira Filho, J.; Papendrecht, M.; Rabbie, J.; Rozpedek, F.; Skrzypczyk, M.; et al. NetSquid, a discrete-event simulation platform for quantum networks. *Commun. Phys.* **2021**, *4*, 164. [arXiv:quant-ph/2010.12535]. https://doi.org/10.1038/s42005-021-00647-8.

[3] Wu, X.; Chung M., J.F.; Kolar, A.; Wang, E.; Zhong, T.; Kettimuthu, R.; Suchara, M. Simulations of Photonic Quantum Networks for Performance Analysis and Experiment Design. In Proceedings of the

2019 IEEE/ACM Workshop on Photonics-Optics Technology Oriented Networking, Information and Computing Systems (PHOTONICS), Denver, CO, USA, 18 November 2019; pp. 28–35. https://doi.org/10.1109/PHOTONICS49561.2019.00010.

[4] Satoh, Ryosuke, et al. QuISP: A Quantum Internet Simulation Package. 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), IEEE, 2022, pp. 35364. Crossref, https://doi.org/10.1109/qce53715.2022.00056.

[5] Diadamo, S.; Nötzel, J.; Zanger, B.; Bee, M.M. QuNetSim: A Software Framework for Quantum Networks. *IEEE Trans. Quantum Eng.* **2021**, *2*, 1–12. https://doi.org/10.1109/TQE.2021.3092395.

[6] Dahlberg, A.; Wehner, S. SimulaQron—A simulator for developing quantum internet software. *Quantum Sci. Technol.* **2018**, *4*, 015001. https://doi.org/10.1088/2058-9565/aad56e.

[7] Morabito, R.; Kjällman, J.; Komu, M. Hypervisors vs. Lightweight Virtualization: A Performance Comparison. In Proceedings of the 2015 IEEE International Conference on Cloud Engineering, Tempe, AZ, USA, 9–13 March 2015; pp. 386–393. https://doi.org/10.1109/IC2E.2015.74.

[8] Diego Lopez, Vicente Martin, Blanca Lopez, and Luis Miguel Contreras, *A Multiplane Architecture Proposal for the Quantum Internet*. Internet Engineering Task Force, 2023. Work in Progress. Available online: https://datatracker.ietf.org/doc/draft-lopez-qirg-qi-multiplane-arch/

[9] Lopez, B.; Diaz-Bricio, A.; Perez, J.; Vidal, I.; Valera, F. Quditto: Emulating and Orchestrating Distributed QKD Network Deployments. arXiv quant-ph, 2025 https://doi.org/10.48550/arXiv.2512.15408

[10] MQTT Version 5.0. Edited by Andrew Banks, Ed Briggs, Ken Borgendale, and Rahul Gupta. 07 March 2019. OASIS Standard. https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html. Latest version: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[11] OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0. 29 October 2012. OASIS Standard. http://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-complete-v1.0-os.pdf

[12] ETSI, *Quantum Key Distribution (QKD); Control Interface for Software Defined Networks*; ETSI GS QKD 015 V2.1.1 (2022-04); ETSI: Sophia Antipolis, France, 2022.

[13] NEXTONIC, an Open Research and Innovation Laboratory Focusing on Next Generation Mobile Networks. Available online: https://www.nextonic.org/.