Aliro™

Quantum Computer Networking:

A Vision for the Future Quantum Data Center

Aliro

www.alirotech.com

Quantum Computer Networking: A Vision for the Future Quantum Data Center

Introduction

What will it take for quantum computing to move from today's noisy, small-scale devices to fault-tolerant, scalable, and reliable computation?

"The Network *is* the Computer" – Coined in 1984 by John Gage, a key proponent of distributed computing.

A growing consensus is emerging: practical quantum advantage will require networked Quantum Processing Units (QPUs). It is unlikely that one monolithic single-device architecture can scale to the very large numbers of physical qubits, often projected as millions of qubits depending on assumptions for practical quantum computation. Instead, the path forward points to modular quantum systems with multiple QPUs interconnected, sharing quantum states, coordinating distributed operations, and executing workloads as a single logical machine. This modular architecture is analogous to how society scaled classical HPC with CPU and GPU clusters.

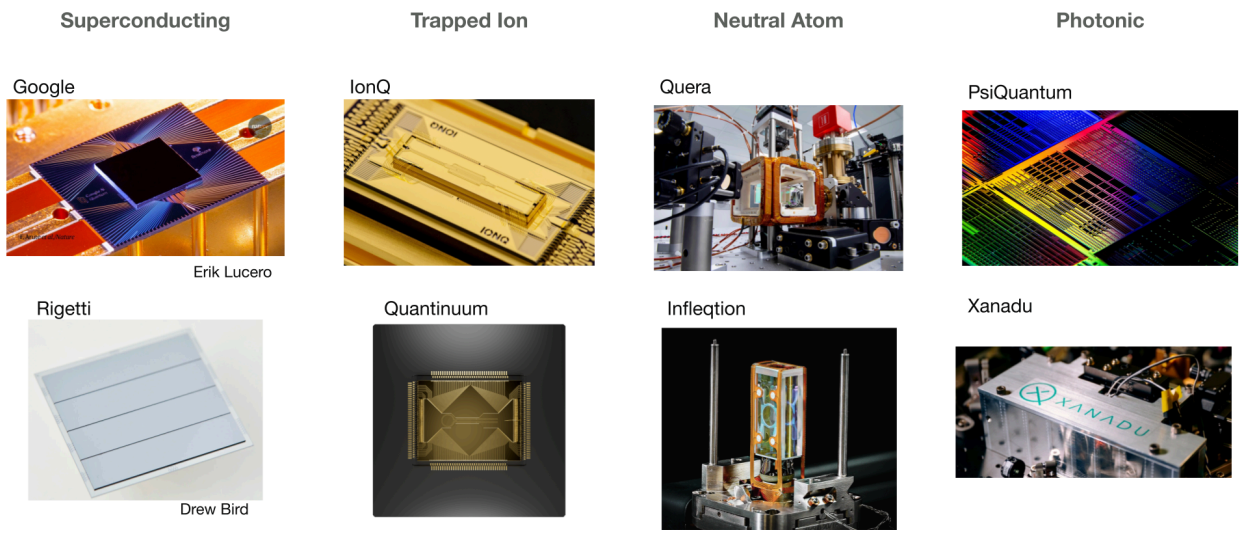
This shift from monolithic devices to modular, distributed devices will reshape compute infrastructure. Future data centers will integrate QPUs alongside CPUs and GPUs, with quantum and classical resources orchestrated together to run hybrid applications. Realizing this vision will require interconnect technologies, network-aware performance software stacks, and operational models for deploying, managing, and scaling distributed quantum workloads.

In this white paper, we outline the architecture and design principles behind quantum computer networking and the concept of the quantum data center. We begin by examining why QPU platforms are so diverse, spanning different materials, control modalities, and device architectures, and how those differences influence networking requirements and trade-offs. We explore the protocol layers and control stack needed to make distributed quantum computing practical, including compilers, schedulers, and orchestrators that can translate application intent into coordinated actions across QPUs, links, and shared network resources.

There has been a lot of recent progress toward integrating quantum systems with existing high-performance computing (HPC) and cloud infrastructure, where hybrid computing is already demonstrating how quantum accelerators can complement classical processors. The quantum data center won't arrive all at once, but the architectural decisions we make now will determine how quickly quantum computing can scale from isolated devices into reliable, interoperable, and commercially valuable computing systems.

Types of Quantum Processing Units

There are many ways to build a Quantum Processing Unit (QPU), using different materials, physical systems, and device architectures. The image below shows some examples of the different platforms, however, this is not a comprehensive list.



On the left are superconducting-circuit QPUs. These QPUs implement “artificial atoms” (superconducting qubits) on a chip. Because these devices operate only at extremely low temperatures, they run inside cryogenic systems, often appearing as chandelier-like dilution refrigerators. This cryogenic structure provides both the temperature environment and the dense wiring and electronics needed to control and read out the qubits.

The center features trapped ion QPUs and neutral atom QPUs. These QPUs use real atoms and ions in a controlled, isolated system to enable quantum computation. An advantage of these atom-based platforms is their intrinsic uniformity: atoms and ions are fundamentally identical, which can reduce some sources of variability that arise in the fabrication process of other platforms. However, these systems introduce their own laser engineering challenges in trapping, control, and scaling.

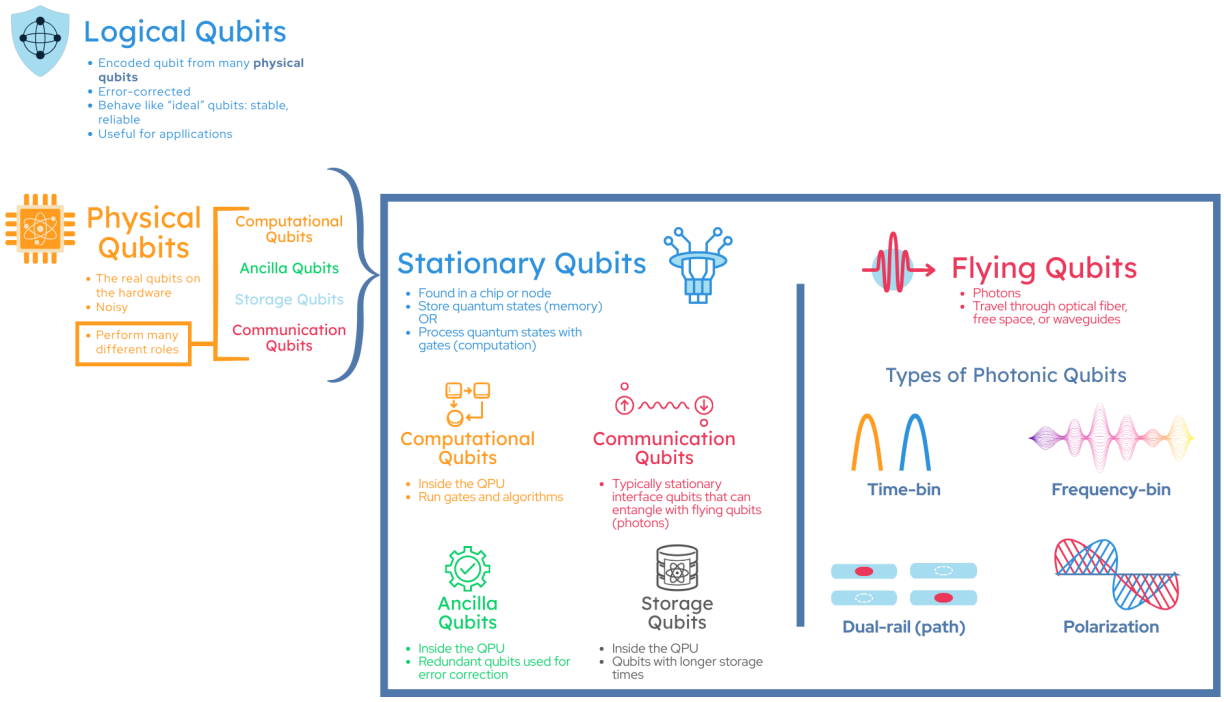
Pictured on the right are examples of photonic QPUs, which encode quantum information in light and use photonic components to generate and process quantum states. Photonic architectures are especially well-suited to producing large entangled resource states (for example, cluster states) and performing computation through controlled measurements on integrated circuits. However, these platforms depend highly on the quality and uniformity of the chip fabrication process.

Platform diversity extends beyond the hardware itself. Programming models, gate implementations, error mechanisms, and system control requirements vary widely across QPU types. This variability influences how each platform may integrate into a networked, data-center-scale environment.

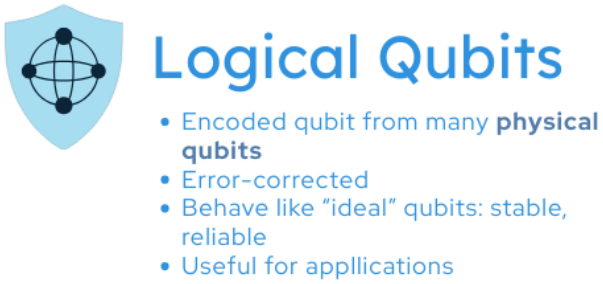
So which of these platforms will play a primary role in the data center? Today this remains an open question. While it's possible that one approach may emerge as dominant for broad deployment, it is also possible that multiple platforms will coexist. In this approach, systems will be optimizing different QPUs for different workloads, performance targets, or roles within a larger distributed algorithm. Each of these approaches has demonstrated meaningful progress and paths toward scalability and fault tolerance.

Types of Qubits in Networked QPUs

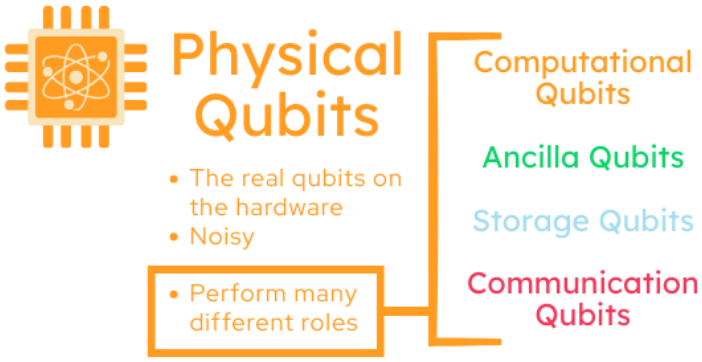
Before we go further, it helps to establish some shared vocabulary around qubits. In networked quantum systems, the term "qubit" can refer to several layers of abstraction: from hardware-level qubits on a device to error-corrected qubits used by applications, and different qubits may play different roles inside a QPU and across a quantum network.



Logical qubits. Error-corrected qubits. Information is encoded across many physical qubits using a quantum error-correcting code. A logical qubit is designed to behave like an idealized qubit for computation, with errors detected and corrected so that the error rate is much lower than any single physical qubit. At the highest level, algorithms are typically expressed in terms of logical qubit operations, even though the hardware must implement them using many physical qubits.



Physical qubits. The actual hardware qubits on a QPU or other quantum device. Depending on the platform and architecture, physical qubits may be specialized for different functions.



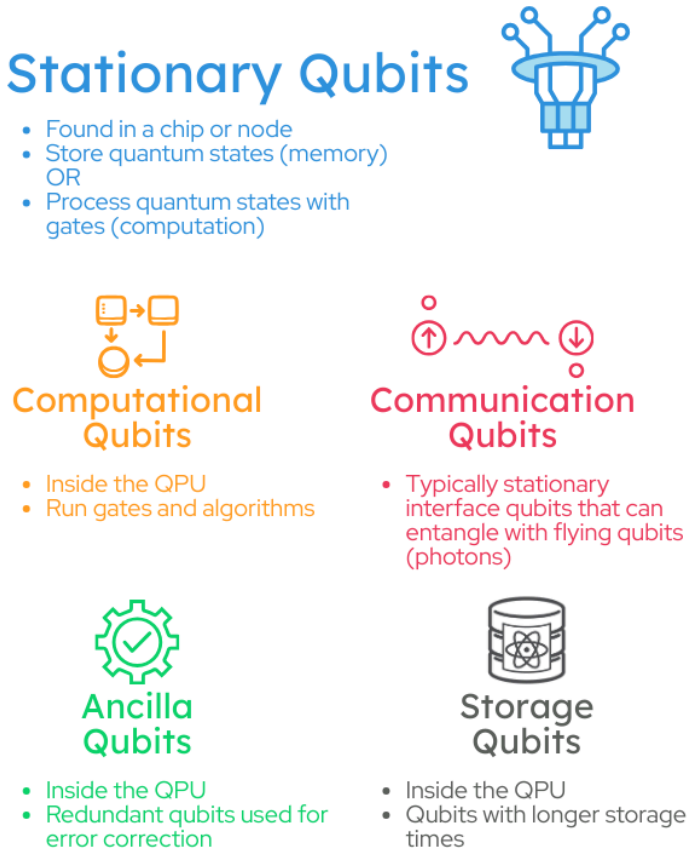
Computational, ancilla, storage, and communication describe functional roles (typically implemented by stationary physical qubits):

Computational qubits. These qubits execute gates and carry the algorithm's quantum state during computation.

Ancilla qubits. These qubits are used for measurement and error correction, such as extracting syndrome information without directly measuring (and collapsing) the computational qubit's logical state.

Storage qubits. These qubits are optimized to preserve quantum states longer or with lower cross-talk.

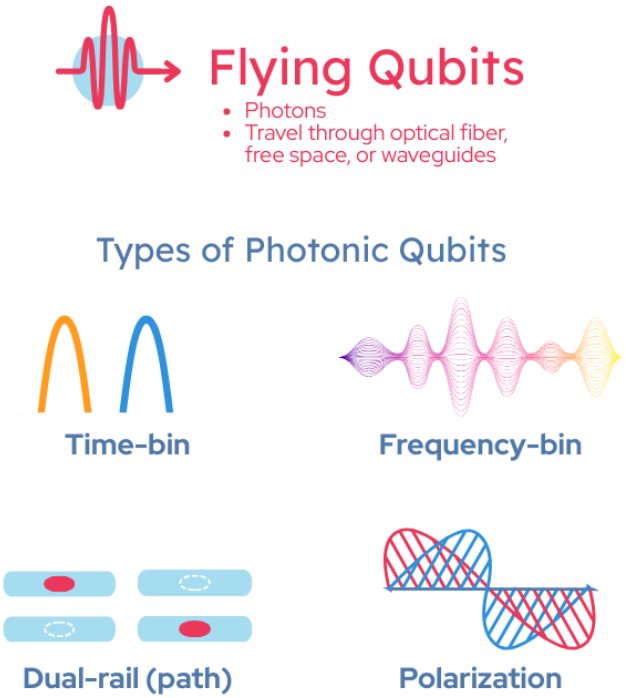
Communication qubits. These qubits – sometimes called interface qubits – mediate the interaction between stationary qubits and flying qubits (typically photons) and are ultimately used to create entanglement between qubits in different QPUs. The specific design of this interface, or quantum interconnect, is very platform-dependent. For example, what a quantum interconnect looks like for a superconducting system can be very different from a light-matter interface for a trapped-ion or neutral-atom implementation.



Flying qubits. These qubits move through a channel and are most commonly photonic qubits traveling in optical fiber or free space. Flying qubits are the workhorse of quantum networking because photons can carry quantum information over distance with low decoherence compared to other qubit platforms, and are compatible with existing optical network infrastructure. Within flying qubits in the photonic regime, there are different ways to encode quantum states in light. There are different photonic degrees of freedom, or dimensions, that can be used to encode a quantum state, including:

- Time-bin encoding. The qubit is encoded in a superposition of the photon's arrival-time modes (early vs. late).
- Polarization encoding. The qubit is encoded in the photon's polarization (e.g., horizontal, vertical, diagonal, circular).

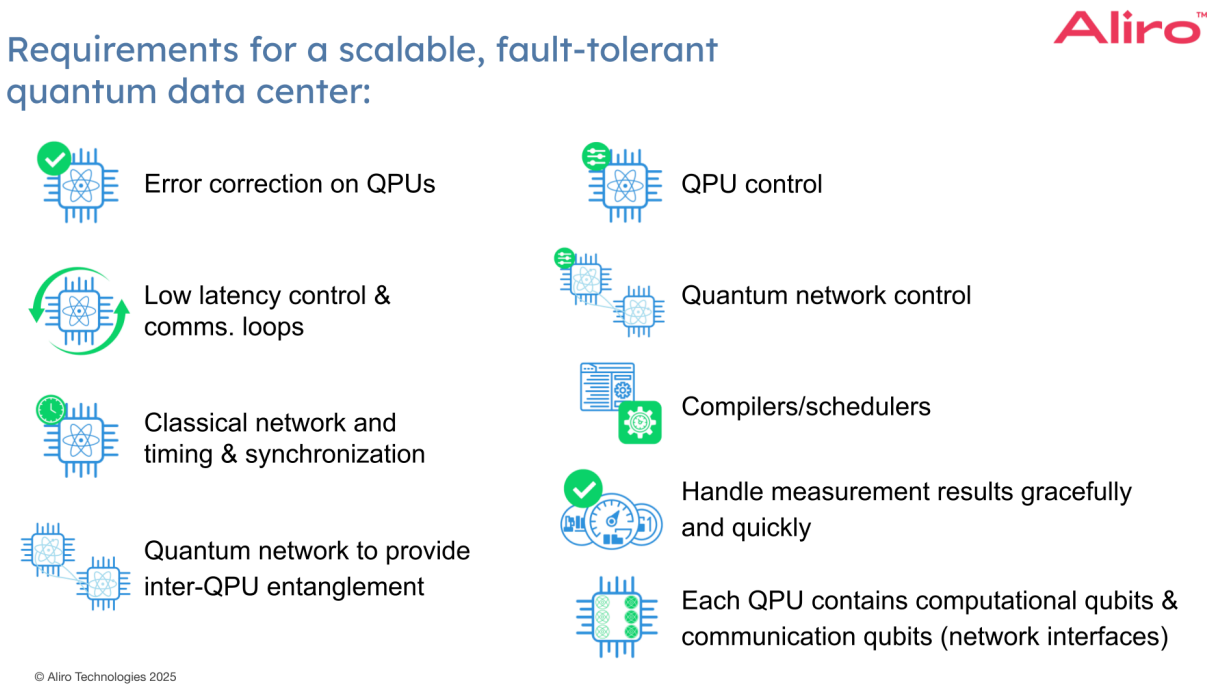
- Frequency-bin encoding. The qubit is encoded in discrete frequency modes.
- Spatial/path encoding. The qubit is encoded based on which optical path a photon takes, often manipulated using beam splitters and phase shifters.



These terms provide a useful shared vocabulary and hint at the complexity of networked quantum systems. Some qubit types describe a level of abstraction (logical vs. physical), others describe the function or role of the qubit (computational, ancilla, memory, communication), and still others reference network transport (stationary vs. flying). A practical quantum data center will likely incorporate many of these categories simultaneously, coordinated through an architecture and control stack designed to support computation and networking holistically.

Requirements for Quantum Data Centers

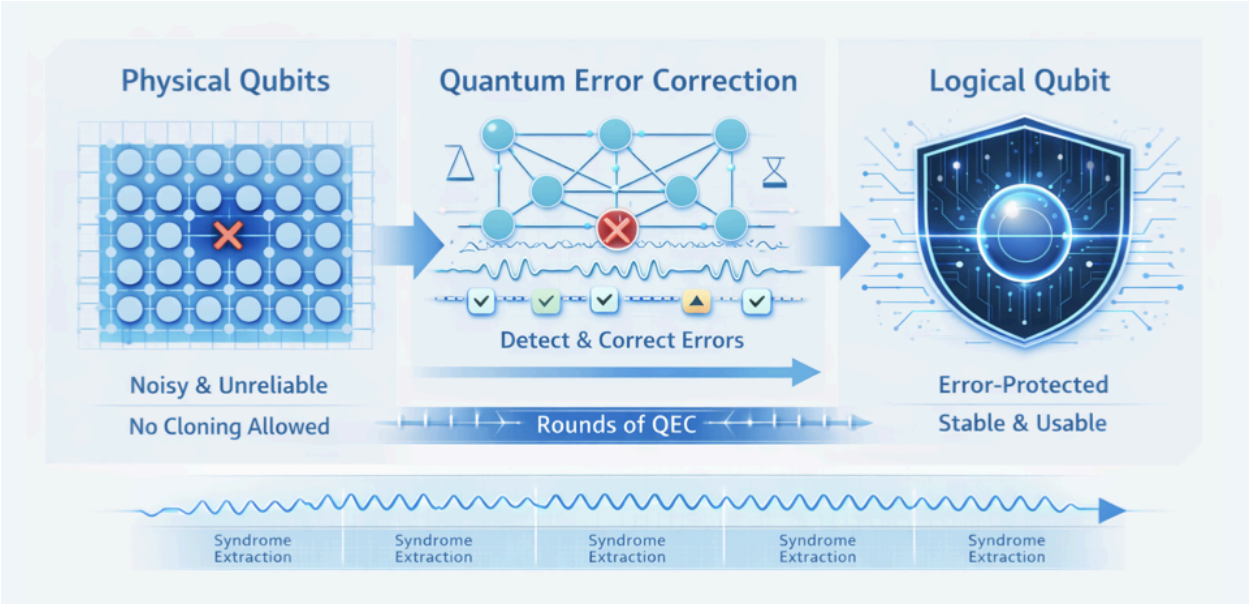
What is needed to enable this vision of a scalable, fault tolerant quantum data center?



Error correction. How are logical qubits built from an array of physical qubits? The key is quantum error correction (QEC), a set of techniques to encode one logical qubit into an entangled state spread across multiple physical qubits. This encoding allows the system to detect and correct errors caused by noise, decoherence, and imperfect gates without directly measuring and destroying the logical quantum information needed for computation.

In recent years, the field has made major progress, including early demonstrations of logical qubits and repeated error detection/correction cycles that preserve encoded quantum information. As a result, quantum error correction and how effectively qubits can suppress errors over time has become an important benchmark for quantum computing progress.

In general, quantum error correction works by encoding a logical qubit across many physical qubits and repeatedly measuring error syndromes (often using dedicated ancilla qubits) to detect and correct errors without directly measuring the encoded logical state.



While there are analogs to classical redundancy-based error correction techniques, it is fundamentally more complex: qubits have finite coherence times, operations are noisy, and quantum information cannot be copied. For these reasons, QEC is widely viewed as a critical enabling technology for scaling from today's noisy devices to fault-tolerant, reliable quantum computing.

Low-latency control and communication. Error correction can be viewed as a kind of feedback loop. During computation, ancilla qubits are used to extract error syndromes, classical measurement outcomes that indicate which error checks on the data qubits were violated. Those check results are analyzed by a classical controller, which decides what correction is needed. The system then applies a fix immediately, or keeps track of the fix in software and accounts for it later.

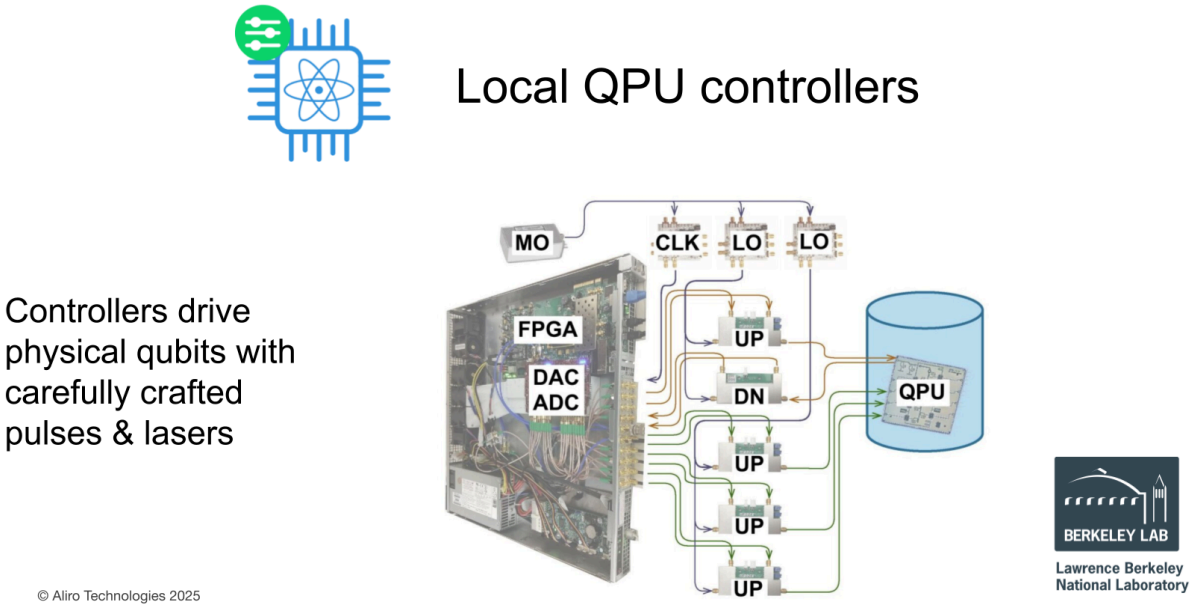
For a quantum data center, this places really strict requirements on the latency of the control systems and the communication between hardware components. Quantum states naturally decohere (drift and degrade over time) so delays in detecting errors, processing them, and responding to the errors will cause them to accumulate. These low-latency demands apply within a single QPU, and become even more important in networked QPUs, where distributed operations introduce additional timing and communication constraints.

Classical network and timing & synchronization. Building a quantum data center requires a robust classical control and communication network that distributes a variety of classical signaling and network communications. These classical interactions operate in multiple timing regimes. Some tasks must be performed in real time, such as distributing timing signals,

coordinating gate sequences, collecting measurement outcomes, and closing fast feedback loops for operations like error correction. Other tasks can tolerate longer latencies, including higher-level orchestration, compilation, and many scheduling decisions, which may run on separate systems and at slower cadences.

The main point is that QPUs, and especially networked QPUs, place stringent requirements on latency, jitter, and synchronization in the classical control plane. Without tight timing and reliable classical communications, it becomes difficult to scale distributed workloads.

Local QPU controllers. To operate a QPU, we need local control hardware that can drive physical qubits and communicate with the quantum states on the device itself. The image below shows an example of a QPU controller system. Essentially this controller, sometimes referred to as a pulse processing unit, is driving the operations being carried out on physical qubits with very carefully crafted pulses, sometimes complex laser systems depending on the underlying QPU platform.



These controllers implement everything the QPU needs to manipulate the qubits: executing the algorithm, preparing qubit states, executing routines for syndrome detection and error correction, and many other operations. In practice, nearly all low-level operations flow through this local controller.

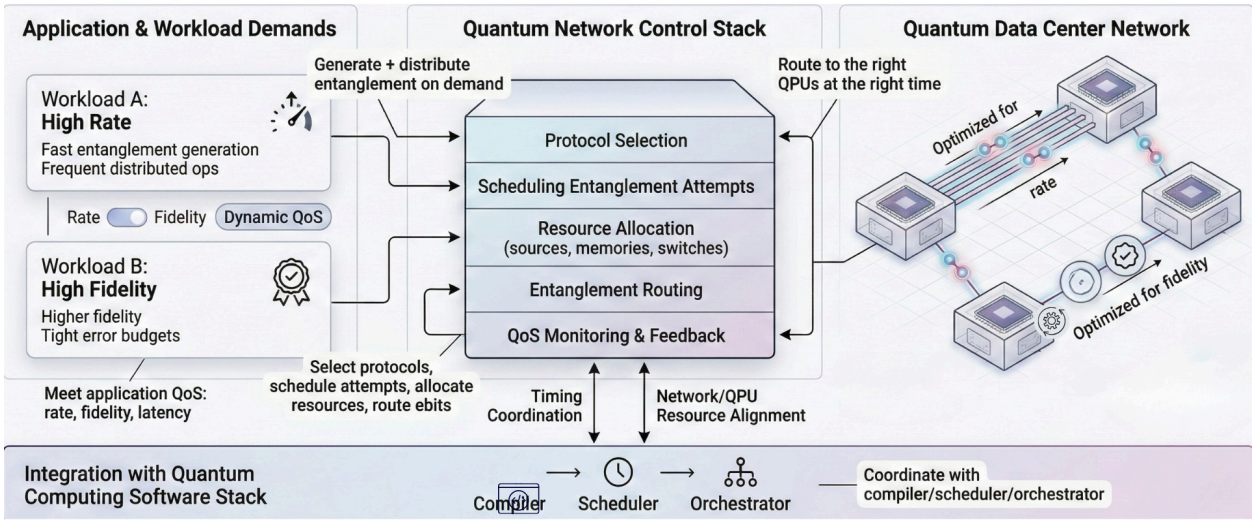
While the role is consistent, the details are platform-dependent. For example, superconducting qubits are typically controlled with precisely timed microwave pulses. In contrast, trapped-ion and neutral-atom systems rely on lasers to manipulate the atoms or ions, or perhaps even shuttling them around and moving them physically. Across platforms, the common requirement

remains the same: tightly timed control and measurement delivered through the local QPU controller.

Quantum network to provide inter-QPU entanglement. To scale beyond a single, monolithic quantum computer, the quantum data center will look more like today's HPC and cloud environments: clusters of processors working together, coordinated by software and connected by networks. The quantum network provides the connections at the quantum level, with entanglement between these different qubits on different QPUs in the network. This shared entanglement becomes a resource that enables multiple QPUs to behave like parts of a larger, coherent system.

Within a quantum data center, local entanglement is created and used within a single QPU as part of error correction protocols and algorithm execution. Entanglement is also generated across QPUs by the quantum networking layer and then consumed by distributed operations, such as teleportation, remote state preparation, qubit transfer, and remote gate execution. In other words, the quantum network acts as an entanglement supply chain for the data center, providing the quantum links that make inter-QPU computation possible.

Quantum network control. Just as QPUs require local controllers and software drivers, a quantum data center also needs a control stack for the quantum network. This network control plane is responsible for generating this entanglement, for distributing it and routing it to the right places at the right times and meeting the quality of service that the application or the algorithm may demand. Some applications and algorithms may require very fast rates of entanglement between certain processors, others may require maybe less of a stringent rate, but a much higher fidelity.



A core requirement is meeting application-level quality of service (QoS) needs and dynamics. Different workloads may demand different trade-offs. For example, high entanglement

generation rates to support frequent distributed operations, or higher-fidelity entanglement when error budgets are tight. The network control stack manages these trade-offs by selecting protocols, scheduling entanglement attempts, allocating hardware resources, and routing entanglement through the network as needed.

To do this effectively, the quantum network control stack must also integrate with the broader quantum computing software stack, especially compilers, schedulers, and orchestrators, so that network resources and timing are coordinated with the execution of the application and the availability of QPU resources.

Compilers and schedulers. In classical computing, developers write code at a high level without needing to fully understand transistors. Quantum computing aims for the same abstraction: applications describe algorithms at a logical level, and a compilation stack translates that intent into native operations the hardware can actually execute.

This translation is nontrivial because QPU platforms differ widely in their native gate sets, connectivity, noise characteristics, error correcting codes, and control primitives. As a result, compilation typically spans multiple layers, including breaking down and decomposing these logical gates into native operations, and error correcting circuits, deciding where logical qubits live on a device (or across devices) given connectivity constraints, resource management, and optimization.

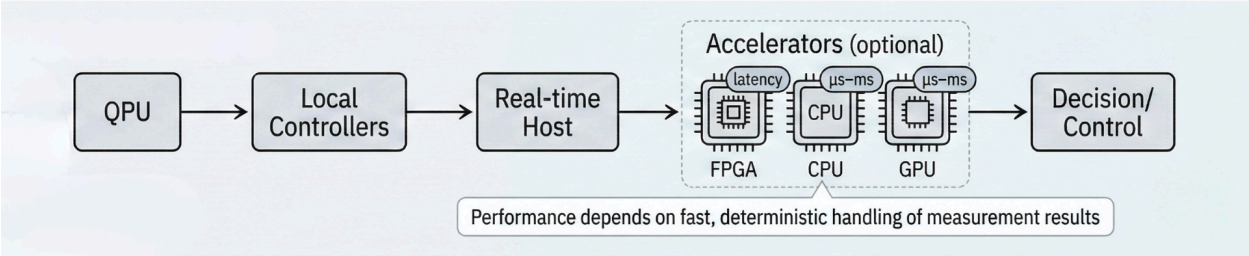
Compilation alone isn't sufficient. Quantum programs run under tight operational constraints: qubits decohere, some operations must occur in specific sequences, and routines such as error correction and entanglement generation introduce additional dependencies. This is where scheduling becomes essential.

Schedulers coordinate when operations happen and which resources they use across the full data center stack. In a networked setting, schedulers must align QPU operations with quantum network availability, such as timing, entanglement generation, reserving communication qubits, and coordinating distributed gates or teleportation steps across individual QPUs and across the network as a unified system. Compilers translate what to run, while schedulers and orchestrators manage when and where it runs.

Fast handling and feedback. At the end of any quantum computation routine, we extract results by measuring qubits, and those measurements produce classical data, strings of 1s and 0s (sometimes even 2s for qudit-based systems). Measurement isn't only used at the end of a program. It plays an ongoing role in quantum computing in at least two critical ways: error detection and algorithm control/outputs. Repeated measurements of ancilla qubits produce error syndrome data that drives decoding and recovery for error correction. Some algorithms rely on intermediate measurements to determine the final result or to choose the next operation, similar to a classical feed-forward algorithm.

Because of this, quantum systems must be able to capture, move, and process measurement data quickly and reliably. Measurement operations vary across QPU platforms in implementation, duration, and fidelity, but the systems' requirement is consistent: support low-latency readout and real-time response when the next steps depend on measurement outcomes.

In practice, measurement results flow from the QPU into the surrounding classical stack: local controllers and real-time hosts, and sometimes accelerators such as FPGAs, CPUs, or GPUs which process the measurement data in different ways at varying latencies. For many workloads, performance depends on a tight, deterministic loop where the system is handling these measurement results gracefully and quickly with low latency and real time feedback loops.



Qubits. To realize this vision of a quantum data center, each QPU needs to contain the computational qubits that will carry out the workflow, as well as the communication qubits (sometimes referred to as quantum interconnects) to ensure the QPUs can interface with the photonic quantum network.

There are many more requirements than what is presented on this list, such as wavelength management, frequency conversion, cryogenics, I/O cabling, and more. These are the basic core requirements for this scalable, fault tolerant quantum data center.

<Edited 03>

Protocols used to Interconnect QPUs

What protocols are used in these quantum networks to connect quantum computers?

Components overview

There are a wide variety of components that can be found in quantum networks. This is not an exhaustive list, but these components and devices are central to a quantum network.

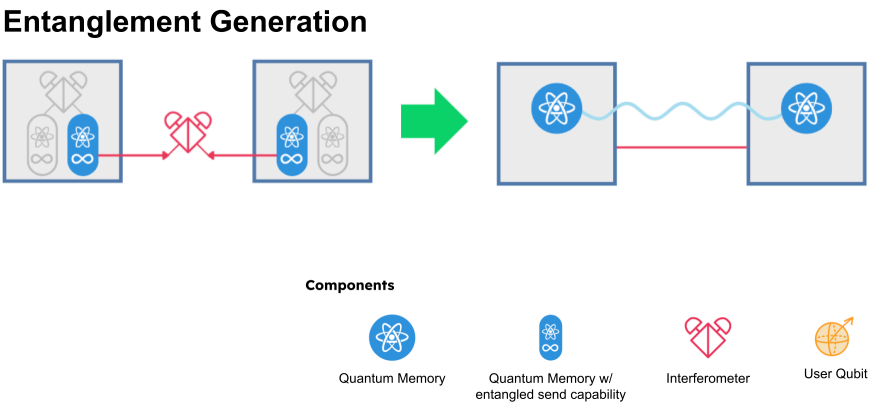
Quantum memories. These devices can store qubits for a finite amount of time, which is useful for buffering and synchronization. In some architectures, a (stationary) memory qubit on a QPU can be entangled with a flying qubit, enabling remote entanglement between distant nodes. In other architectures, quantum memory banks can be independent components in the inter-QPU network fabric.

Interferometers. These are devices that carry out photonic interference, an important process for entanglement-based protocols. Quantum networks typically use photons as the flying qubits, carried in optical fiber in the case of quantum data centers. In many architectures, interferometers are used in Bell-state measurement setups to support higher-level entanglement distribution protocols such as entanglement swapping and routing.

Stationary and flying qubits. Quantum networks typically include stationary qubits, which reside at fixed device locations, and flying qubits, which are commonly photons traveling through an optical channel that carry quantum states between nodes and mediate remote entanglement.

These building blocks make up the physical layer infrastructure required for executing the entanglement-based protocols that a quantum data center will use to interconnect QPUs.

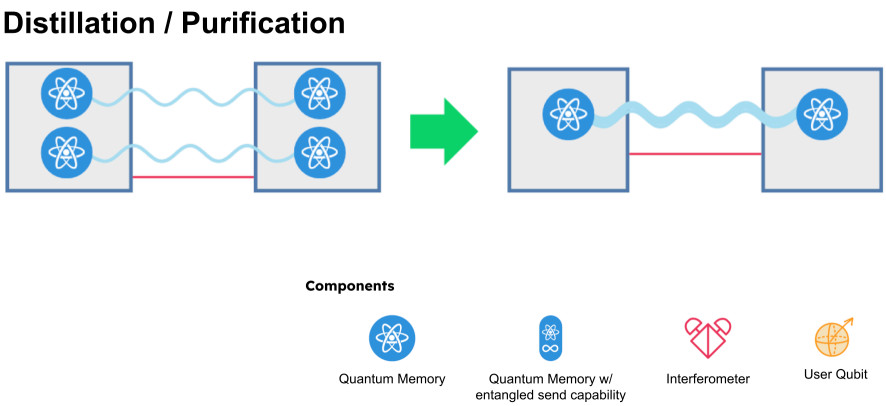
Entanglement generation



Many entanglement generation protocols use photon interference to create remote entanglement. In one common approach, two memories each emit a photon that is entangled with the memory. Those photons then arrive at an interferometer, ideally synchronously, so they are indistinguishable from each other. When this happens, they produce certain patterns that result in entanglement across those memories. This probabilistic method of generating entanglement is referred to as a meet-in-the-middle architecture. Other architectures may use photon sources that are located between nodes, sending out pairs of entangled photons (one to each node) for a midpoint-source architecture. Different architectures will place different requirements on the components and the broader system.

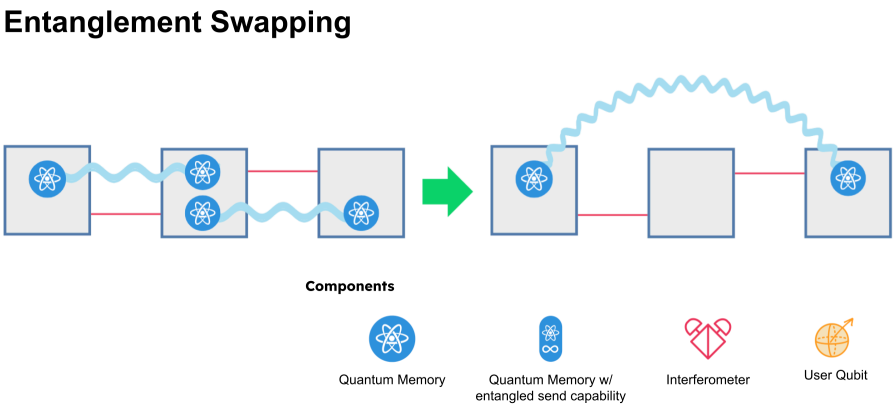
Supporting multiple approaches to entanglement generation and distribution provides flexibility in how a quantum data center interconnects QPUs.

Entanglement distillation / purification



Another core protocol in quantum networking is entanglement distillation, also called purification. Many distributed applications require entanglement above a minimum fidelity to meet their error budgets. In addition to generating entanglement quickly (rate), the network must often improve its quality of entanglement. Distillation protocols take multiple independent, lower-fidelity entangled pairs and produce fewer, higher-fidelity pairs. This provides a practical lever for network-level resource optimization.

Entanglement swapping

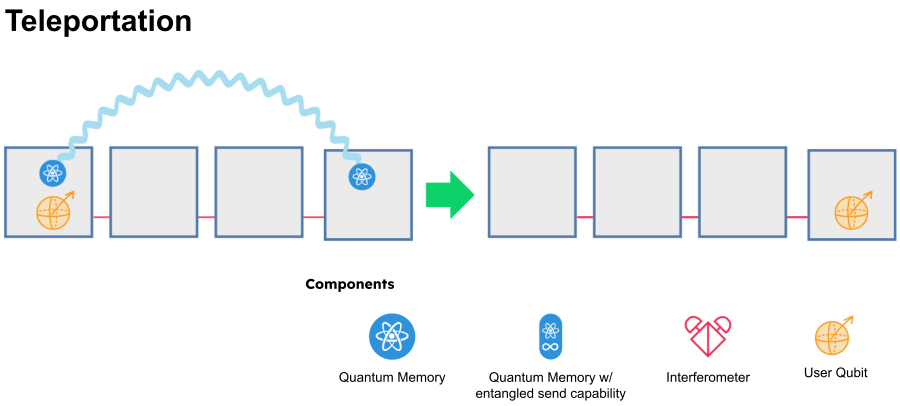


Entanglement swapping is a quantum networking protocol used to extend the range of entanglement beyond a single link. The basic idea is to start with elementary, point-to-point entangled links. For example, one entangled pair shared between devices A and B and another between devices B and C. The middle node B performs a measurement (a Bell state measurement) on its two local qubits (one from each pair originating at node A and node C). Nodes A and C then become entangled, even though they never interacted directly. In this way, entanglement swapping “stitches” short entangled links into longer-range entanglement, and it

serves as a foundational mechanism for multi-hop entanglement distribution in quantum repeaters and quantum data center networks.

In a quantum data center, swapping can also function as a form of entanglement routing, enabling entanglement to be established between the specific QPUs (or nodes) that need to perform a distributed operation.

Teleportation

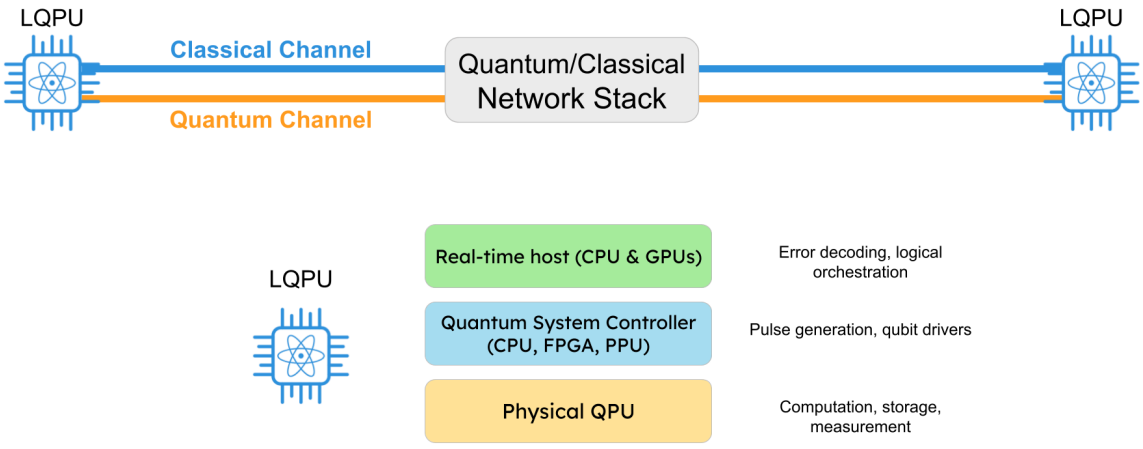


Quantum teleportation is a protocol for transferring an unknown quantum state from one location to another, such as from one QPU to a different QPU, without physically sending that state through the network. Instead, teleportation uses a pre-shared entangled pair between the sender and receiver, and a small amount of classical communication (typically two classical bits). The sender performs a joint measurement between the qubit holding the state and their half of the entangled pair. The measurement outcome is then sent over a classical channel to the receiver, who applies a corresponding correction operation. After this step, the receiver's qubit becomes the original quantum state. The original state is not duplicated, it is effectively transferred, consistent with the no-cloning principle.

In a quantum data center, teleportation is a powerful tool for moving quantum information between QPUs, enabling distributed algorithms by consuming entanglement as a networking resource.

Compiling, Scheduling, and Orchestrating for Quantum Computation

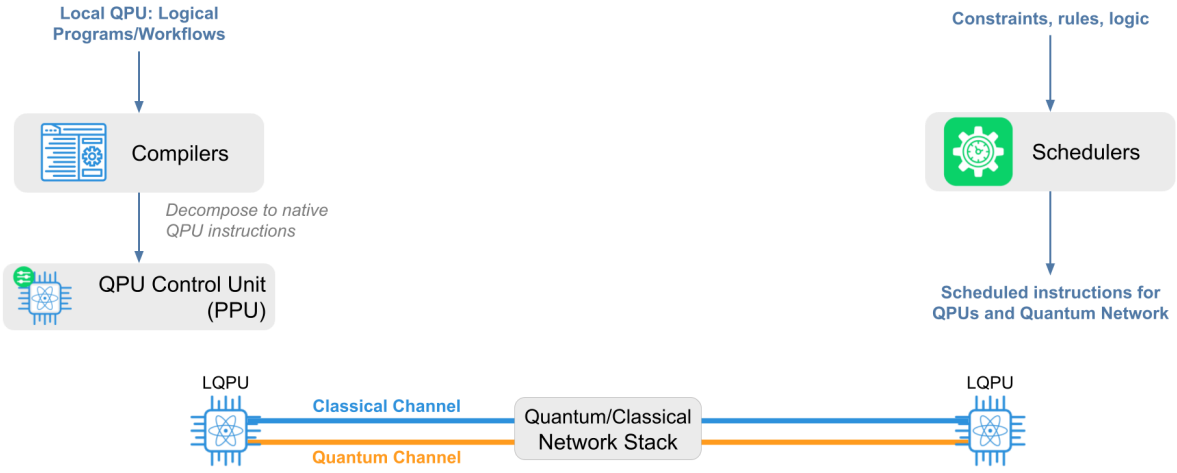
A quantum data center requires both quantum and classical network connectivity. The quantum channel supports entanglement distribution, while the classical channel carries timing, control messages, measurement outcomes, and coordination signals. Together, these are managed by a quantum/classical network stack that sits between logical QPU (LQPU) nodes.



An LQPU is more than the physical QPU technology (superconducting, trapped-ion, neutral-atom, etc.). It is an operational compute unit that includes the physical device plus the surrounding classical infrastructure needed to run workloads reliably. As shown in the diagram above, that stack typically includes:

- A real-time host (CPU plus GPU accelerators) to handle tasks like syndrome processing/decoding and logical orchestration.
- A quantum system controller (often CPU/FPGA/PPU hardware) for tasks such as pulse generation, laser control, and/or qubit drivers.
- The physical QPU, where computation, storage, and measurement occur.

Compilers and schedulers are also essential to quantum data centers.



Compilers translate high-level quantum programs into native QPU instructions that match the platform's gate set, connectivity, and control primitives. Schedulers then determine when and where those instructions execute, respecting constraints from the algorithm (timing, resources, and coherence/error budgets) and the hardware across both the QPUs and the quantum network.

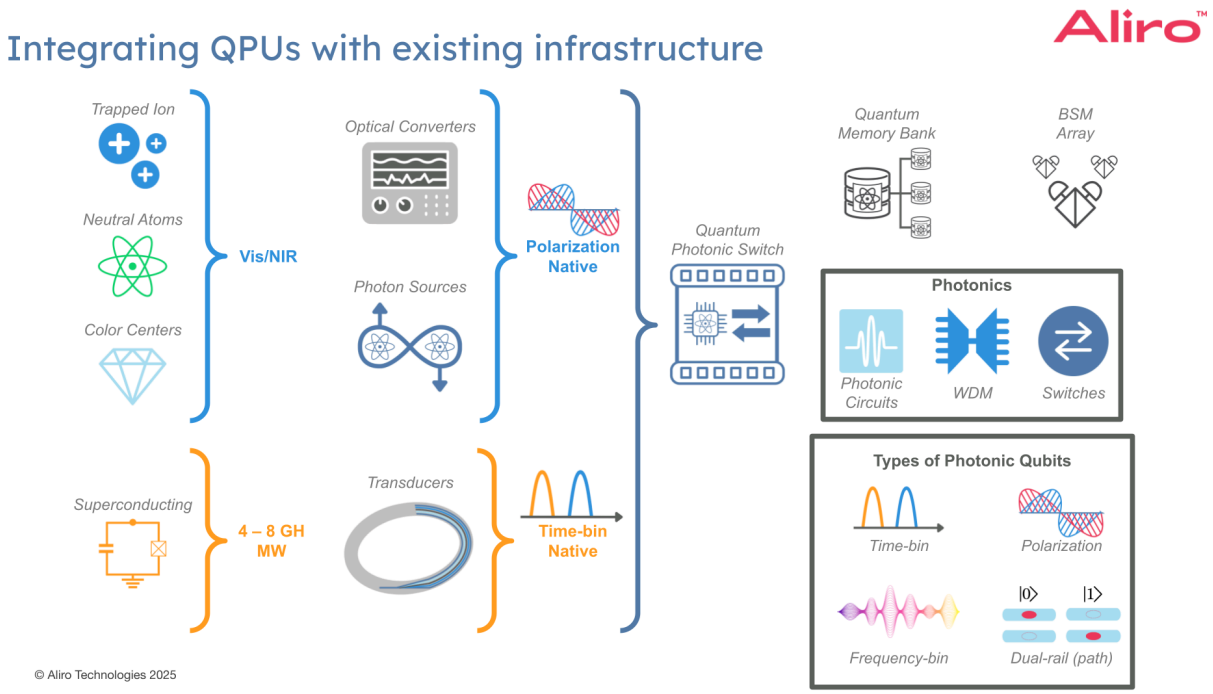
To provide the services needed to generate and deliver entanglement between LQPUs, and to make those services usable by applications, a quantum data center needs an orchestrator that coordinates entanglement requests, allocation, and delivery across the cluster.

This orchestration layer distributes workloads, manages network protocols, and enforces Quality of Service targets (such as entanglement fidelity, generation rate, and latency) based on what the workflow demands. It must coordinate with the schedulers and compilers and track current network resources and link conditions (available interface qubits, path options, and entanglement inventory).

In practice, the compilers, schedulers, and orchestration layers must work together tightly so distributed workflows across QPUs on the network can execute reliably and efficiently.

Realistic Implementation of QPU Networks

Moving from single monolithic QPUs toward a quantum data center linking many QPUs, the networking architecture has to accommodate an important reality: different QPU platforms "speak" very different physical languages. They operate at different control frequencies, use different interfaces for generating photons, and naturally align with different photonic encodings.



At the hardware level, QPUs span very different electromagnetic regimes. For example, superconducting qubits are controlled in the microwave domain (often a few GHz). Many matter-based platforms, such as trapped ions, neutral atoms, and color centers, interact with light in the visible and near-infrared (Vis/NIR). This mismatch is notable, because a scalable quantum data center will typically rely on photons as the flying qubits, carried over photonic links that can leverage mature telecommunications fiber and optical infrastructure.

To bridge these regimes, each QPU will have a quantum network interface (sometimes called a quantum interconnect). The specific implementation is highly platform-dependent, but common building blocks include:

Transducers. For superconducting systems, a key challenge is converting fragile quantum states between microwave and optical frequencies with low added noise and high efficiency. This is an active engineering research area precisely because the frequency gap is large and preserving quantum coherence through conversion is difficult.

Optical frequency converters. For platforms that already operate in Vis/NIR, the interface problem is often closer to optical frequencies, but still nontrivial. Frequency conversion may be used to translate between wavelengths (for example, from a platform's native optical transition toward telecom-friendly wavelengths) while preserving the quantum state.

Photon sources and entangled-pair generation. Some architectures use dedicated photon sources with dual-wavelength approaches. These components produce photons compatible with a given QPU on one side, and photons compatible with the photonic network on the other

side. The goal is the same: create a reliable pathway for memory/processor qubits to become entangled across nodes.

Photonic qubit encodings. There is diversity in how quantum information is carried in light. Photonic qubits may be encoded in time-bin, polarization, frequency-bin, or path (dual-rail) degrees of freedom. Certain QPU interfaces and link technologies naturally favor particular encodings, which means a practical quantum network for QPUs must be designed with encoding compatibility in mind. Where necessary, mechanisms for translation or interoperation (e.g., through photonic circuits and carefully engineered measurement stages) can be integrated.

The quantum network fabric provides the shared infrastructure that makes a multi-QPU system behave like a coordinated whole. This can include:

- Components that generate, manipulate, and stabilize optical states.
- Wavelength-division multiplexing (WDM) to increase efficiency and rates of entanglement services.
- Switches and routers (WDM-compatible) to ensure photons are sent to the correct destinations
- Quantum memory banks to buffer quantum states for synchronization and scheduling
- Bell-state measurement (BSM) arrays that enable interference-based entanglement generation and support protocols like entanglement swapping and routing

Taken together, these elements show what realistic integration looks like: heterogeneous QPUs connected through photonic interfaces, unified by a quantum network fabric that can generate, route, buffer, and verify entanglement. This is also the path to leveraging existing infrastructure, especially optical components and networking concepts, while meeting the unique constraints of quantum networks.

NVQlink by Nvidia

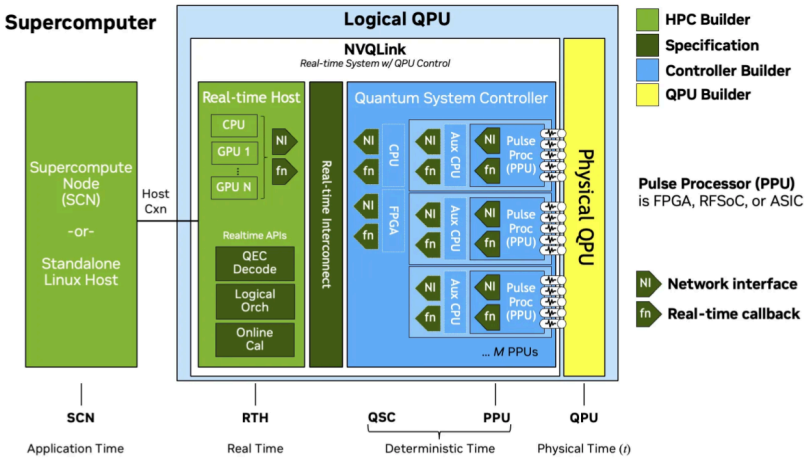
NVIDIA has proposed a reference architecture, NVQLink. It's an interconnect approach meant to support connectivity between QPUs, GPUs, and the broader supercomputing/HPC stack. The goal is to provide the industry with a common reference point for how these systems can fit together.

NVQLink

The role of GPUs in a quantum datacenter

GPUs for control and variational optimization

Networking and latency considerations for GPU integration



The GPU's role is to support tasks like quantum error correction, including decoding syndrome data, and it can also support logical orchestration of the system around the QPU: driving the QPU controllers, pulse drivers, etc. What's particularly interesting in the NVQLink reference architecture is that it breaks the system into different timing regimes,: a regime where the GPU is more tightly coupled to the logical QPU stack, and one where this coupling is much looser.

In this tightly coupled regime, the GPU relies on real-time interconnects to communicate with the rest of the control system, often RDMA over Ethernet or other kinds of interconnect technologies. This is a classical link, but it needs to be fast enough to support real-time workflows and communication with the rest of the LQPU control stack. This is very useful for driving the error correction feedback loop, but also doing the logical orchestration of the physical QPU itself.

The loosely coupled regime is closer to the application and algorithm layer, where GPUs interact with the logical QPU (LQPU) abstraction. This opens the door to hybrid workflows where GPUs and QPUs work together on a shared computation, i.e. hybrid quantum-classical algorithms. An example of this is variational algorithms, where certain quantum workloads are carried out on the QPUs, results are read out, and then the GPUs run an optimization or feedback loop to drive the next iteration of that circuit and adjust parameters until the computation converges.

Overall, NVQLink provides a useful framework for how latency constraints and interconnect requirements change depending on how tightly the GPU is coupled to the QPU control stack, or

interacting at the application layer. As this reference architecture gains adoption within industry, it provides a measure of standardization around integration of QPUs into existing HPC/supercomputing environments.

In the context of a quantum data center, the next evolution is to extend this approach beyond a single logical QPU toward a network of logical QPUs. Rather than a monolithic system, the long-term vision is a distributed environment with multiple QPUs working together. The question then becomes how architectures like this scale to support a networked model when coordination must span both the classical and the quantum layers.

Common Questions

1. What does a realistic "least common denominator" interface between different QPU platforms look like? Is it a hardware standard, a control-plane abstraction, or something else?

Standardization can help vendors interoperate and accelerate adoption, but timing matters: in a fast-moving field, standards introduced too early can unintentionally hamper innovation.

A useful near-term approach is already underway with reference architectures: not formal standards, but best practices that are widely adopted enough to create a shared "shape" for integration, similar to how classical HPC matured through layers of abstraction over time. From there, the ecosystem can iterate toward clearer abstractions and, eventually, more formal standards where they make sense.

2. You mentioned compilers, schedulers, and orchestrators. What's the most underappreciated software challenge in making distributed quantum workloads run across a network of QPUs?

All three are essential, but scheduling is often underrepresented in literature, research or products in the space.

One reason is the reality of widely differing timing regimes across QPU platforms, paired with the uncertainty of which QPU platform will "win" or play a primary role in the future quantum data center. For example, some platforms execute gates much faster, but have shorter coherence times and lower-fidelity operations, while others offer longer coherence times, higher-fidelity operations, but longer gate times. In a distributed system, you're not just scheduling operations on a single device, you're coordinating QPU instructions, network-dependent steps (like entanglement generation/consumption), and communication overheads across nodes, all while preventing bottlenecks where one subsystem blocks another. That cross-domain timing coordination is where scheduling becomes especially critical.

3. Do QPU networks work like a LAN or "quantum LAN"? What's different about QPU networks, if anything?

At a high level, there are similar architectural ideas (nodes, links, routing/control), and there can be overlap in components (photon sources, memories, interference/measurement modules).

Some of the same technology that appears in long-distance quantum networking also appears, at a smaller scale, in QPU networks.

One important difference is the roles devices play and the performance metrics they need to meet. Long-distance networks are dominated by challenges like loss over distance and repeater-oriented optimization. In a quantum data center, nodes are closer and the optimization focus shifts toward throughput, coordination, and tight integration with computation.

4. Conceptually, where does NVQLink fit in the stack?

NVQlink as a reference architecture has implications at different layers of the stack.

In the tightly coupled regime, GPUs sit close to the system controllers and help enable early logical QPU systems. In this regime, NVQLink fits best as a classical integration layer between GPUs and a logical QPU.

A physical QPU can't run useful workloads on its own. It needs a control stack (controllers, pulse generation, readout, real-time processing). When the physical QPU is bundled with that control and real-time compute around it, it becomes a logical QPU (the operational unit). NVQLink is about connecting GPUs to the logical-QPU stack so they can support time-sensitive tasks, such as processing measurement/syndrome data quickly, helping drive feedback loops (e.g., for error correction), and supporting real-time orchestration close to the QPU.

There's also a loosely coupled regime, where GPUs interact with the logical QPU at the application layer (hybrid algorithms, iterative workloads). NVQLink is primarily a framework for connecting GPUs to the logical-QPUs, especially where low latency matters, rather than the quantum network that connects QPUs to each other.

Conclusion

A quantum data center is not just a bigger quantum computer; it is a tightly integrated system-of-systems, where many technologies must interoperate: heterogeneous QPU platforms, network interfaces, photonic links, controllers, and the software stack that coordinates computation across devices. A central challenge is harmonizing differences between QPUs so that networks of logical QPUs can work together reliably and deliver value in an HPC setting.

There are three foundational concepts for the future quantum data center:

The quantum network is critical. Interconnecting QPUs through entanglement distribution is a foundational requirement for scaling toward fault-tolerant, modular quantum computing.

Software and control layers must be tightly coordinated. Compilers, schedulers, orchestrators, controllers, and error-correction workflows operate across multiple timing regimes. Making the system practical requires low-latency communication and careful coordination so that these layers work together efficiently.

This is just the beginning. Many core building blocks are advancing quickly, but there are still significant open problems, from quantum interconnects and control systems to networking protocols and end-to-end architecture. What's encouraging is the pace of recent progress, and the growing alignment around the need for scalable, networked quantum computing.

A Full-stack Solution for Quantum Networking

Scaling quantum computing will require more than larger individual devices. The emerging architecture for fault-tolerant, scalable systems points toward networked QPUs: multiple processors working together, coordinated by classical control and connected through entanglement services. In this model, the network is not an add-on. It becomes a core part of the compute fabric, enabling distributed quantum workloads, modular scaling, and new system architectures for the quantum data center.

Aliro is focused on making QPU networking practical. Our approach is to provide the tools and expertise needed to design and operate entanglement-driven network services that can interconnect QPUs, whether within a virtual lab environment, across a facility, or as a foundation for future quantum data centers.

Aliro's entanglement-based quantum networking solutions span orchestration, control, and simulation layers to provide the software and services designed to simulate, design, run, and manage quantum networks that support distributed quantum operations.

Aliro Simulator helps teams test and validate network behavior, characterize performance, and optimize hardware and protocols for the requirements that matter in QPU networking: entanglement generation and delivery, resource scheduling, and latency optimization.

Aliro's solutions are built by a team with deep experience in quantum networking and classical network engineering, with a focus on real-world interoperability and scalable architectures. To get started, or to extend your current efforts, reach out to discover how Aliro can support your QPU networking goals and accelerate the path toward networked quantum computing.

Bibliography

Caldwell, Shane A., et al. “Platform Architecture for Tight Coupling of High-Performance Computing with Quantum Processors.” arXiv, 29 Oct. 2025, arXiv:2510.25213. arXiv, https://arxiv.org/abs/2510.25213.

Choi, Charles Q. “New Interface Uses Light to Scale Up Quantum Computers.” IEEE Spectrum, 14 Oct. 2024, https://spectrum.ieee.org/quantum-network-interface.

Infleqtion. “Media Kit.” Infleqtion, n.d., https://infleqtion.com/media-kit/.

IonQ. “IonQ Delivers First Overseas Ion Trap to Switzerland, Progressing Development of Quantum Innovation Hub for EMEA.” Business Wire, 13 Aug. 2024, https://www.businesswire.com/news/home/20240813717294/en/IonQ-Delivers-First-Overseas-Ion-Trap-to-Switzerland-Progressing-Development-of-Quantum-Innovation-Hub-for-EMEA.

Khalifa, Othman O., Nur Sharif, Rashid Saeed, S. Abdel-Khalek, Abdulaziz Alharbi, and Ali Alkathiri. “Digital System Design for Quantum Error Correction Codes.” Contrast Media & Molecular Imaging, vol. 2021, Article ID 1101911, 2021, pp. 1–8. https://doi.org/10.1155/2021/1101911.

PsiQuantum. “Press Materials.” PsiQuantum, n.d., https://www.psiquantum.com/press-materials.

QuEra Computing. “Media Kit.” QuEra, n.d., https://www.quera.com/media-kit.

Quantinuum. “Quantinuum Announces Commercial Launch of New Helios Quantum Computer That Offers Unprecedented Accuracy to Enable Generative Quantum AI (GenQAI).” PR Newswire UK, 6 Nov. 2025, https://www.prnewswire.co.uk/news-releases/quantinuum-announces-commercial-launch-of-new-helios-quantum-computer-that-offers-unprecedented-accuracy-to-enable-generative-quantum-ai-genqai-302607869.html.

Shapourian, Hassan, et al. Quantum Data Center Infrastructures: A Scalable Architectural Design Perspective. arXiv, 9 Jan. 2025, arXiv:2501.05598 [quant-ph], https://arxiv.org/abs/2501.05598.

Sutcliffe, Evan, et al. Distributed Quantum Error Correction Based on Hyperbolic Floquet Codes. arXiv, 23 Jan. 2025, https://arxiv.org/html/2501.14029v1