

Distributed Quantum Computing: a Survey

MARCELLO CALEFFI^{*†‡}, University of Naples Federico II, Italy

MICHELE AMORETTI^{*§}, University of Parma, Italy

DAVIDE FERRARI^{*§}, University of Parma, Italy

DANIELE CUOMO[†], University of Naples Federico II, Italy

JESSICA ILLIANO^{¶†}, University of Naples Federico II, Italy

ANTONIO MANZALINI, TIM, Italy

ANGELA SARA CACCIAPUOTI^{*†‡}, University of Naples Federico II, Italy

Nowadays, quantum computing has reached the engineering phase, with fully-functional quantum processors integrating hundred of noisy qubits available. Yet – to fully unveil the potential of quantum computing out of the labs and into business reality – the challenge ahead is to substantially scale the qubit number, reaching orders of magnitude exceeding the thousands (if not millions) of noise-free qubits. To this aim, there exists a broad consensus among both academic and industry communities about considering the *distributed computing* paradigm as the key solution for achieving such a scaling, by envision multiple moderate-to-small-scale quantum processors communicating and cooperating to execute computational tasks exceeding the computational resources available within a single processing device. The aim of this survey is to provide the reader with an overview about the main challenges and open problems arising with distributed quantum computing, and with an easy access and guide towards the relevant literature and the prominent results from a computer/communications engineering perspective.

Additional Key Words and Phrases: Quantum Computing, Quantum Computation, Distributed Quantum Computing, Quantum Algorithms, Quantum Internet, Quantum Networks, Quantum Compiler, Quantum Compiling, Simulator

1 INTRODUCTION

The *Quantum Internet* [1, 2] is envisioned as the final stage of the quantum revolution, opening new communication and computing capabilities. In synergy with the classical Internet, the Quantum Internet will connect quantum processors and devices to achieve capabilities that are provably impossible using classical communication.

Within the last few years, a major effort is being undertaken by the research community and by the major ICT companies towards the Quantum Internet design and deployment.

^{*}These authors contributed equally to this article.

[†]Also with www.QuantumInternet.it research group, FLY: Future communications Laboratory, University of Naples Federico II.

[‡]Also with CNIT, National Inter-university Consortium for Telecommunications.

[§]Also with Quantum Software Laboratory, University of Parma.

[¶]Jessica Illiano acknowledges support from TIM S.p.A. through the PhD scholarship.

Authors' addresses: Marcello Caleffi, Department of Electrical Engineering and Information Technologies (DIETI), University of Naples Federico II, Naples, Italy, marcello.caleffi@unina.it; Michele Amoretti, Department of Engineering and Architecture, University of Parma, Parma, Italy, michele.amoretti@unipr.it; Davide Ferrari, Department of Engineering and Architecture, University of Parma, Parma, Italy, davide.ferrari1@unipr.it; Daniele Cuomo, Department of Physics, University of Naples Federico II, Naples, Italy, daniele.cuomo@unina.it; Jessica Illiano, Department of Electrical Engineering and Information Technologies (DIETI), University of Naples Federico II, Naples, Italy, jessica.illiano@unina.it; Antonio Manzalini, TIM, Turin, Italy; Angela Sara Cacciapuoti, Department of Electrical Engineering and Information Technologies (DIETI), University of Naples Federico II, Naples, Italy.

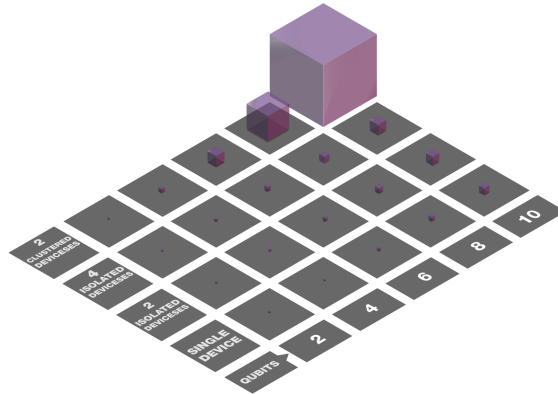


Fig. 1. Quantum computing power of isolated vs interconnected processors. The volume of each cube graphically represents the ideal – i.e., noise free – quantum computing power as the number of qubits within each processor scales. Figure reproduced from [12].

Within the EU, The Quantum Flagship’s projects are developing some of the most advanced physical quantum computing and quantum communication platforms in the world [3]. Specifically, long-term ambition of the Quantum Internet Alliance project is to build a Quantum Internet that enables quantum communication applications between any two points on Earth [4]. In this decade, a quantum communication infrastructure (EuroQCI) will cover the whole EU, including its overseas territories [5].

Overseas, the US government opened a number of publicly funded centers for quantum research in the last decade [6]. One of them is Q-NEXT [7], whose mission is to “deliver quantum interconnects and establish a national foundry to provide pristine materials for new quantum devices”. Q-NEXT’s vision is to help developing the technology that will enable applications in secure communication, distributed sensing, and scaling quantum computers. Another US public center is the Hybrid Quantum Architectures and Networks (HQAN) [8], whose goal is “[tackling] the challenge of scaling quantum processors by pursuing an alternative paradigm: distributed quantum processing and networks composed of a hybrid architecture”. Furthermore, the Center for Quantum Networks (CQN) [9] is working directly on key challenges facing the construction of large-scale quantum networks. CQN’s goals include developing foundational technology – such as optical fibers, quantum repeaters, and switches – for metropolitan-scale quantum networks.

But Quantum Internet research is not limited to western countries. China is actively advancing research on quantum communications, with technology implementations such as their quantum secure communication networks (regional “Trunks”) combined with their quantum satellite project (nicknamed *Micius*) [10]. It is estimated that there has been at least a 25 billion dollars Chinese government investment from the mid-1980s until 2022 into quantum technology [11]. One of Beijing’s aims for its 14th five-year plan, which ends in 2025, is to establish an intercity quantum demonstration network based on secure relays.

Meanwhile, companies like IBM, Google, and Amazon are making significant investments in quantum computing and quantum networking. In May 2017, AT&T announced that it was working with the California Institute of Technology to build out its quantum networking technology to offer more secure communications. British Telecommunications (BT), Toshiba Research, ADVA Optical

Networking, and the UK National Physical Laboratory, are collaborating to research and implement quantum encryption.

In June 2018, BT announced that it had built a “quantum-secured” internet network that spanned between Cambridge, UK and BT’s laboratory in Ipswich, a distance of around 50 miles [13]. Amazon has recently established the AWS Center for Quantum Networking [14]. Researchers at the center will work on technologies like quantum repeaters and transducers, to allow for the creation of global quantum networks.

Among the killer applications of the Quantum Internet [15], broad interest has been lately devoted to *distributed quantum computing*, where individual quantum processors – limited in the number of qubits – work together to solve computational tasks exceeding the computational resources available within a single processing device [12, 16–19]. This interest came as no surprise since – differently from classical distributed computing – a linear increase in the number of interconnected quantum processors unlocks an exponential increase of the quantum computational power [12, 17], as summarized in Figure 1.

Unfortunately, the existing literature on distributed quantum computing is spread among different research communities – ranging from the physics through the communications/computer engineering to the computer science community – leading to a fundamental gap. The aim of this survey is precisely to bridge this gap, by introducing the reader to the astonishing and intriguing properties of distributed quantum computing.

Stemming from this, in the following we shed the light on the distinctive characteristics of distributed quantum computing, with the objective of allowing the reader:

- i) to own the implications of the novel, astonishing and intriguing properties of quantum information for understanding the differences between distributed (classical) computing vs. distributed quantum computing;
- ii) to grasp the challenges as well as to appreciate the marvels arising with the paradigmatic shift from monolithic to distributed quantum computing

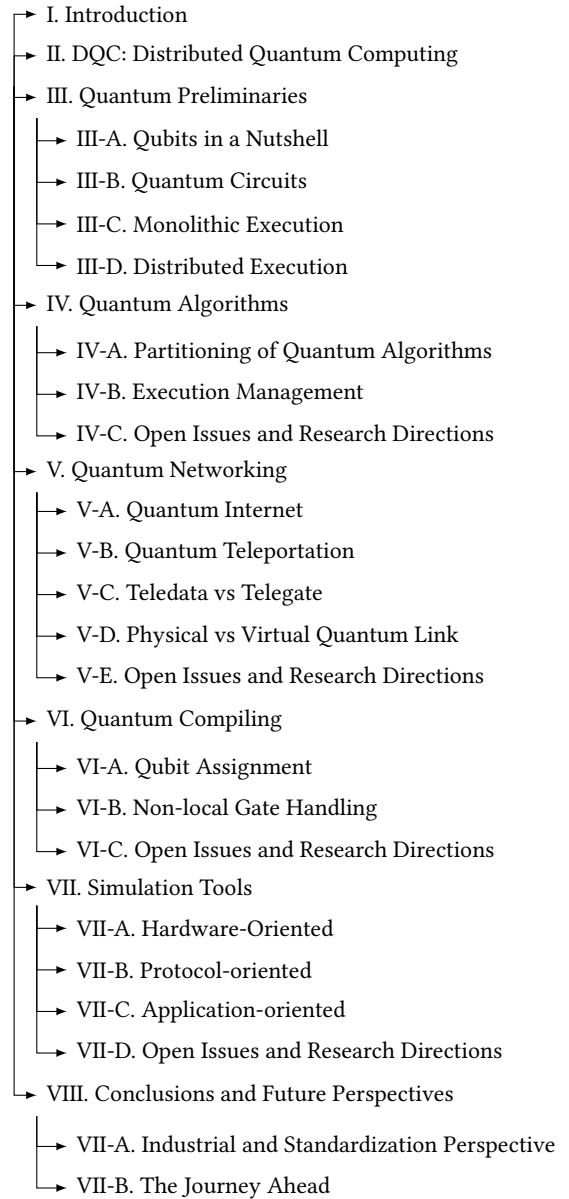


Fig. 2. Structure of the Survey

Indeed, due to the fast grow of this research field, such an understanding will serve the computer science community and the communications/computer engineering community alike to have an easy access and guide towards the relevant literature and to the prominent results, which will be of paramount importance for advancing the state-of-the-art.

To the best of authors' knowledge, a tutorial of this type is the first of its own. The paper is structured as depicted in Figure 2. Specifically, in Section 2, we introduce the rationale for distributed quantum computing, and we present the four different perspectives – algorithms, networking, compiling and simulation – discussed within the survey. Then, in Section 3, we provide the reader with a concise overview about quantum computing, with reference to the quantum circuit model. In Section 4, we focus on quantum algorithms, the extent to which they can be distributed as well as their execution management, once they are executed according to a distributed paradigm. In Section 5, we detail the pivotal role played by quantum networking for enabling distributed quantum computing, by discussing in detail the unconventional features of quantum communications.

In Section 6, we describe some relevant approaches to the problem of compiling quantum algorithms for distributed execution, i.e., splitting them conveniently to fit the available networking and computing hardware. In Section 7, we provide an overview of the most advanced simulation tools for quantum networking, discussing their suitability for the design and analysis of distributed quantum computing architectures. Finally, we conclude our survey in Section 8 by first providing an industrial perspective on distributed quantum computing, and then by discussing the possible stages of distributing quantum computing development.

2 DQC: DISTRIBUTED QUANTUM COMPUTING

Nowadays, all major quantum computing technologies – e.g., ion traps, superconductors, quantum dots, etc. – exhibit hard technological limitations on the number of qubits that can be embedded in a single quantum chip [16]. Accordingly, the consensus of both academic and industry communities for realizing large-scale quantum processors goes toward a quantum computing paradigm-shift, which consists in relying on a quantum network infrastructure to cluster together modular and small quantum chips in order to scale the number of qubits [1, 17, 20, 21].

The aforementioned vision is expected to be realized in a very near future. For instance, IBM plans to introduce in 2025 *Kookaburra* – a 1386 qubit multi-chip processor with communication link support for quantum parallelization – with three Kookaburra chips inter-connected into a 4158-qubit system [22]. With such modular systems, an ordinarily monolithic quantum computation can be “split into pieces” and executed on multiple inter-connected processors by following the distributed quantum computing (DQC) paradigm [12, 23]. Despite being expected further into the future, metropolitan-area and wide-area quantum networks are also under research and development [24–26], which would enable DQC among geographically-distributed quantum device.

This survey focuses on DQC, and it analyzes the state-of-the-art and challenges arising by looking at the DQC according to four main different perspectives, namely: *algorithms*, *networking*, *compiling*, and *simulation*.

As illustrated in Figure 3, for each of these four pillars, the most relevant aspects are discussed. Regarding algorithms, the focus is on the crucial and specific challenges arising when moving from monolithic to distributed quantum computing, namely, quantum algorithm partitioning and execution management. Being DQC an application of quantum networking – namely, being some sort of (quantum) network a fundamental pre-requisite for any form of distributed (quantum) computing – through the survey we will shed the light on the challenges arising with inter-networking different quantum processors, by introducing the reader to the fundamental differences between interconnecting remote *classical* processors versus interconnecting remote *quantum* processors. As regards to compiling, it deals with translating a hardware-agnostic description of

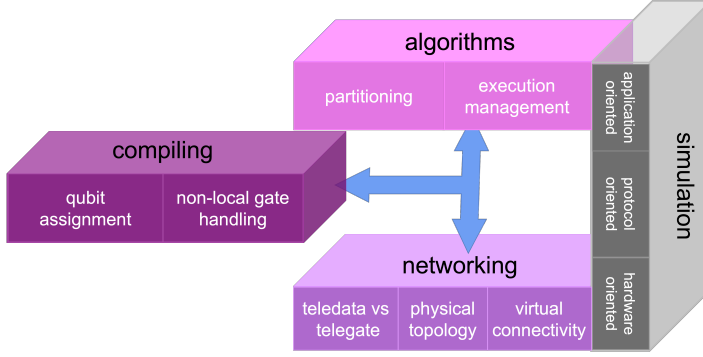


Fig. 3. Distributed Quantum Computing. The four different perspectives overviewed within the survey, with *algorithms*, *compiling* and *networking* represented as interdependent layers of a distributed quantum computing architecture, and *simulation* represented as an inter-layer enabler, covering layers exhibiting quantumness, namely, *algorithms* and *networking*.

the algorithm – namely, the quantum circuit – into a functionally equivalent description that takes into account the physical constraints of the underlying computing architecture [18]. Indeed, within the context of DQC, the compiling must account also for the network constraints, which impact on the strategy adopted for splitting the circuit into “portions” to be concurrently executed on the individual quantum processing units (QPUs)¹. As a matter of fact, a key goal is to minimize the number of remote operations, i.e., operations involving different QPUs. Last but not least, the design of DQC architectures can be highly facilitated by adequate simulation tools, as discussed and detailed in the manuscript.

3 QUANTUM PRELIMINARIES

In this section, we provide a short overview about quantum computing, with reference to the quantum circuit model. We refer the reader to [27] for a concise introduction to the peculiarities and the challenges arising with quantum information, whereas an overview about quantum computing and an in-depth treatise about quantum information and quantum computation are provided in [28, 29] and [30], respectively.

3.1 Qubits in a Nutshell

Information, either classical or quantum, can be encoded in the state of the simplest quantum mechanical system, namely, the quantum bit (qubit). Examples of two-level quantum systems are the spin of the electron, the polarization of a photon, or an atom with a ground state and an excited state.

Mathematically, the state of a qubit is defined as a vector in a two-dimensional complex Hilbert space. Hence qubit states can be treated as mathematical objects, thus enabling the construction of a general theory of quantum computing, which does not rely on the particulars of the underlying technology. By adopting the *bra-ket* notation², the state of an arbitrary qubit can be expressed as a

¹Throughout the manuscript, the two terms quantum processor and quantum process unit are used as synonyms.

²The bra-ket notation, also known as Dirac’s notation [31], is a standard notation for describing quantum states. In a nutshell, a *ket* $|\cdot\rangle$ represents a column vector, whereas a *bra* $\langle\cdot| \doteq |\cdot\rangle^\dagger$ represents the conjugate transpose of the corresponding ket.

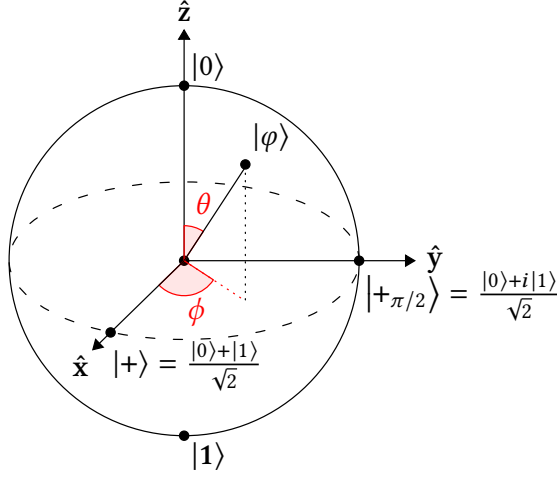


Fig. 4. Bloch sphere: geometrical representation of a qubit in spherical coordinates. A pure state $|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle$ is represented by a point on the sphere surface, with $\alpha = \cos \frac{\theta}{2}$ and $\beta = e^{i\phi} \sin \frac{\theta}{2}$.

linear combination – namely, as a *superposition* – of two basis states $|0\rangle$ and $|1\rangle$:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle. \quad (1)$$

with $\alpha, \beta \in \mathbb{C} : |\alpha|^2 + |\beta|^2 = 1$ (normalization condition). The state of a single-qubit system is often represented geometrically in spherical coordinates by the *Bloch sphere*, illustrated in Figure 4. Specifically, a pure state is represented by a point on the sphere's surface, with θ and ϕ denoting the spherical coordinates:

$$|\varphi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle. \quad (2)$$

ϕ is known as relative phase of the quantum state and it is crucial in creating the interference patterns exploited for instance by quantum algorithms.

The above can be generalized for a composite system of n -qubits, to which an Hilbert space of dimension 2^n can be associated, since the vector spaces associated with the constituent quantum systems are combined through the tensor product. The state of an n -qubit system can be in a superposition of all the 2^n basis states:

$$|\phi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle, \quad (3)$$

with $\alpha_k \in \mathbb{C} : \sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1$. It must be noted, though, that the vast majority of n -qubit states cannot be written as the tensor product of n single-qubit states. These states are referred to as *entangled states* and they represent the key ingredient in quantum computing [32, 33]. As an example, a popular two-qubit entangled state, referred to as *Bell state* or *EPR pair*³, is given by:

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}. \quad (4)$$

³With Bell states named in honor of Bell [34], and EPR pairs named in honor of Einstein, Podolsky and Rosen [35].

Gate Name	Gate Matrix	Operation
Identity	$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	does not modify the quantum state
Pauli-X	$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	bit-flip
Pauli-Y	$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	combined bit- and phase-flip
Pauli-Z	$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	phase-flip
Square-root-of-X	$SX = \frac{1}{\sqrt{2}} \begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}$	SX such that $X = SX \circ SX$
Hadamard	$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	maps an element of the computational basis – either $ 0\rangle$ or $ 1\rangle$ – into an even superposition of the basis elements (and vice versa)
S	$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	$\pi/2$ phase shift
T	$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$	$\pi/4$ phase shift
Phase shift	$P_\theta = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$	θ phase shift
x-Rotation	$R_X(\theta) = e^{-i\frac{\theta}{2}X} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} X$	rotation by θ along \hat{x} -axis of the Bloch sphere
y-Rotation	$R_Y(\theta) = e^{-i\frac{\theta}{2}Y} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} Y$	rotation by θ along \hat{y} -axis of the Bloch sphere
z-Rotation	$R_Z(\theta) = e^{-i\frac{\theta}{2}Z} = \cos \frac{\theta}{2} I - i \sin \frac{\theta}{2} Z$	rotation by θ along \hat{z} -axis of the Bloch sphere
Controlled-NOT	$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	bit-flips the second (target) qubit whenever the first (control) qubit is $ 1\rangle$

Table 1. Common quantum gates.

3.2 Quantum Circuits

The *quantum circuit* [30] is the most popular model of quantum computation, where quantum operators are described as quantum gates. More into details, by sequentially interconnecting different quantum gates, a quantum circuit models the processing of quantum information corresponding to a specific *quantum algorithm* is obtained [18]. Indeed, there exist several equivalent quantum circuits modeling the same computation with a different arrangement or different ordering of gates.

A very simple example of quantum circuit is provided in Figure 5, where each horizontal line represents the time evolution of the state of a single (logical) qubit, with time flowing from left to right, dictating the order of execution of the different gates.

Quantum gates (and, overall, quantum circuits) are described by *unitary matrices* relative to some basis, i.e., matrix U such that $U^\dagger U = I$. It follows that quantum computation is *reversible*: it is always possible to invert a quantum computation.

Some widely-used gates⁴ are reported in Table 1 and, as a matter of fact, every unitary operator U on a single qubit can be formulated as:

$$U = e^{i\theta_1} R_X(\theta_2) R_Y(\theta_3) R_Z(\theta_4), \quad \theta_i \in \mathbb{R} \quad (5)$$

⁴It is worth noting that most quantum gates are self-inverse (like Hadamard and Pauli gates) or determining the inverse is straightforward (e.g., by taking the negated rotation angle for rotation gates).

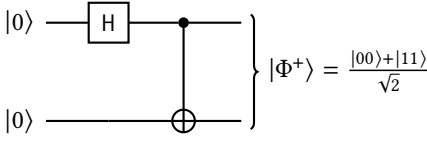


Fig. 5. Quantum circuit for generating a two-qubit entangled state – namely, the Bell state in (4) – starting from the input state $|00\rangle$. Time flows from left to right: the first qubit undergoes through a single-qubit Hadamard gate – denoted with H – followed by a two-qubit CNOT gate – represented by \bullet and \oplus symbols interconnected by a vertical line – both defined in Table 1.

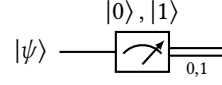


Fig. 6. Quantum circuit for measuring a qubit. Single wires denote quantum states, whereas double wires denote classical states, namely, bits. The measurement of a qubit – whose output is a classical bit – induces the state of the qubit to collapse into the measured state.

with R_i denoting the i -axis rotation operator, defined in Table 1. More precisely, the possibility of implementing two arbitrary rotation operators is sufficient, as their combined application can be exploited to obtain the third type of rotation in (5).

Among two-qubit gates, highly relevant are the *controlled* ones. The generic Controlled-U gate operates on two⁵ qubits, namely a *control qubit* (controlling the operation) and a *target qubit* (subjected to the operation). By denoting with $|\varphi_c\rangle$ and $|\varphi_t\rangle$ the control and target qubits respectively, the effect of the controlled U gate on the target qubit is the following:

$$\begin{cases} I |\varphi_t\rangle & \text{if } |\varphi_c\rangle = |0\rangle \\ U |\varphi_t\rangle & \text{if } |\varphi_c\rangle = |1\rangle. \end{cases} \quad (6)$$

The most famous example of controlled gate is represented by the CNOT gate, which is a Controlled-U gate where the U gate is a Pauli-X one. The CNOT gate can be used to create or destroy entanglement among the inputs. Specifically, to obtain an entangled state, we may start from the separable input $|00\rangle$ and, by applying H to the first qubit, obtaining $(|00\rangle + |10\rangle)/\sqrt{2}$. Finally, by applying a CNOT gate (where the first qubit is the control one, as shown in Figure 5), the resulting state is exactly the Bell state given in (4), namely, $(|00\rangle + |11\rangle)/\sqrt{2}$.

Gates H, S and CNOT constitute the *Clifford group* [36], which can be simulated efficiently on a classical computer according to the Gottesman-Knill theorem [30]. The Clifford group is not universal, i.e., it cannot be used to describe any arbitrary quantum algorithm. However, it is sufficient to add the T gate to the Clifford group, and the resulting set is universal, yet it is not the only possible one. Indeed, each family of quantum computers – e.g., IBM Q one [37] – has its own specific universal gate set, which usually depends on the particulars of the underlying quantum hardware technology.

It is worthwhile to note that, regardless the particulars of the adopted gate set, deterministic cloning of quantum states is impossible. Specifically, there exists no quantum gate (or circuit) able to make a perfect copy of an arbitrary unknown quantum state. Conversely, if the state is known in advance – specifically, if we know that the state belongs to some orthonormal basis such as $\{|0\rangle, |1\rangle\}$ or $\{|+\rangle, |-\rangle\}$ – we can design a specific quantum gate to clone that state. This fundamental property is known as *no-cloning theorem*, and it has deep impact on distributed quantum computing as we will discuss in Section 5.

Another unconventional quantum phenomenon arises with the important operation constituted by *measurement*, through which information from a quantum state is extracted [38], as illustrated

⁵Controlled operations can be also defined for multi-qubit targets.

in Figure 6. In fact, according to the quantum measurement postulate, although a qubit may reside in a superposition of two orthogonal states as in (1), when we want to observe or measure its value, it collapses into one of the two orthogonal states $|0\rangle$ – with probability $|\alpha|^2$ – and $|1\rangle$ – with the probability $|\beta|^2$. After its measurement/observation, the original quantum state collapses to the measured state. Hence, the measurement irreversibly alters the original qubit state [27].

Finally, it is worthwhile to mention that a quantum circuit exhibits three important quantitative features: the *width*, i.e., the number of qubits, the *gate count* and the *depth*, i.e., the longest path in the circuit. Each and all of them affect the overall *quality* of the computation result. To get an easy flavor about the aforementioned statement, it is enough to think that due to imperfect hardware the expected error propagation is upper bounded by $2(1 - (1 - r)^m)$ [41] where $0 \leq r < 1$ is a constant independent of the qubit number and m is the number of gates in the circuit.

3.3 Monolithic Execution: Gate Synthesis and Circuit Compilation

As mentioned in the previous subsection, even if there exists an uncountable number of quantum logic gates, the set of gates that can be executed on a certain quantum processor is limited, as a consequence of the constraints imposed by the underlying qubit technology [42]. In this case, any gate outside this *reduced set* must be obtained with a proper combination of the allowed gates through a process known as *gate synthesis*. As an example, IBM quantum processors are realized exploiting the superconducting technology, and any logical gate that can be run on current IBM quantum processors is built from a gate set composed by the controlled-not (CNOT) gate and four single-qubit gates (namely, I, R_Z , SX, X gates).

Furthermore, regardless of the underlying qubit technology, the abstract qubits subjected to quantum gates as specified by the quantum circuit, known as *logical qubits*, should not be confused with the physical qubits embedded within a quantum processor [18]. With reference to the physical qubits, any quantum processor exhibits hardware constraints affecting the allowed interactions between them. As an example, CNOT gates cannot be applied to any physical qubit pair of an IBM quantum processor, but they are instead restricted⁷ to certain pairs. The allowed pairs are usually represented with the *coupling map* – namely, with a graph where vertices denote qubits and arrows denote the possibility of realizing a two-qubit CNOT gate between the connected qubits – as illustrated in Figure 7. From the above, it becomes clear that the monolithic execution of a quantum algorithm on a single quantum processor requires a circuit pre-processing known as *quantum compiling* [18, 43–45]. In a nutshell, compiling a quantum circuit is a two-step⁸ process where:

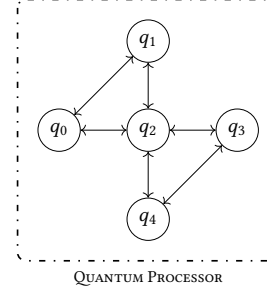


Fig. 7. Coupling maps of a IBM Yorktown quantum processor [39, 40]. The five physical qubits stored within the processor are represented by circles. The arrows denote the possibility to realize a two-qubit CNOT gate between the five qubits. As an example, a CNOT between qubits q_0 and q_1 can be directly executed by the quantum processor, whereas a CNOT between qubits q_0 and q_2 cannot.

⁶The measurement of a qubit state may also be carried out in a basis different from that in which the qubit was prepared in [28–30]. In the above description, for the sake of clarity, we assumed the standard basis also for the measurement.

⁷These limitations arise as a consequence of both the: i) noise effects induced by qubit-interactions, and ii) physical-space constraints within a single processor [18].

⁸With the two steps being inter-dependent, affecting each others.

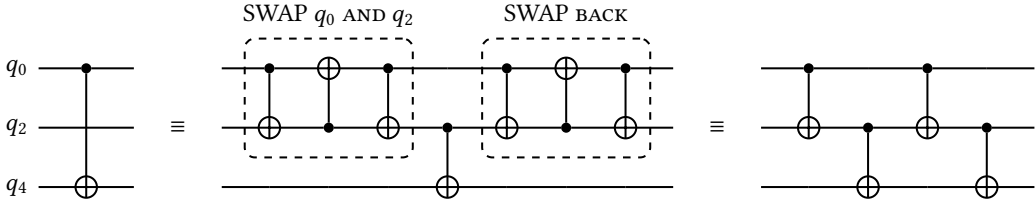


Fig. 8. Example of equivalent quantum circuits generated during quantum compiling for mapping an arbitrary CNOT into a sequence of CNOTs that can be directly executed by a given quantum processor. A CNOT between qubits q_0 (control) and q_4 (target) with the coupling map given in Figure 7 can be obtained through either: i) *quantum state transfer*, by first swapping qubits q_0 and q_2 , then by performing a CNOT between q_2 and q_4 , and finally by swapping again qubits q_0 and q_2 so that they recover their initial position, or ii) *ancilla qubit*, by performing four CNOT operations between neighbour qubits with the help of the intermediate qubit q_2 . Figure reproduced from [18].

- i) each logical qubit of the quantum circuit must be mapped onto one (or more, when adopting fault-tolerant techniques [46]) physical qubit of the quantum processor, and
- ii) each two-qubit gate – as instance, a CNOT – between physical qubits non-adjacent within the coupling map must be mapped into a computational-equivalent sequence of gates between adjacent physical qubits, as exemplified in Figure 8.

Clearly, the overall process must be optimized to account for the key performance metrics affecting quantum computation [47–49]. Typically, this consists in minimizing the depth of the *compiled circuit*, namely, the equivalent quantum circuit satisfying all the constraints imposed by the quantum processor coupling map.

3.4 Distributed Execution

So far, we focused on quantum circuits by abstracting from the particulars of the underlying computing technology. Indeed, we mentioned that the natively-available gates depends on the underlying hardware. Yet, this is not an issue since – as long as the available gate set is universal – any arbitrary gate can be implemented with a finite sequence of the available gates up to arbitrary accuracy [29]. From the above, we can safely assume that any quantum circuit can be directly executed on a given quantum processor, either in the original form or by properly replacing unavailable gates with sequences of available ones. But the task becomes significantly harder when we move from monolithic quantum computing to distributed quantum computing. Specifically, any universal gate set must include one multi-qubit gate – typically a 2-qubit gate such as the CNOT – and any quantum circuit of some interest includes multi-qubit gates as well. As a consequence, as illustrated with the toy-model in Figure 9, a distributed quantum computation involves operations between qubit pairs across different end-nodes (i.e., non-local gates). At a first sight, this might seem not a big deal. Also (classical) distributed computing involves operations between classical information located at different processors. And these operations are performed by simply moving the information from one processor to another. So one might be tempted to believe that the same strategy can be adopted when it comes to distributed quantum computing. Unfortunately, this is not true: quantum information requires a paradigm shift for dealing with inter-processor communications, and the rationale for this would be deeply discussed in the next sections.

4 QUANTUM ALGORITHMS

There exists several quantum algorithms known or expected to outperform classical algorithms for problems spanning different areas, including cryptography, search and optimization, simulation of quantum systems and learning [50]. Remarkably, most known quantum algorithms use a combination of algorithmic paradigms – namely, sub-routines – specific to quantum computing [37]. These paradigms include the Quantum Fourier Transform (QFT) [51], the Grover Operator (GO) [52], the Harrow/Hassidim/Lloyd (HHL) method for linear systems [53], Variational Quantum Algorithms (VQA) [54], and direct Hamiltonian simulation (SIM). A prominent example is Shor’s algorithm for integer factorization [51], which is based on QFT, illustrated by the quantum circuit in Figure 10.

For most practical applications, quantum algorithms require large quantum computing resources – in terms of qubit number – much larger than those available with current noisy intermediate-scale quantum (NISQ) processors. For example, the recently announced IBM Quantum Osprey device has 433 qubits, which is an impressive progress with respect to state-of-the-art quantum processors, but not yet sufficient, as an example, for running practical implementations of Shor’s algorithm⁹.

Distributed quantum computing is envisioned as a scalable approach for increasing the number of qubits available for computational tasks. However, moving from monolithic to distributed quantum computing implies crucial and specific challenges.

4.1 Partitioning of Quantum Algorithms

A first issue that arises with quantum algorithms is whether a given algorithm – equivalently, a given quantum circuit – is natively suitable for distributed execution. More specifically, a *perfectly distributable* quantum algorithm is a quantum algorithm that can be split into autonomous parts that do not interact – or, at least, weakly interact – with each others. If this is the case, each part can be assigned to some quantum processor, and each processor can contribute autonomously to the overall computation without introducing communication overhead for interacting with other processors.

Unfortunately, this is not the usual case. As an example, let us consider the QFT algorithm, whose circuit is given in Figure 10, notably used as sub-routine in many quantum algorithms – e.g., Shor’s algorithm and the quantum phase estimation algorithm – as mentioned above. From Figure 10, it is easy to assess that QFT requires each qubit to strongly interact with all the other qubits through controlled R_n gates. Hence, QFT can be considered as the archetype of monolithic quantum algorithms, namely, of an algorithm not natively-suitable for distributed execution.

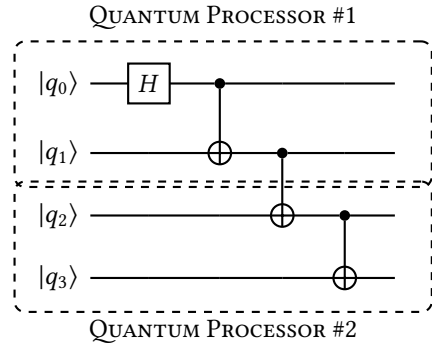


Fig. 9. Toy model for distributed quantum computation. The quantum circuit is composed by three two-qubit gates, i.e. CNOTs. First and last gates operate *locally*, namely, between qubits stored within the same QPU, whereas the intermediate gate operates *remotely*, namely, between qubits stored within different QPUs.

⁹Factoring $L = 2048$ bit primes – for breaking current RSA implementations – requires about $3L = 6144$ noise-free qubits [30]. It is worth noting that merely increasing the number of physical qubits is not sufficient, as some sort of quantum error correction [55] is also required to guarantee high-quality – namely, noise-free – computations.

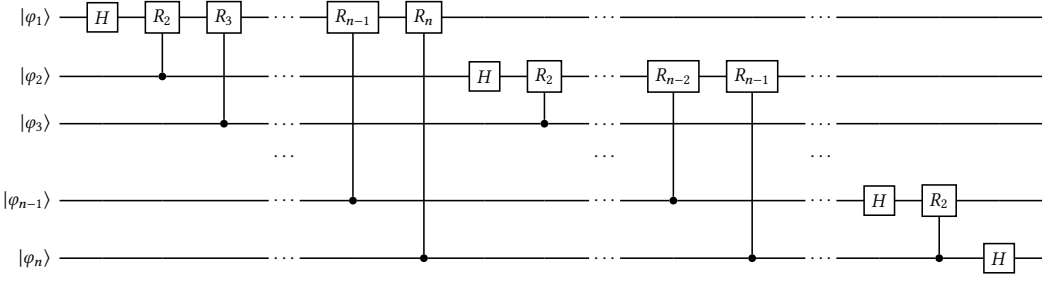


Fig. 10. Quantum Fourier Transform (QFT) circuit. The i -th qubit is obtained through an Hadamard gate followed by $n - i$ controlled R_n operations – with $R_i = P_{2\pi/i}$ denoting the phase gate given in Table 1 – with the controlled operations controlled by the $n - i$ higher-order qubits.

As we anticipated in Section 2, to distribute a monolithic quantum algorithm, a quantum compiler must be used to find the best breakdown, i.e., the one that minimizes the number of gates that are applied to qubits stored at different devices. Quantum compilation is reviewed in Section 6. Here we discuss some literature that addresses the partitioning of relevant quantum algorithms, using techniques that are tailored to the specific considered algorithms rather than general-purpose. These works may represent a good reference for a comparative evaluation of quantum compilers.

In [56], Neumann et al. present two distribution schemes for the *quantum phase estimation* algorithm, give the resource requirements for both and show that using less noisy shared entangled states results in a higher overall fidelity. Introduced by Kitaev [57], the quantum phase estimation algorithm returns an approximation of an eigenvalue of a given unitary U and a corresponding eigenvector. It has numerous applications, including Shor's algorithm [51]. The solution proposed by Neumann et al. is based on the distributed version of the QFT circuit, obtained by means of non-local controlled U -gates¹⁰.

Another example of distributable quantum algorithm is the *Variational Quantum Eigensolver* (VQE), a VQA that can be used to estimate ground state energies of molecular chemical Hamiltonians. In [59], DiAdamo et al. provide a *Local to Distributed Circuit* algorithm that, given a circuit representation as a series of layers and a mapping of qubits, searches for any control gates where the control and target are physically separated between two QPUs. When found, the algorithm inserts, between the current layer and next layer in the circuit, the necessary steps to perform the control gate in a nonlocal way¹¹. The size (maximum number of qubits) of the achievable Ansatz state for the VQE algorithm grows linearly with the number of QPUs, with slope linearly increasing with the number of qubits per QPU. The depth of the resulting quantum circuit is $\Omega(n)$, meaning it has a tight upper and lower bound proportional to the number n of qubits.

In [61], the authors present a distributed adder and a distributed distance-based classification algorithm. Both applications are framed in a way where a quantum server and K other quantum nodes interact, with specific behaviors. In particular, the server is responsible for orchestrating the computation by means of non-local CNOT gates, while the K parties provide inputs. It is possible to reframe these applications, such that the proposed quantum circuits are considered as monolithic and subsequently split in $K + 1$ parts to be submitted for execution to a quantum network.

¹⁰Non-local controlled U -gate generalizes the telegate operation discussed in Section 5.3 to arbitrary unitary U [58].

¹¹By using the *cat-entangling* method by Yimsiriwattana et al. [60], which is substantially equivalent to telegate introduced in Section 5.3.

4.2 Execution Management

Another challenge is related to the execution management of distributed quantum computations. In general, given a collection \mathcal{P} of quantum circuit instances to be executed, this collection should be partitioned into non-overlapping subsets \mathcal{P}_i , such that $\mathcal{P} = \cup_i \mathcal{P}_i$. One after the other, each subset will be assigned to the available QPUs. In other words, for each execution round i , there exists a schedule $S(i)$ that maps some quantum circuit instances to the quantum network. If DQC is supported, some quantum circuit instances may be split into sub-circuit instances, each one to be assigned to a different QPU, as illustrated in Figure 11). A QPU scheduling algorithm that partially address this service was proposed by Parekh et al. [23]. Such an algorithm is based on a greedy approach, trying to fill all available QPUs while minimizing the number of distributed quantum circuit instances. Here the partitioning of quantum circuit instances is arbitrary, not taking into account the features of the programs.

Recalling Section 4.1, we stress that partitioning should be an orthogonal service with respect to QPU scheduling. It is worth noting that the QPU scheduling plane must be clearly separated from the networking plane. We demand that any subset of the available QPUs can be the target of any quantum computation, provided that the total number of physical qubits fits the circuit width. This means that the underlying network should allow to create entangled quantum states across any two QPUs. Technical details on entanglement distribution are presented in Section 5. Here we recall a recent work by Cicconetti et al. [62], which investigates the requirements and objectives of DQC from the perspective of quantum network provisioning. In particular, the authors elaborate on two different classes of traffic, namely constant-rate flows and DQC applications.

4.3 Open Issues and Research Directions

Future directions are both theoretical and practical. Despite a considerable amount of work on the fundamentals of distributed quantum computing [63–65], an ultimate theory of distributable quantum algorithms is still missing. It is known that the quantum circuit model and the DQC model are equivalent up to polylogarithmic depth overhead [64], but a general framework for ranking quantum algorithms in terms of distributability has not been defined. To this purpose, it is necessary to provide a quantitative definition of quantum circuit distributability. Regarding execution management, the broad literature on job scheduling for high performance computing may be a starting point, but it is clear that the peculiarities of quantum computing – quantum parallelism, no-cloning, entanglement, etc. – demand for novel and specific strategies for the efficient execution of concurrent distributed quantum computations. A trade-off between the complexity of the distributed quantum circuit and the physical distance between quantum processors is also envisaged.

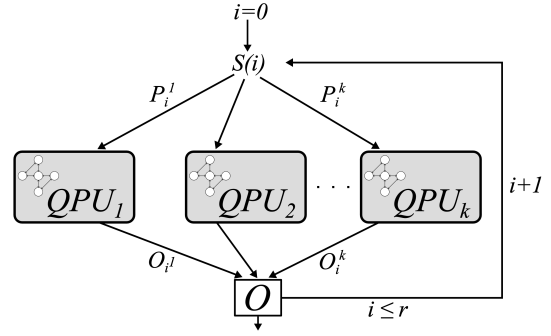


Fig. 11. Execution of multiple quantum circuit instances with k QPUs. For each execution round i , a schedule $S(i)$ maps some quantum circuit instances to the quantum network – each QPU receiving a quantum circuit P_i^j that is either a monolithic one or a sub-circuit of a monolithic one. The classical outputs are accumulated into an output vector O .

To compare different deployments and schedules, DQC-specific key performance indicators must be defined. Recently, two frameworks with similar names have been proposed almost at the same time, namely Quantum Network Utility Maximization (QNUM) [66] and Quantum Network Utility (U_{QN}) [67]. While QNUM is specifically tailored to the evaluation of entanglement routing schemes in quantum networks (see Section 5 for details about entanglement), U_{QN} is more abstract, aiming to capture the social and economic value of quantum networks, for a variety of applications (from secure communications to distributed sensing). Incidentally, in [67] the example of DQC is studied in detail, through the lens of U_{QN} . More specifically, a quantum network utility metric is presented, which applies the Quantum Volume¹² proposed in [68] to the U_{QN} framework. Such a metric quantifies the value derived from performing QC tasks, and it is viewed as a “quantum volume throughput”. It differs from the quantum volume in two ways: i) it explicitly considers the rate at which non-local operations can be performed, and ii) it accounts for the utility derived simultaneously from tasks executed on different parts of the network.

5 QUANTUM NETWORKING

As mentioned in the previous sections, when it comes to distributed quantum computing, qubits are distributed among multiple smaller quantum processors, interconnected by some sort of quantum network.

Accordingly, whenever a quantum gate must operate on *remote qubits* – namely, qubits located in different quantum processors – some sort of *communication primitive* must be available for performing inter-processor operations. Unfortunately, this communication primitive cannot be easily accomplished through classical protocols. Indeed, the different physical phenomena underlying quantum communications impose a paradigm shift.

To better understand the above statement, in the following we shed the light on the challenges arising with networking different quantum processors. To this aim, in the following we first substantiate in Section 5.1 the fundamental differences arising with interconnecting remote *classical* processors versus interconnecting remote *quantum* processors. Then, in Section 5.2 we introduce the marvels of quantum teleportation, which represents the underlying communication functionality enabling remote quantum operations. Stemming from this, in Section 5.3 we discuss the two

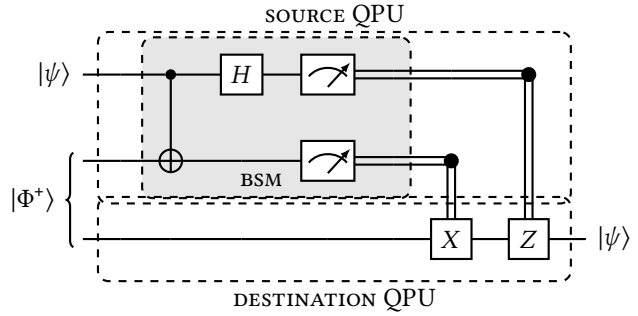


Fig. 12. Pictorial representation of the quantum teleportation circuit. The first two wires belong to the source node, whereas the bottom wire belongs to the destination node. A generic state $|\psi\rangle$ is initially stored at the source, and a Bell state such as $|\Phi^+\rangle$ given in (4) must be distributed through a quantum link so that one entangled member is stored at the source and the other at the destination. Once the Bell state is available, the teleportation is obtained with some processing of $|\psi\rangle$ and the entangled member at the source, followed by two conditional gates on the entangled pair at the destination, depending on the measurement of the two qubits at the source. Each double line denotes the transmission of one classical bit – i.e., the measurement output – between the remote processors. The two classical bits are thus used as detailed in Table 2 for determining whether the two conditional gates X and Z must be applied to recover the original state $|\psi\rangle$ from the entangled member available at the destination.

¹²Quantum Volume (QV) is a single-number metric that can be measured using a concrete protocol on near-term quantum computers of modest size. The QV method quantifies the largest random circuit of equal width and depth that the quantum processor successfully executes.

possible strategies – namely, *telegate* and *teledata* – for implementing quantum gates between remote qubits. Then, in Section 5.4 we present a key strategy – referred to as *entanglement swapping* – for virtually-augmenting the connectivity among different quantum processors. And finally, in Section 5.5, we discuss the open problems arising with interconnecting remote quantum processors.

5.1 Quantum Internet

According to the on-going IETF RFC draft on Quantum Internet architectures [69], the Quantum Internet can be defined as an interconnection of heterogeneous¹³ quantum networks, able to exchange qubits and to generate and share entangled states among themselves. Hence, the Quantum Internet services ground on the manipulation and transmission of qubits as well as on the distribution of entangled states. This, in turn, imposes several challenges with no-counterpart in classical network and that cannot be solved through existing classical protocols.

As an example, Internet (and classical networks in general) extensively relies on the possibility of freely duplicating information. But this basic assumption does not hold when it comes to the Quantum Internet [70, 71] accordingly to the *no-cloning* theorem¹⁴. Furthermore, according to the *measurement postulate*, even the simple action of measuring a qubit – i.e., reading the quantum information stored within – irreversibly alters its quantum properties, such as superposition and entanglement.

The above peculiarities of quantum mechanics have deep implications on the design of quantum communication techniques for the Quantum Internet [1]. To further elaborate on the above statement, let us clarify that it is possible to map a qubit into a photon degree of freedom by directly transmitting this qubit to a remote node via a fiber link or free space. However, if the traveling photon is lost due to attenuation or it is corrupted by *decoherence*¹⁵, the associated quantum information cannot be recovered via a measuring process or by re-transmitting a copy of the original information. As a consequence, the techniques mitigating the imperfections imposed on the qubits cannot be directly borrowed from classical communications [27].

Thankfully, quantum entanglement [73] can be exploited as a communication resource to face with the aforementioned challenges. Indeed, entanglement enables a communication technique, known as *quantum teleportation*, for transmitting an unknown qubit without the physical transfer of the particle storing the qubit, as described in the following.

5.2 Quantum Teleportation

As introduced in Section 3, whenever two qubits are entangled, they exist in a shared state, such that any action on a qubit affects instantaneously the other qubit as well, regardless of the distance [70].

This unconventional correlation is exploited by the so-called *quantum teleportation process* [27], which enables the possibility of “transmitting” – namely, *teleporting* – an unknown qubit without the physical transfer of the particle storing the qubit.

¹³With heterogeneity arising, as instance, since different networks may be owned by different organizations and/or based on different quantum hardware technologies.

¹⁴Distributed quantum computing represents the perfect archetype for scenarios where an unknown qubit must be transmitted. Indeed, let us consider distributed computing between two remote quantum processors, where one processor executes a quantum algorithm – more precisely, some portion or task of a quantum algorithm – and the other processor must further process the result of the task. The partial result of the processing at the first node is indeed an unknown quantum state – otherwise, if known in advance, the whole processing would have been pointless – and any attempt to measure it would irreversibly alter the partial result, hence wasting the computation performed so far.

¹⁵Any quantum system inevitably interacts with the environment and it is afflicted by decoherence, a phenomenon that irreversibly scrambles the quantum state and therefore its inner information [72]. This kind of quantum noise affects every quantum operation, from qubit processing through qubit storing to qubit transmission, and it causes an irreversible loss of the quantum information as time passes.

More into details, quantum teleportation requires:

- i) an EPR pair, namely a pair of maximally entangled qubits such as the Bell state in (4), with one qubit of the pair distributed at the source node and the other qubit distributed at the destination;
- ii) local quantum operations both at the source and at the destination;
- iii) the transmission of two classical bits from the source to the destination.

The circuital representation of the quantum teleportation process is illustrated in

Figure 12. More into details, the source performs a pre-processing, namely, a *Bell State Measurement* (BSM) on both the unknown qubit encoding the information, say $|\psi\rangle$, to be transmitted and the entangled qubit. As represented in the gray box in the figure, the BSM consists of a CNOT gate – with the information qubit acting as control and the entangled qubit acting as target – followed by an Hadamard gate on the information qubit and, finally, a measurement of both the qubits. Then, the source transmits – though classical communications – two classical bits encoding the measurement outcomes of the BSM. Remarkably, after the BSM, the source quantum state has been already teleported at the destination. Nevertheless, the teleported state may have been undergone a phase and/or a bit-flip, with each flip event occurring individually with a probability equal to 0.25. Luckily, the measurement of the two qubits at the source allows the destination – once the measurement outcomes have been received through a classical communication channel – to determine whether these flip events occurred. Hence, the destination performs a post-processing to reconstruct the original state $|\psi\rangle$, as detailed in Table 2.

In conclusion, by pre-sharing a maximally-entangled pair of qubits¹⁶, two nodes can reliably exchange quantum information through the teleportation process [74], which represents the underlying communication functionality enabling remote quantum operations, as further elaborated in the following subsection.

5.3 Teledata vs Telegate

In distributed quantum computing, quantum teleportation constitutes the fundamental communication primitive underlying the communication paradigms known as TeleData and TeleGate [75], which generalize the concept of moving quantum states among remote devices.

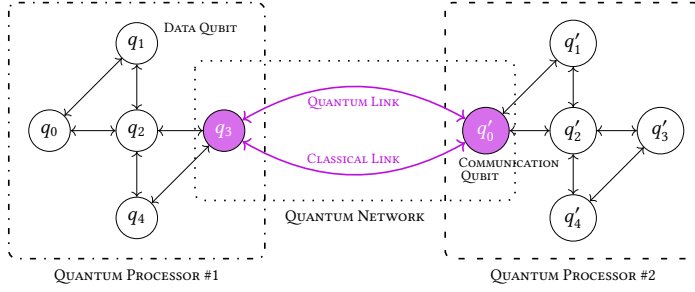
To provide concrete examples of the TeleData and TeleGate concepts, we must classify qubits within a QPU either as *communication qubits* or as *data qubits* [12]. Specifically, within each quantum processor, a subset of qubits is reserved for inter-processor communications and we refer to these qubits as communication qubits [69], to distinguish them from the remaining qubits within the device devoted to processing/storage, which we refer as data qubits.

More into detail, entanglement distribution among network nodes requires that at least one qubit at each processor, referred to as communication qubit, must be reserved for the generation of the entangled state [69]. Clearly, the more communication qubits are available within a network

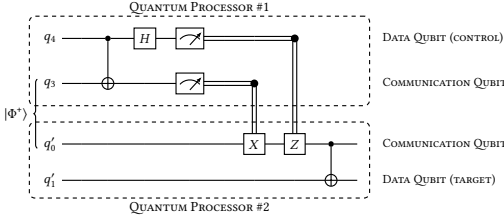
Measurement Output	Decoding operation
00	I
01	X
10	Z
11	X followed by Z

Table 2. Quantum teleportation: post processing operations to be performed at the destination for recovering the original quantum state. Measurement output aligned with right-most digit representing the outcome of the entangled qubit measure in Figure 12, and gates X and Z – corresponding to a bit- and a phase-flip, respectively – detailed in Table 1.

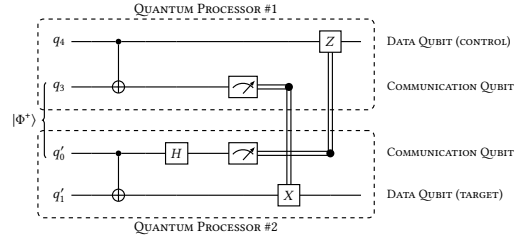
¹⁶We may observe that direct transmission of qubits is still needed to distribute entangled states among the network nodes. However and as deeply clarified in [70], differently from unknown qubits, entangled states can be repeatedly prepared for facing with losses and/or noise corruptions.



(a) Two IBM Yorktown quantum processors given in Figure 7 interconnected through a quantum network, composed by a classical and a quantum link. The classical link is used to transmit classical information, whereas the quantum link is needed for distributing entangled states between the two remote processors to enable communication functionalities. Indeed, at least one physical qubit at each processor must be reserved for entanglement generation. This kind of qubits – dark-blue-colored in the figure – are the *communication qubits* to distinguish them from the *data qubits* – white-colored in the figure.



(b) TeleData. To perform a TeleData between remote processors – say to move the quantum state $|\varphi\rangle$ stored by data qubit q_4 in Figure 13a to communication qubit q'_0 – a Bell state such as $|\Phi^+\rangle$ must be distributed through the quantum link so that each pair member is stored within the *communication qubit* at each processor. Once $|\varphi\rangle$ is teleported at q'_0 (with local quantum operations and classical transmission), the remote operation – for instance, a CNOT with $|\varphi\rangle$ as control and the state stored by qubit q'_1 as target as shown with the last CNOT in the figure – can be executed through local operations.



(c) TeleGate. A TeleGate enables a direct gate between remote physical qubits stored at different processors without the need of quantum state teleportation, as long as a Bell state such as $|\Phi^+\rangle$ is distributed through the quantum link. As instance, a remote CNOT between q_4 and q'_1 in Figure 13a can be implemented with two local CNOTs between the *data* and the *communication qubit* at each processor, followed by a conditional gate on the data qubit depending on the measurement of the remote communication qubit.

Fig. 13. Remote operations through either TeleData or TeleGate. Figure 13a shows the network topology along with the processors coupling maps, whereas Figures 13b and 13c illustrate the quantum circuit detailing the classical (2 bits) and the quantum (the Bell state) resources needed to execute a TeleData and a TeleGate, respectively. Figure reproduced from [18].

node, the more entanglement resource is available at that node, with an obvious positive effect on entanglement rate achievable by that node [70]. But the more communication qubits are available, the less data qubits are available for quantum computing.

As an example, consider two quantum processors interconnected via a quantum network as depicted in Figure 13. Qubits q_3 and q'_0 are communication qubits and any interaction between

the two remote processors is carried out by exploiting them via either a TeleData or a TeleGate process.

With a TeleData, quantum information stored within a data qubit at the first processor, say $|\varphi\rangle$ in q_4 in Figure 13a, is teleported into a communication qubit of the second processor, say q'_0 in the same figure. Once the quantum state $|\varphi\rangle$ is teleported in q'_0 , any remote operation – originally involving q_4 and some data qubits at the second processor – can be now implemented through local operations as shown with the last CNOT in Figure 13b. It must be noted, though, that whether the teleported quantum state should subsequently interact with data qubits at the first processor, a new teleportation process must be performed for teleporting the quantum state back to the first processor. TeleData is not the only available option for implementing remote operations. In fact, a TeleGate enables to execute a direct gate between qubits belonging to remote processors by exploiting again entanglement. As instance, a remote CNOT with data qubit q_4 and q_0^1 in Figure 13a acting as control and target, respectively, can be implemented with local CNOTs at each quantum processor, as shown with the quantum circuit in Figure 13c.

5.4 Physical vs Virtual Quantum Links

From Figure 13, one might assume that distributed quantum computing requires a fully-connected network topology – namely, that each quantum processor must be directly inter-connected with all the other processors – as a consequence of the unconventional characteristics of quantum information. In other words, one might assume the connectivity between quantum processors strongly dependent on the availability of a direct entanglement generation and distribution architecture. As a matter of fact, the very opposite is true. Distributed quantum computing can exploit a strategy – called *entanglement swapping* [42] and summarized in Fig 14 – to implement a remote CNOT between qubits stored at remote processors, even if the processors are not directly connected through a quantum link.

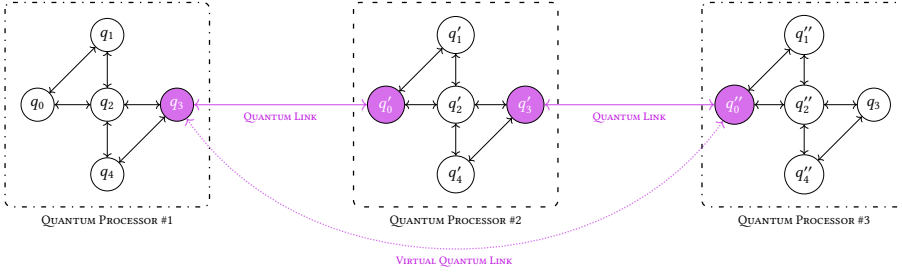
In a nutshell, to distribute a Bell state between remote processors – say quantum processor #1 and #3 in Figure 14a – two Bell states must be first distributed through the quantum links so that one Bell state is shared between the first processor and an intermediate node and another Bell state is shared by the same intermediate node and the second processor. Then, by performing a BSM on the communication qubits at the intermediate node – i.e., qubits q'_0 and q'_3 in Figure 14b – a Bell state is obtained at the remote communication qubits q_3 and q''_0 in Figure 14b – by applying some local processing at the remote nodes depending on the (classical) output of the Bell state measurement.

From the above, it becomes clear that entanglement swapping significantly increases the connectivity within the virtual quantum processor. And the higher is the number of available quantum processors, the higher is the number of possible interactions. Indeed, the number of additional interactions via entanglement swapping scales linearly with the number of available processors when only two communication qubits are available at each intermediate processor. If this constraint is relaxed, the number of additional interactions via entanglement swapping scales more than linearly.

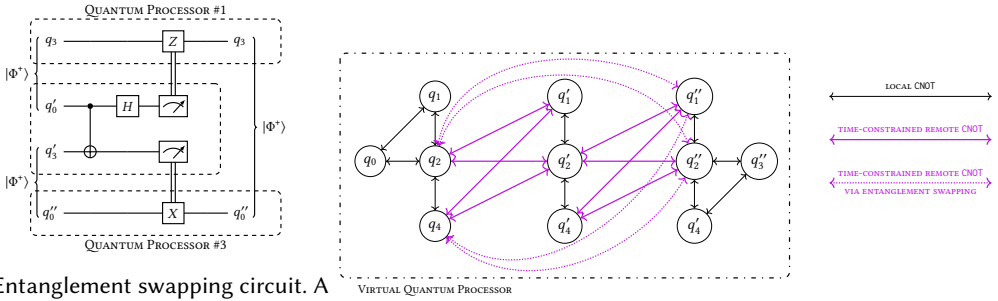
5.5 Open Issues and Research Directions

Stemming from the discussion carried out in the previous subsections, here we summarize some fundamental open issues and research directions towards the interconnection of different quantum processors for enabling distributed quantum computing.

First, in Section 5.3 we introduced the two possible strategies – TeleGate and TeleData – for implementing quantum gates between remote qubits. From a communication resource perspective,



(a) By swapping the entanglement at an intermediate node – namely, quantum processor #2 – it is possible to distribute a Bell state between remote processors – namely, processors #1 and #3 – even if they are not directly connected through a quantum link. Hence, entanglement swapping enhances the quantum processors connectivity through *virtual quantum links*.



(b) Entanglement swapping circuit. A Bell state can be distributed between remote processors by swapping the entanglement at an intermediate node – as instance, processors #2 in Figure 14a – through local processing and classical communication.

(c) Dynamic coupling map for the network topology shown in Figure 14a. The solid blue lines denote remote CNOTs between adjacent processors, whereas the dotted blue lines denote remote CNOTs between distant processors achievable via entanglement swapping.

Fig. 14. Augmented connectivity. Entanglement swapping increases the connectivity between physical qubits, with a number of possible remote CNOTs that scales at least linearly with the number of processors. Figure reproduced from [18].

TeleData and TeleGate consume the same amount of quantum and classical resources, namely one EPR pair and the transmission of two classical bits. Yet the overall performance of the two strategies depends on a range of factors, including i) the *pattern* of remote operations exhibited by the quantum circuit to be executed, ii) the characteristics of the network interconnecting the remote quantum processors, and iii) the ratio between data and communication qubits [70, 75].

With reference to the latter factor, a fundamental trade-off arises [18]. Specifically, each remote operation – regardless whether it is implemented with a TeleData or a TeleGate – consumes the entangled resource. Consequently, a new Bell state must be distributed between the remote processors before another remote operation could be executed. Hence, the more communication qubits are available within each processor, the more remote operation can be executed in parallel, reducing the communication overhead induced by the distributed computation. But the more communication qubits, the less data qubits are available for computing in each processor.

Accordingly to the above reasoning, the selection of the set of communication qubits is a crucial task for distributed quantum computing, with profound effects on the overall performance of the distributed computation. As a matter of fact, the fundamental role played by communication qubits is further stressed by the augmented connectivity enabled by entanglement swapping, discussed in Section 5.4. Indeed, it must be acknowledged that such an augmented connectivity does not come for free. Entanglement swapping consumes the Bell states at each intermediate processor. And the longer is the path between the two processors involved in the remote operation, the higher is the number of consumed Bell states. Clearly, the more Bell states are devoted to entanglement swapping, the less Bell states are available for implementing remote operations between neighbor quantum processors. Hence, a trade-off between “augmented connectivity” and “EPR cost” arises with entanglement swapping [18], and the impact of this trade-off on the overall performance of distributed quantum computing must be carefully accounted for.

Another fundamental issue arising with networking remote quantum processors is represented by noise and imperfections affecting the *quality* of the distributed Bell states. Clearly the noisier is the distributed Bell state, the noisier is the overall distributed quantum computation. Luckily, a well-known technique for counteracting the noise impairments affecting the entanglement generation/distribution process is constituted by *entanglement distillation* (also known as *entanglement purification*) [63, 76–81]. Accordingly, as long as the “quality” of the noisy entanglement exceeds a certain threshold, it is possible to purify multiple imperfect Bell states into a single “almost-maximally entangled” pair, albeit at the price of consuming multiple noisy entangled states within the process. From the above, it follows that one of two orthogonal resources must be exploited for implementing the distillation process, namely, time or space. More into details, time-expensive distillation requires multiple rounds of entanglement generation and distribution, with each round involving few¹⁷ communication qubits. Conversely, space-expensive distillation can be completed with few rounds, but with each round involving several communication qubits. Hence, there exists a fundamental trade-off between i) quality of the overall computation, ii) delay induced by entanglement distillation, and iii) communication qubits reserved for distilling a high-quality Bell state.

6 QUANTUM COMPILING

As mentioned in Section 3.3, quantum compilation means translating an input quantum circuit into the most efficient equivalent of itself, considering the characteristics of the device(s) that will execute the computation and minimizing the number of required multi-qubit gates. An example of quantum compilation is provided with Figure 15, where the original quantum circuit is translated into the compiled one to account for the coupling characteristics of IBM Yorktown quantum processors, shown in Figure 7. Clearly, as long as the hardware provides a universal set of operations, there exists a feasible transformation.

Compilers are well-established in NISQ architectures, because of their role as intermediary between the user and the hardware. Specifically, in designing a quantum algorithm using the quantum circuit formalism introduced in Section 3.2, the designer is generally focused on expressing the computation required by the algorithm with a circuit that minimizes the number of utilized qubits and gates, regardless from the particulars of the quantum hardware that will execute the circuit. This *abstract* circuit is then mapped to a circuit to be executed on a specific quantum hardware by means of a suitable compiler. Clearly, introducing such an abstract circuit has two main advantages: i) the user can focus on the logic of the circuit, namely, on the essence of the

¹⁷At least two communication qubits at each processor are required.

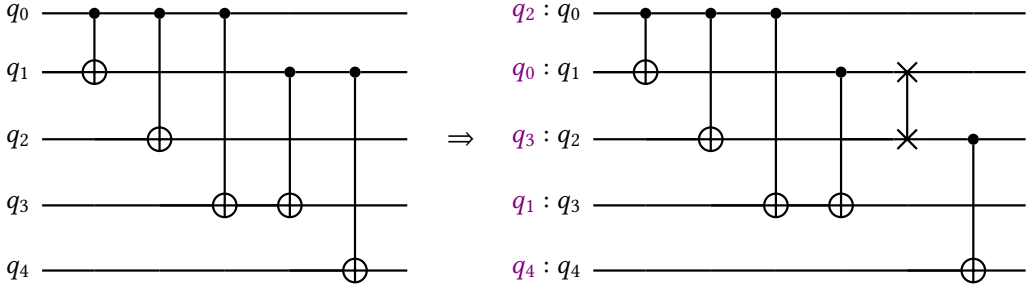


Fig. 15. Pictorial representation of quantum compiling. The circuit on the left is translated into the circuit on the right, in order to cope with the coupling map provided in Figure 7. Within the rightest figure, the q_i with purple font denotes the physical qubits assigned to the logical qubits q_j with black font. The SWAP gate—represented by two \times symbols interconnected by a vertical line—introduced between logical qubits q_1 and q_2 swaps their quantum states, so that the last CNOT gate can be applied between two neighbor physical qubits.

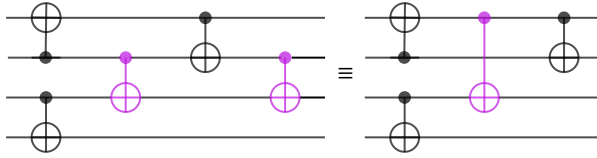


Fig. 16. Example of ebit optimization for the circuit of Figure 17: the left part of the equivalence can be optimized to the right one, which reduces the number of non-local gates.

quantum algorithm, without caring too much about the hardware constraints, and ii) the designed quantum circuit is portable, in theory, to any quantum back-end.

Intuitively, a circuit transformation may introduce some overhead, in terms of number of operations and noise. In DQC architectures, there is also a non-negligible communication cost, as discussed in Section 5. Therefore, the compiler faces an optimization problem, i.e., finding a feasible transformation while minimizing the overhead. In general, this problem is known to be NP-hard [43, 82], even for the case of a single processor.

A fundamental issue in quantum compiling is related to qubit connectivity. From the perspective of the quantum algorithm designer, any qubit is assumed to be directly connected with any other qubit, i.e., any two-qubit gate can be placed across any qubit pair. However, even on a single quantum processor as introduced in Section 3.3, the actual connectivity degree is usually low, to mitigate the noise caused by cross-talking phenomena [83]. *Qubit routing* refers to the task of modifying quantum circuits so that they satisfy the connectivity constraints of a target quantum computer. This involves inserting SWAP gates into the circuit so that the logical gates only ever occur between adjacent physical qubits. Of course, the number of SWAP gates should be minimized, in order keep the circuit depth reasonably small. The problem gets harder when considering distributed quantum processors, where the connectivity degree of the physical qubits can be even lower.

For DQC to be effective and efficient, the quantum compiler must perform some preliminary ebit optimization (such as the one illustrated in Figure 16), then find the best split for the abstract circuit, i.e., the split that minimizes the overall communication cost required to execute the distributed

Compiler	Language	Network Topologies	Qubit Assignment	Non-local Gate Handling	Open Source
[85]	Haskell	hypergraph	minimum k-cut	telegate and teledata	YES
[86]	unknown	hypergraph	minimum k-cut	telegate	NO
[87]	unknown	any	Tabu search	telegate and teledata	NO
[88]	MATLAB	/	heuristic	teledata	NO
[89]	MATLAB	/	dynamic programming	teledata	NO
[90]	C++ and CPLEX	n.a.	minimum k-cut	telegate and teledata	NO
[18]	Python	LLN	sorting	telegate and teledata	NO
[91]	pseudo-code	any	integer linear programming	telegate	/
[92]	MATLAB	n.a.	genetic alg.	teledata	NO
[64]	/	any	sorting	teledata	/
[65]	/	hypercube	sorting	teledata	/

Table 3. Comparison of DQC-oriented quantum compiling strategies. Some strategies find the best partition of the input monolithic quantum circuit in a completely network-agnostic fashion. Some strategies are purely theoretical, not supported by a software implementation.

circuit. At the same time, the quantum compiler must find the best local transformation for each piece of computation.

From the above, it should be clear that designing an efficient compiler is a tough task. Because of this, a plethora of proposals to tackle the problem emerges from the literature. In future work, some of them may be combined to more sophisticated compilers. This already happened for local computing. For example, the quantum compiler from the IBM Q framework [84] has several layers of optimization, each tackling the problem from different perspectives.

Most quantum compilers for DQC are characterized by two fundamental steps, namely *qubit assignment* and *non-local gate handling*. In the following, we present these two compilation steps, with reference to the most relevant literature. In Table 3, we compare some prominent DQC-oriented quantum compiling strategies. To this purpose, we consider the programming language, the supported network topologies, the qubit assignment strategy, the non-local gate handling strategy, and the availability of an open source release of the software.

In the remainder of the section, we first present some of the most representative strategies for qubit assignment and non-local gate handling. Then, we discuss some open issues.

6.1 Qubit Assignment

An abstract circuit is composed by *logical qubits*, while a quantum processor is equipped with a register of *physical qubits*. An assignment, in its most basic form, is a one-to-one mapping between logical and physical qubits.¹⁸ Whether it is better to tackle it dynamically – changing the assignment while computing – or statically – defining the assignment at the beginning and keeping it for the whole execution of the computation – is an open problem, which also depends on whether the partition between communication qubits and computing qubits is static or dynamic.

In DQC, qubit assignment is a general-purpose approach to the partitioning problem, introduced in Section 4.2. Specifically, for a given set of logical qubits, we need choose a partition that maps sub-sets of logical qubits to processors, while minimizing the number of required interactions among different sub-sets, as shown in Figure 17.

¹⁸One can also consider fault-tolerant mappings, where more than one physical qubit encode a single logical qubit. However we consider this as side work, out from the scope of this survey.

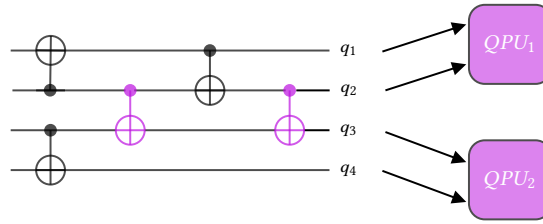


Fig. 17. Toy example of qubit assignment. Once the logical qubits composing the quantum circuit have been assigned to the different QPUs, the CNOTs between remote qubits – highlighted in violet – becomes non-local.

Several authors investigate this research direction [18, 85–87]. The reader will find in these works different proposals to address the qubit assignment problem. Not all the papers match in the minimum assumptions for the technology. Specifically, as described in Section 5, we are at a stage where one needs to make predictions on the most likely DQC architecture that will run in the next future. If one assumes any connectivity, the resulting model is general-purpose, but it is also hard to tackle. Restricting the connectivity to one that satisfies some properties makes the model less general, but a good set of assumptions in this direction may shape future implementations as well. Currently, the preferred line is to keep connectivity general [87].

Andrés-Martínez and Heunen [85] propose to encode a logical circuit as an hypergraph. An hyperedge represents one *ebit* – i.e., one EPR shared between QPUs – which allows for a telegate to be performed. Qubit assignment works by minimizing the number of cuts, as each cut corresponds to an ebit. Sundaram et al. [86] present a two-step solution, where the first step is quantum assignment. Circuits are represented as edge-weighted graphs with qubits as vertices. The edge weights correspond to an estimation for the number of *cat-entanglements*¹¹. The problem is then solved as a minimum k-cut, where partitions have roughly the same size. In [87], the same authors extend their approach to the case of an arbitrary-topology network of heterogeneous quantum computers by means of a Tabu search algorithm. In [88], by Daei et al., the circuit becomes an undirected graph with qubits as vertices, while edge weights correspond to the number of two-qubit gates between them. In [89], the authors represent circuits as bipartite graphs with two sets of vertices – one set for the qubits and one for the gates – and edges to encode dependencies of qubits and gates. Then for the qubit assignment problem, they propose a partitioning algorithm via dynamic programming to minimize the number of teledata operations.

When qubit assignment is dynamic, new challenges – as well as new possibilities – arise. Nikahd et al. [90] propose a minimum k-cut partitioning algorithm formulated as an ILP optimization problem, to minimize the number of remote interactions. They use a moving window and apply the partitioning algorithm to small sections of the circuit, thus the partition may change with the moving window by means of teledata operations. In [18], Ferrari et al. consider the worst-case scenario of QPUs interconnected through an LNN topology¹⁹. Rather than focusing on the number of remote interactions, they design a sorting algorithm to reduce the depth overhead induced by such time consuming operations. The authors show that the overhead is upper-bounded by a factor that grows linearly with the number of qubits. Cuomo et al. in [91] model the compilation problem with an Integer Linear Programming formulation. The formulation is inspired to the vast theory on

¹⁹The Linear Nearest Neighbor (LNN) topology [93] consists of processors arranged in a single line – namely, in a 1-dimensional lattice – where each processor is interconnected with two neighbors. In the worst-case scenario – namely, the most challenging one – each QPU is equipped with a single computational qubit, and only neighboring qubits can interact each others.

dynamic network problems. Authors managed to define the problem as a special case of *quickest multi-commodity flow*. Such a result allows to perform optimization by means of techniques coming from the literature, such as a *time-expanded* representation of the distributed architecture.

6.2 Non-local Gate Handling

As described in Section 5, assumptions on the architectures not only concern connectivity. Predicting the best kind of remote interactions is of critical importance as well. In this sense, the general agreement is that the generation and distribution of entangled states is a fundamental resource to be used sparingly. Indeed, a common goal in the literature is to minimize the number of consumed ebits, as it is the main bottleneck to distributed quantum computation. To this aim, qubit assignment discussed above represents a starting point for further optimization steps, which now concern circuit manipulation.

As described in Section 5.3, there are two main approaches for implementing non-local gates, namely teledata and telegate.

The teledata approach is considered, for example, in [64, 65, 88, 89, 92]. Beals et al. [64] prove that the quantum circuit model, the quantum parallel RAM model, and the DQC model are equivalent up to polylogarithmic depth overhead. Other than this major result, they provide an algorithm for emulating circuits on any network graph. Brierley [65] focuses on n -qubit cyclic butterfly networks (a special case of hypercubic network) and proves that there is a sequence of local gates with depth $6 \log n$ such that the qubit at node a is sent to node $\pi(a)$ for all $a = 1, \dots, n$ and any permutation $\pi : [1, n] \rightarrow [1, n]$. In other words, the butterfly network can implement any quantum algorithm with an overhead of $6 \log n$. Such a network topology is suitable for multi-chip quantum devices or small controlled networks. In medium-scale or global networks, it is hard to implement such a constrained architecture. Daei et al. [88] propose a method to minimize the number of quantum teleportations between DQC partitions. The main idea is to turn the monolithic quantum circuit into an undirected weighted graph, where the weight of each edge represents the number of gates involving a specific pair of qubits for execution. Then, the graph is partitioned using the Kernighan-Lin (K-L) algorithm for VLSI design [94], so that the number of edges between partitions is minimized. Finally, each graph partition is converted to a quantum circuit. Davarzani et al. [89] propose an algorithm for minimizing teleportations consisting of two steps: first, the quantum circuit is converted into a bipartite graph model, and then a dynamic programming approach (DP) is used to partition the model into low-capacity quantum circuits. Finally, Dadkhah et al. [92] propose a heuristic approach to replace the equivalent circuits in the initial quantum circuit. Then, they use a genetic algorithm to partition the placement of qubits so that the number of teleportations could be optimized for the communications of a DQC.

The telegate direction is pursued, for example, in [85, 86, 91]. Andrés-Martínez et al. [85] use cat-entanglement¹¹ to implement non-local quantum gates. The chosen gate set contains every one-qubit gate and a single two-qubit gate, namely the CZ gate (i.e., the controlled version of the Z gate). The authors consider no restriction on the ebit connectivity between QPUs. Then, they reduce the problem of distributing a circuit across multiple QPUs to hypergraph partitioning. The proposed approach is evaluated against five quantum circuits, including QFT. The proposed solution has some drawbacks, in particular that there is no way to customize the number of communication qubits of each QPU. As previously mentioned, in Sundaram's et al. paper [86], a two-step quantum compiling approach is introduced. The first step is qubit assignment, while the second step is finding the smallest set of cat-entanglement operations that will enable the execution of all telegates. The authors state that, in a special setting, this problem can be reduced to a vertex-cover problem, allowing for a polynomial-time optimal solution based on integer linear programming. They also provide a $O(\log n)$ -approximate solution, where n is the total number of

global gates, for a generalized setting by means of greedy search algorithm. Also the aforementioned work by Cuomo et al. [91] adopts the telegate approach.

6.3 Open Issues and Research Directions

The most advanced quantum compilers for execution on single quantum processors are noise-aware, i.e., they take the noise statistics of the device into account, for some or all steps [45, 95–98]. A noise-aware quantum compiler for DQC is still missing. Indeed, it is still an open question what kind of noise-awareness such a compiler should have. The different options range from a compiler that has complete knowledge of the target execution platform (quantum processors, quantum links, etc.) to a compiler that only knows generic features of the target quantum processors and network – as the execution manager will decide the actual execution platform assigned to the computation.

Further work could be done regarding the integration of quantum compilers with simulation tools – in line with the preliminary attempt that was made by Ferrari et al. [99] – allowing for automated workflows that would allow for faster comparative evaluation of compiling strategies.

So far, testing the quality of compiled circuits on real execution platforms has not been possible for the majority of researchers. Once a quantum network will be available to the public – much like current IBM Q, Rigetti, etc. single quantum devices – it will be possible to evaluate DQC compilers more effectively, with key performance indicators including the resulting computation quality, state fidelity, and other performance metrics [100].

7 SIMULATION TOOLS

To support the research community in the design and evaluation of quantum computing and quantum network technologies, including hardware, protocols and applications, many simulation tools have been developed recently.

Simulations are very important for several reasons. First of all, they allow for defining hardware requirements using a top-down approach, i.e., starting from applications and protocols. In this way, hardware design is driven by high-level KPIs (key performance indicators), rather than proceeding by trial and error. Another advantage of simulations is related to network sizing. Given the number of potential users and the number of available quantum processors, simulation allows for devising and evaluating different network topologies and entanglement routing schemes, which results in saving time and money. Regarding DQC, simulation plays a crucial role for establishing the correctness of the compiled distributed quantum programs, and evaluating the quality of their execution against different hardware platforms, network configurations and scheduling algorithms.

In Table 4, we compare some prominent simulation tools that, in our view, can be used for designing and evaluating DQC systems. We propose to classify each tool as belonging to one of three possible classes: i) hardware-oriented (HW), ii) protocol-oriented (PR), and iii) application-oriented (AP). In the remainder of the section, we first present each class with some of the most representative simulation tools. Then, we discuss some open issues.

7.1 Hardware-oriented

We denote as HW simulation tools those that allow the user to model the physical entities with the desired degree of detail, including noise models. Prominent examples are SQUANCH [101] and NetSquid [102], discussed in the following. Regarding DQC, we note that HW simulation tools are useful for evaluating the impact of different hardware technologies (including noise models) on the quality of the distributed program execution.

The *Simulator for Quantum Networks and Channels* (SQUANCH) [101] is an open-source Python framework for creating parallelized simulations of distributed quantum information processing. Despite the framework includes many features of a general-purpose quantum computing simulator,

Simulation Tool	Language	Multiprocessing	Multithreading	Noise Models	Open Source	Class
SQUANCH [101]	Python	NO	NO	YES	YES	HW
NetSquid [102]	Python	NO	NO	YES	NO	HW
SimulaQron [103]	Python	YES	NO	NO	YES	PR
SeQUeNCe [104]	C++/Python	YES	NO	YES	YES	PR
QuiSP [105]	C++	NO	NO	YES	YES	PR
QuNetSim [106]	Python	NO	YES	NO	YES	PR
NetQASM SDK [107]	C++/Python	NO	YES	YES	YES	AP
QNE-ADK [108]	C++/Python	NO	NO	YES	NO	AP

Table 4. Comparison of simulation tools that can be used for designing and evaluating DQC systems.

it is optimized specifically for simulating quantum networks. It includes functionality to allow users to design complex multi-party quantum networks, extensible classes for modeling noisy quantum channels, and a multiprocessed NumPy backend for performant simulations. The core modules are QSystem, representing a multi-body quantum system as a density matrix in the computational basis, and QStream, which is an iterable ensemble of separable N -qubit QSystems optimized for cache locality. By default QStream state is stored in a shared memory as a C-type array of doubles, which is type-casted as a 3D array of `np.complex64` values. During simulations, Agents run in parallel from separate processes, synchronizing clocks and passing information between each other through Channels. There is no explicit concurrency safety when a QSystem is modified by multiple agents, as sending and receiving Qubits are blocking operations that allow for naturally safe parallelism. However, the scalability of this simulation tool is hindered by the lack of support for distributed multiprocessing, as all the processes must run on the same machine. The source code is not maintained since 2018.

NetSquid [102] is one of the most advanced platforms for simulating quantum networking and modular computing systems subject to physical non-idealities. It ranges from the physical layer and its control plane up to the application level. This is achieved by integrating several key technologies: a discrete-event simulation engine, a specialized quantum computing library, a modular framework for modeling quantum hardware devices, and an asynchronous programming framework for describing quantum protocols. NetSquid has been used for different purposes, such as the evaluation of a benchmarking procedure for quantum protocols [109], the evaluation of end-to-end entanglement generation strategies in terms of capacity bounds and impact on Quantum Key Distribution (QKD) [110, 111], and the performance evaluation of request scheduling algorithms for quantum networks [112].

7.2 Protocol-oriented

In the proposed classification, PR simulation tools are mostly devoted to the design and evaluation of general-purpose quantum protocols, – such as quantum state teleportation, quantum leader election, etc. [113] – with the possibility to model hardware-agnostic networked quantum processors, with very limited (if not missing) support for noise modeling. Relevant examples are SimulaQron [103], SeQUeNCe [104], QuiSP [105] and QuNetSim [106]. Regarding DQC, PR simulation tools are useful for evaluating the impact of different compiling and execution management strategies on the quality of the distributed program execution, in (almost) ideal conditions.

SimulaQron [103] is a tool for developing distributed software that runs on real or simulated classical and quantum end-nodes, connected by classical and quantum links. SimulaQron spawns

three stacked processes per network node: the lowest one for wrapping a simulated quantum registry, based on an hardware-specific third-party simulator; the intermediate process exposing simulated qubits that map 1-to-1 to those of the quantum registry; the upper process providing virtual qubits that are manipulated within a platform-independent application. For example, if two virtual qubits belonging to different processes, running on physically-separated servers, are manipulated in order to share an entangled state (let say, a Bell state), the corresponding simulated qubits (and quantum register ones) are both stored in the memory of one server, in order to make it possible to simulate measurements in a consistent fashion. This process-oriented approach makes SimulaQron quite scalable and able to leverage multicore server architecture in order to speed up the execution of the simulations. However, SimulaQron does not come with noise model support, thus preventing the simulation of quantum protocols over non-ideal networks.

SeQUeNCe [104] is an open-source discrete-event quantum network simulator, whose latest release fully supports parallel simulation. The authors designed and developed a quantum state manager (QSM) that maintains shared quantum information distributed across multiple processes, and also optimized their parallel code by minimizing the overhead of the QSM and by decreasing the amount of synchronization among processes.

QuiSP [105] is an event-driven Quantum Internet simulation package. QuiSP is built on top of the OMNeT++ discrete event simulation framework. Compared to the simulators discussed so far, many of which focus on physically realistic simulation of a single small network, QuiSP is oriented to protocol design for complex, heterogeneous networks at large scale while keeping the physical layer as realistic as possible. Emphasis has been placed on realistic noise models. The declared long-term goal for the simulator is to be able to handle an internetwork with 100 networks of 100 nodes each. To simulate quantum networks at the cost of only a few classical bits per qubit, QuiSP works *in the error basis*, i.e., tracking only errors, not states. The premise is that the desired quantum state is known and only deviations from this ideal state must be tracked. This is a novel approach for simulating quantum networks, adapted from quantum error correction [114]. The performance of QuiSP was investigated in terms of events processed per second and the duration of CPU time taken to generate one end-to-end Bell pair, using the Docker environment that QuiSP provides. It was shown in [105] that the average CPU time (in seconds) per end-to-end Bell pair generated grows no worse than polynomially in the number of quantum repeaters. Increasing the number of repeaters results in longer simulation time in the scaling, as expected. It also emerged that that QuiSP might have some kind of unintended overhead which scales linearly on the number of buffer qubits, which the authors expect to fix in a near-term release [105].

QuNetSim [106] implements a layered model of network component objects inspired by the OSI model. In particular, application, transport, and network layers are considered. QuNetSim does not explicitly incorporate features of the link and physical layers. Indeed, QuNetSim relies on open-source qubit simulators that are used to simulate the physical qubits in the network, namely SimulaQron [103], ProjectQ [115] and EQSN [116] (the latter one being the default backend, as it was developed by the QuNetSim team). In QuNetSim, network nodes can run both classical and quantum applications. The transport layer component prepares classical packets, encodes qubits for superdense message transmission, handles the generation of the two correction bits for quantum state teleportation, etc. The network layer component can route classical and quantum information using two internal network graphs and two different routing algorithms. The network component objects are implemented using threading and observing queues. Extensive use of threading allows each task to wait without blocking the main program thread, which simulates the behavior of sending information and waiting for an acknowledgment, or expecting information to arrive for some period of time from another host. QuNetSim works well for small scale simulations using five

to ten hosts that are separated by a small number of hops, while it tends to reach its limits when many entangled qubits are being generated across the network with many parallel operations.

7.3 Application-oriented

The third class is devoted to AP simulation tools, which are tailored to the design and implementation of quantum network applications. Usually, these tools rely on simulated backends offered by other packages that are not directly accessible to the user – for example, NetQASM SDK [107] relying on NetSquid [102]. Regarding DQC, AP simulation tools are useful for quickly assessing the quality of quantum circuit splits produced by quantum compilers. The execution management scheme (i.e., job scheduling, entanglement routing, etc.) is hidden to the user, which is at most allowed to specify the network topology (from a short list of preconfigured networks) and the values of a few parameters characterizing the hardware of the quantum processors.

The process of setting up a simulation requires strong expertise in the simulator itself, thus being inconvenient for those who are only interested in quantum protocol evaluation or in the design of supporting tools such as quantum compilers. Recently, Ferrari *et al.* [99] presented a software tool, denoted as DQC Executor, that accepts as input the description of the network and the code of the algorithm, and then executes the simulation by automatically constructing the network topology and mapping the computation onto it, in a framework-agnostic way and transparently to the user. The tool is in its early stages and currently supports automatic deployment of distributed quantum algorithms to the NetSquid [102] simulator. The description of the network is provided by the user in a specific YAML format. The distributed algorithm, instead, is defined with the OpenQASM [117] language.

NetQASM SDK [107] is a high-level software development kit, in Python, whose purpose is to make easier to write quantum network applications, to simulate them through NetSquid [102] or SimulaQron [103], and (expected in the near future) to execute them on real hardware. Indeed, the quantum programs developed with NetQASM SDK are translated into low-level programs based on the NetQASM language, similar in nature to classical assembly languages. With respect to other QASM languages, NetQASM provides elements for remote entanglement generation. On the other hand, NetQASM contains no provision for classical communication with remote nodes. Synchronization between the NetQASM programs (through classical send/recv primitives) of multiple nodes is the responsibility of the application programmer.

The Quantum Network Explorer Application Development Kit (QNE-ADK) [108] allows the user to create applications and experiments and run them on a simulator. When configuring an application, the user specifies the different roles and what types of inputs the application uses. In addition, the user writes the functionality of the application using the NetQASM SDK [107]. When configuring an experiment, the user can give values to the inputs that were specified when creating the application. The user also chooses which channels and nodes are used in the network and which role is linked to which node. Once configured, the experiment is parsed and sent to the NetSquid simulator [102]. QNE-ADK is particularly useful when the application code developed with NetQASM SDK is provided to the user, whose only duty is to configure and perform experiments. Indeed, using the execution environment is straightforward. There is also a visual interface that further simplifies the experiment configuration.

Both NetQASM SDK [107] and QNE-ADK [108] are very useful tools. Without them, configuring DQC simulations is quite a complex task.

7.4 Open Issues and Research Directions

There is a sufficiently variegated choice of simulation tools for quantum networks and backends to support DQC research, with specialization on hardware, protocols, or applications. On the other

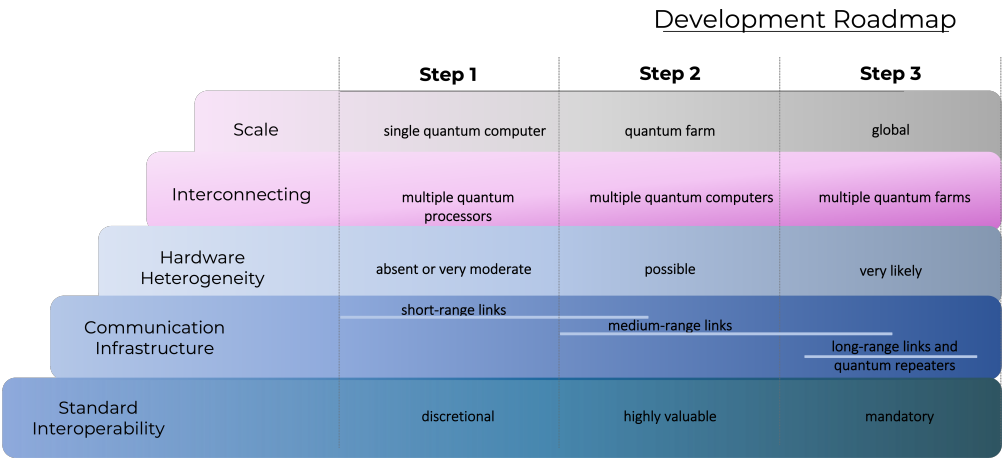


Fig. 18. Possible stages for the development of distributed quantum computing. It is reasonable to assume that the underlying hardware complexity scales proportional with three dimensions: i) the extension of the communication infrastructure, ii) the number of interconnected quantum devices, iii) and the hardware heterogeneity among the quantum devices.

hand, a simulation tool allowing for full-stack simulation of large networks is still missing. Such a tool should support multiprocessing and multithreading, and simple deployment of DQC simulations on high performance computing facilities.

Another possible direction is the development of tools for orchestrating DQC simulations, with automated instantiation of simulation objects representing QPUs and quantum network components. Having quantum compilers for DQC in the loop would be also very useful. Last but not least, it would be great to have the possibility to seamlessly replace simulated hardware with real devices.

8 CONCLUSIONS AND FUTURE PERSPECTIVES

Here we conclude the survey by first providing an industrial perspective on distributed quantum computing, and then by discussing the possible stages of distributing quantum computing development.

8.1 Industrial and Standardization Perspective

A first quantum revolution has already exploited quantum technologies in our everyday life, creating a deep techno-economic and social impact. Today, a second revolution is underway, and it is safe to predict it will have a major impact in many markets, ranging from Telecom and ICT, through Medicine, to Finance and Transportation, and so on.

Clearly, significant work is still needed to develop enabling components and systems for DQC. Yet, considering the foreseen industrial opportunities, significant investments are being made worldwide across public and private organizations.

One major obstacle on the way of industrial exploitation of distributed quantum computing is that, nowadays, the industry has not yet consolidated around one type of quantum hardware technology. In this scenario, a *quantum hardware abstraction layer* (Quantum-HAL) – embracing the two killer domains of quantum technologies for ICT, namely, quantum computing and quantum

networking – would allow applications and services developers to start using the abstractions of the underneath quantum hardware, even if still under consolidation. This would definitely simplify and speed-up the development of quantum platforms, services, and applications. Indeed, a Quantum-HAL for distributed quantum computing would provide unified northbound quantum application programming interfaces (APIs) for the higher layers, decoupling from the different types of quantum hardware technologies (e.g., trapped ions, superconducting qubits, silicon photonic qubits).

Another key aspect for increasing the TRL (Technology Readiness Level) of distributed quantum computing concerns its integration with current Telecom and ICT infrastructures. This implies the definition and standardization of a management and control approach (architectures and APIs) able of interworking with current solutions. All these activities require coordinated and joint efforts including – where appropriate – existing projects, industry bodies and standard (ITU-T, ETSI, IETF and IEEE just to mention a few) active in the area of quantum technologies.

Overall, the final goal is to bridge the gap between DQC and the established cloud and edge computing platforms, tools and methods, and to focus in on the inter-related constraints between the different aspects of the architectural design, so to enable the development of practical DQC solutions. To achieve this goal, research and innovation activities are required in diverse and complementary fields, ranging from computational complexity and networked systems through quantum information and optics to communications and computer science engineering.

8.2 The Journey Ahead

Paving a journey towards distributed quantum computing is a challenging task, as hard as any other prediction about technological developments. Yet, we can sketch roughly three stages, as discussed in the following and summarized with Figure 18.

The first step involves distributed quantum computing exploiting multiple quantum processors within a single quantum computer. The quantum hardware underlying the qubits is likely to be homogeneous among the different processors. Yet, some sort of hardware heterogeneity may arise within each processor due to the differences in terms of requirements²⁰ between memory qubits and computational qubits. The physical distance between remote qubits is clearly very short. Hence, it is reasonable to assume, as communication infrastructure, short-range microwave links. The network topology is likely static, so that only simple quantum network functionalities are required. Clearly, quantum decoherence must be carefully accounted for, so that the decoherence time can be used as overall key metric. Local operations between qubits within a single processor must be complemented by remote operations between qubits placed at different processors. The trade-off between qubits devoted to computation and entangled qubits devoted to communication represents a fundamental issue with no counterpart in classical distributed computing. The very challenging task of designing distributed quantum algorithms must explicitly take such trade-off – as well as the delay induced by remote operations – into consideration.

The second step involves inter-rack distributed quantum computing, where the computation is performed collectively by multiple quantum computers located within the same farm. At this stage, some sort of hardware heterogeneity might arise, given that different quantum computers are involved in the computation. Clearly, such heterogeneity must be taken into consideration at each layer of a distributed quantum computing ecosystem. Yet, entanglement distribution still benefits from a tightly controlled environment – reasonable to assume available within a single quantum farm – and the relatively short distances. As a matter of fact, the communication infrastructure can still be composed by cold microwave links [118] although optical links would greatly simplify the

²⁰Quantum memory requires coherence times several order of magnitude larger than computational qubits.

hardware requirements albeit at the price of significant technological advances in the microwave-optical conversion. Delay imposed by classical and quantum communication times is slightly longer – when compared to stage one – hence more sophisticated timing and synchronization functionalities are required. The network topology becomes more complex, and it may present some sort of temporal dynamics as the number of interconnected quantum computers might change in time. This, in turn, induces network functionalities dynamics that must be carefully taken into account. The problem of remote operations compiling – and, hence, the trade-off between computational and communication qubits – becomes even more intricate. Finally, at this stage, the execution management problem (previously discussed in Section 4.2) will arise, with multiple users performing concurrent access to the resources.

The third step involves interconnecting multiple geographically-distributed quantum farms. Two are the key challenges here. First, there exists a likely spread heterogeneity – given that the different quantum farms will be likely operated by different companies –, which requires significant efforts in terms of standardization and interoperability. Furthermore, the heterogeneity among quantum links, e.g., optical, free-space or satellite, will arise. The delays induced by the distances will introduce severe challenges on the entanglement generation and distribution. The increasing number of quantum devices to be wired and the heterogeneity of the environments hosting the quantum computers must be taken into account as well. At this stage, the compiling and execution management problems would be even more complex, demanding for specific network services to be integrated with those of the classical Internet (such as DNS, DHCP, etc.).

We underline that, although each successive stage is distinguished by an increasing amount of interconnected quantum resources, the actual deployment evolution will strongly depend on the technological advances and the experimental implementations of the different entities composing a distributed quantum computing ecosystem [12].

One of the judicious questions raised from this discussion is: when will we see the distributed quantum computing? There is no definite answer to this question. However, we firmly believe this is a goal that requires a collaborative effort and a multi-disciplinary approach between academics and industries. The required competences and skills are many and diverse and each is interconnected with and vital to the others.

REFERENCES

- [1] Angela Sara Cacciapuoti, Marcello Caleffi, Francesco Tafuri, Francesco Saverio Cataliotti, Stefano Gherardini, and Giuseppe Bianchi. Quantum internet: Networking challenges in distributed quantum computing. *IEEE Network*, 34(1): 137–143, 2020. doi: 10.1109/MNET.001.1900092.
- [2] P. P. Rohde. *The Quantum Internet – The Second Quantum Revolution*. Cambridge University Press, 2021.
- [3] EU Quantum Flagship. The Future is Quantum. URL <https://qt.eu/>.
- [4] QIA Team. Quantum Internet Alliance. URL <https://quantum-internet.team/>.
- [5] European Commission and European Space Agency. The European Quantum Communication Infrastructure (EuroQCI) Initiative. URL <https://digital-strategy.ec.europa.eu/en/policies/european-quantum-communication-infrastructure-euroqci>.
- [6] Will Finigan. Quantum network research centers in the US. URL <https://www.aliroquantum.com/blog/quantum-network-research-centers-in-the-us>.
- [7] Q-NEXT. Q-NEXT website. URL <https://q-next.org/>.
- [8] HQAN. HQAN website. URL <https://hqan.illinois.edu/>.
- [9] CQN. CQN website. URL <http://cq-n-erc.org/>.
- [10] Juan Yin, Yuan Cao, Yu-Huai Li, et al. Satellite-based entanglement distribution over 1200 kilometers. *Science*, 356(6343):1140–1144, 2017.
- [11] Amara Graps. How Much Money Has China Already Invested into Quantum Technology? - Part 2. URL <https://quantumcomputingreport.com/how-much-money-has-china-already-invested-into-quantum-technology/>.
- [12] Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. Towards a distributed quantum computing ecosystem. *IET Quantum Communication*, 1:3–8(5), July 2020. doi: 10.1049/iet-qtc.2020.0002.

- [13] QCommHub. QCommHub website. URL <https://www.quantumcommshub.net/>.
- [14] Amazon. Announcing the AWS Center for Quantum Networking. URL <https://aws.amazon.com/blogs/quantum-computing/announcing-the-aws-center-for-quantum-networking/>.
- [15] Chonggang Wang, Akbar Rahman, Ruidong Li, Melchior Aelmans, and Kaushik Chakraborty. Application scenarios for the quantum internet. Internet-Draft draft-irtf-qirg-quantum-internet-use-cases-12, Internet Engineering Task Force, 2022. Work in Progress.
- [16] Rodney Van Meter and Simon J. Devitt. The Path to Scalable Distributed Quantum Computing. *Computer*, 49(9): 31–42, September 2016. ISSN 0018-9162. doi: 10.1109/MC.2016.291.
- [17] Marcello Caleffi, Angela Sara Cacciapuoti, and Giuseppe Bianchi. Quantum internet: From communication to distributed computing! In *Proc. of ACM NANOCOM '18*, pages 1–4. Association for Computing Machinery, 2018. ISBN 9781450357111. doi: 10.1145/3233188.3233224.
- [18] Davide Ferrari, Angela Sara Cacciapuoti, Michele Amoretti, and Marcello Caleffi. Compiler design for distributed quantum computing. *IEEE Transactions on Quantum Engineering*, 2:1–20, 2021. doi: 10.1109/TQE.2021.3053921.
- [19] J. Avron, Ofer Casper, and Ilan Rozen. Quantum advantage and noise reduction in distributed quantum computing. *Phys. Rev. A*, 104:052404, Nov 2021. doi: 10.1103/PhysRevA.104.052404.
- [20] Stephanie Wehner, David Elkouss, and Ronald Hanson. Quantum Internet: a Vision for the Road Ahead. *Science*, 362 (6412), 2018.
- [21] Marcello Caleffi, Daryus Chandra, Daniele Cuomo, Shima Hasaanpour, and Angela Sara Cacciapuoti. The Rise of the Quantum Internet. *IEEE Computer*, 2020.
- [22] IBM. Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing. URL <https://research.ibm.com/blog/ibm-quantum-roadmap-2025>.
- [23] 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)*, pages 9–19, Los Alamitos, CA, USA, nov 2021. IEEE Computer Society. doi: 10.1109/QCS54837.2021.00005. URL <https://doi.ieeecomputersociety.org/10.1109/QCS54837.2021.00005>.
- [24] Youpeng Zhong, Hung-Shen Chang, Audrey Bienfait, et al. Deterministic multi-qubit entanglement in a quantum network. *Nature*, 590(7847):571–575, 2021.
- [25] M. Pompili, S. L. N. Hermans, S. Baier, et al. Realization of a multinode quantum network of remote solid-state qubits. *Science*, 372(6539):259–264, 2021.
- [26] S. L. N. Hermans, M. Pompili, H. K. C. Beukers, et al. Qubit teleportation between non-neighbouring nodes in a quantum network. *Nature*, 605(7911):663–668, 2022.
- [27] Angela Sara Cacciapuoti, Marcello Caleffi, Rodney Van Meter, and Lajos Hanzo. When entanglement meets classical communications: Quantum teleportation for the quantum internet. *IEEE Transactions on Communications*, 68(6): 3808–3833, 2020. invited paper.
- [28] Eleanor Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists. *ACM Computing Survey*, 32(3):300–335, sep 2000.
- [29] Eleanor Rieffel and Wolfgang Polak. *Quantum Computing: A Gentle Introduction*. The MIT Press, 2011.
- [30] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2011.
- [31] P. A. M. Dirac. A new notation for quantum mechanics. *Mathematical Proceedings of the Cambridge Philosophical Society*, 35(3):416–418, 1939.
- [32] Richard Jozsa and Noah Linden. On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 459(2036):2011–2032, 2003. doi: 10.1098/rspa.2002.1097.
- [33] Damian Markham and Barry C. Sanders. Graph states for quantum secret sharing. *Phys. Rev. A*, 78:042309, 2008.
- [34] J. S. Bell. On the Einstein Podolsky Rosen paradox. *Physica Physique Fizika*, 1:195–200, Nov 1964.
- [35] Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical review*, 47(10):777, 1935.
- [36] Daniel Gottesman. Theory of fault-tolerant quantum computation. *Phys. Rev. A*, 57:127–137, Jan 1998.
- [37] Abhijith J., Adetokunbo Adedoyin, John Ambrosiano, et al. Quantum algorithm implementations for beginners. *ACM Transactions on Quantum Computing*, 3(4), jul 2022. ISSN 2643-6809.
- [38] Seid Koudia, Angela Sara Cacciapuoti, Kyrlo Simonov, and Marcello Caleffi. How deep the theory of quantum communications goes: Superadditivity, superactivation and causal activation. *IEEE Communications Surveys & Tutorials*, 2022. In press.
- [39] Norbert M. Linke, Dmitri Maslov, Martin Roetteler, et al. Experimental comparison of two quantum computing architectures. *Proceedings of the National Academy of Sciences*, 114(13):3305–3310, 2017.

- [40] Abhinav Kandala, Kristan Temme, Antonio D. Córcoles, et al. Error mitigation extends the computational reach of a noisy quantum processor. *Nature*, 567(7749):491–495, 2019.
- [41] Ziang Yu and Yingzhou Li. Analysis of error propagation in quantum computers. *arXiv e-prints*, 2022. arXiv:2209.01699.
- [42] R. Van Meter and S. J. Devitt. The path to scalable distributed quantum computing. *Computer*, 49(9):31–42, Sept 2016. ISSN 0018-9162. doi: 10.1109/MC.2016.291.
- [43] A. Botea, A. Kishimoto, and R. Marinescu. On the Complexity of Quantum Circuit Compilation. In *The Eleventh International Symposium on Combinatorial Search (SOCS 2018)*, 2018.
- [44] Janusz Kussyk, Samah M. Saeed, and Muharrem Umit Uyar. Survey on quantum circuit compilation for noisy intermediate-scale quantum computers: Artificial intelligence to heuristics. *IEEE Transactions on Quantum Engineering*, 2:1–16, 2021. doi: 10.1109/TQE.2021.3068355.
- [45] Seyon Sivarajah, Silas Dilkes, Alexander Cowtan, et al. t|ket>: a retargetable compiler for nisq devices. *Quantum Science and Technology*, 6(1):014003, nov 2020.
- [46] A. D. Carcoles, A. Kandala, A. Javadi-Abhari, et al. Challenges and opportunities of near-term quantum computing systems. *Proc. of the IEEE*, pages 1–15, 2020. in press.
- [47] Davide Ferrari and Michele Amoretti. Efficient and effective quantum compiling for entanglement-based machine learning on ibm q devices. *International Journal of Quantum Information*, 16(08):1840006, 2018.
- [48] Lukasz Cincio, Yiğit Subaşı, Andrew T Sornborger, and Patrick J Coles. Learning the quantum algorithm for state overlap. *New Journal of Physics*, 20(11):113022, Nov. 2018.
- [49] A. Zulehner, A. Paler, and R. Wille. An efficient methodology for mapping quantum circuits to the ibm qx architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 38(7):1226–1236, 2019.
- [50] Ashley Montanaro. Quantum algorithms: An overview. *npj Quantum Information*, 2(1):15023, January 2016. ISSN 2056-6387. doi: 10.1038/npjqi.2015.23.
- [51] Peter W. Shor. Polynomial time algorithms for discrete logarithms and factoring on a quantum computer. In *Algorithmic Number Theory*, pages 289–289. Springer Berlin Heidelberg, 1994. ISBN 978-3-540-49044-9.
- [52] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96*, page 212–219, 1996. ISBN 0897917855.
- [53] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009.
- [54] M. Cerezo, Andrew Arrasmith, Ryan Babbush, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9): 625–644, September 2021.
- [55] Barbara M. Terhal. Quantum error correction for quantum memories. *Rev. Mod. Phys.*, 87:307–346, Apr 2015.
- [56] Niels M. P. Neumann, Roy van Houte, and Thomas Attema. Imperfect Distributed Quantum Phase Estimation. In *Computational Science – ICCS 2020, Lecture Notes in Computer Science*, pages 605–615. Springer International Publishing, 2020.
- [57] A Yu Kitaev. Quantum computations: algorithms and error correction. *Russian Mathematical Surveys*, 52(6):1191–1249, dec 1997.
- [58] J. Eisert, K. Jacobs, P. Papadopoulos, and M. B. Plenio. Optimal local implementation of nonlocal quantum gates. *Phys. Rev. A*, 62:052317, Oct 2000.
- [59] Stephen DiAdamo, Marco Ghibaudo, and James Cruise. Distributed Quantum Computing and Network Control for Accelerated VQE. *IEEE Transactions on Quantum Engineering*, 2:1–21, 2021. ISSN 2689-1808. doi: 10.1109/TQE.2021.3057908.
- [60] Anocha Yimsiriwattana and Samuel J. Jr. Lomonaco. Generalized GHZ states and distributed quantum computing. *Contemp. Math.*, 381, 2005. doi: 10.1090/conm/381.
- [61] Niels M. P. Neumann and Robert S. Wezeman. Distributed quantum machine learning. In *Innovations for Community Services*, pages 281–293. Springer International Publishing, 2022.
- [62] Claudio Ciconetti, Marco Conti, and Andrea Passarella. Resource allocation in quantum networks for distributed quantum computing. In *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 124–132, 2022.
- [63] J. I. Cirac, A. K. Ekert, S. F. Huelga, and C. Macchiavello. Distributed quantum computation over noisy channels. *Phys. Rev. A*, 59:4249–4254, Jun 1999.
- [64] Robert Beals, Stephen Brierley, Oliver Gray, et al. Efficient distributed quantum computing. *Proc. of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153):20120686, 2013.
- [65] Stephen Brierley. Efficient implementation of quantum circuits with limited qubit interactions. *Quantum Info. Comput.*, 17(13–14):1096–1104, November 2017. ISSN 1533-7146.
- [66] Gayane Vardoyan and Stephanie Wehner. Quantum Network Utility Maximization. *arXiv e-prints*, 2022. arXiv:2210.08135v1.
- [67] Yuan Lee, Wenhen Dai, Dan Towsley, and Dirk Englund. Quantum Network Utility: A Framework for Benchmarking Quantum Networks. *arXiv e-prints*, 2022. arXiv:2210.10752v1.

- [68] Andrew W. Cross, Lev S. Bishop, Sarah Sheldon, Paul D. Nation, and Jay M. Gambetta. Validating quantum computers using randomized model circuits. *Phys. Rev. A*, 100:032328, Sep 2019. doi: 10.1103/PhysRevA.100.032328.
- [69] Wojciech Kozłowski, Stephanie Wehner, Rodney Van Meter, Bruno Rijsman, Angela Sara Cacciapuoti, Marcello Caleffi, and S. Nagayama. Architectural principles for a quantum internet. Internet-Draft draft-irtf-qirg-principles-10, Internet Engineering Task Force, 2022. Work in Progress.
- [70] Jessica Illiano, Marcello Caleffi, Antonio Manzalini, and Angela Sara Cacciapuoti. Quantum internet protocol stack: a comprehensive survey. *Computer Networks*, 213, 2022.
- [71] Angela Sara Cacciapuoti, Seid Illiano, Jessica Koudia, and Marcello Caleffi. The quantum internet: Enhancing classical services one qubit at a time. *IEEE Networks*, 2022.
- [72] Angela Sara Cacciapuoti and Marcello Caleffi. Toward the quantum internet: A directional-dependent noise model for quantum signal processing. In *IEEE ICASSP '19*, pages 7978–7982, 2019. doi: 10.1109/ICASSP.2019.8683195.
- [73] R. Horodecki, Paweł Horodecki, Michał Horodecki, and Karol Horodecki. Quantum entanglement. *Reviews of modern physics*, 81(2):865, 2009.
- [74] Anupama Unnikrishnan and Damian Markham. Authenticated teleportation and verification in a noisy network. *Phys. Rev. A*, 102:042401, 2020.
- [75] R. Van Meter, K. Nemoto, W.J. Munro, and K.M. Itoh. Distributed arithmetic on a quantum multicomputer. In *33rd International Symposium on Computer Architecture (ISCA'06)*, pages 354–365, 2006.
- [76] Charles H Bennett, Gilles Brassard, Sandu Popescu, et al. Purification of noisy entanglement and faithful teleportation via noisy channels. *Phys. Rev. Lett.*, 76(5):722, 1996.
- [77] C. H. Bennett, David P DiVincenzo, John A Smolin, and William K Wootters. Mixed-state entanglement and quantum error correction. *Physical Review A*, 54(5):3824, 1996.
- [78] Wolfgang Dür and Hans J Briegel. Entanglement purification and quantum error correction. *Reports on Progress in Physics*, 70(8):1381, 2007.
- [79] L. Ruan, Wenhan Dai, and Moe Z Win. Adaptive recurrence quantum entanglement distillation for two-kraus-operator channels. *Physical Review A*, 97(5):052332, 2018.
- [80] Filip Rozpędek, Thomas Schiet, David Elkouss, et al. Optimizing practical entanglement distillation. *Physical Review A*, 97(6):062333, 2018.
- [81] L. Ruan, Brian T Kirby, Michael Brodsky, and Moe Z Win. Efficient entanglement distillation for quantum channels with polarization mode dispersion. *Physical Review A*, 103(3):032425, 2021.
- [82] M. Soeken, G. Meuli, B. Schmitt, et al. Boolean satisfiability in quantum compilation. *Phil. Trans. Royal Soc. A*, 378(2164):1–16, 2019. doi: 10.1098/rsta.2019.0161.
- [83] C. Chamberland, G. Zhu, T. J. Yoder, et al. Topological and Subsystem Codes on Low-Degree Graphs with Flag Qubits. *Physical Review X*, 10(011022), 2020.
- [84] IBM Q. Transpiler. <https://qiskit.org/documentation/apidoc/transpiler.html>, 2022.
- [85] Pablo Andrés-Martínez and Chris Heunen. Automated distribution of quantum circuits via hypergraph partitioning. *Phys. Rev. A*, 100:032308, Sep 2019. doi: 10.1103/PhysRevA.100.032308.
- [86] Ranjani G. Sundaram, Himanshu Gupta, and C. R. Ramakrishnan. Efficient Distribution of Quantum Circuits. In *35th International Symposium on Distributed Computing (DISC 2021)*, 2021.
- [87] R. G. Sundaram, H. Gupta, and C. R. Ramakrishnan. Distribution of Quantum Circuits Over General Quantum Networks. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 415–425, 2022.
- [88] Omid Daei, Keivan Navi, and Mariam Zomorodi-Moghadam. Optimized quantum circuit partitioning. *Int J Theor Phys*, 59(12):3804–3820, December 2020. ISSN 1572-9575. doi: 10.1007/s10773-020-04633-8.
- [89] Z. Davarzani, M. Zomorodi-Moghadam, M. Houshmand, and M. Nouri-baygi. A dynamic programming approach for distributing quantum circuits by bipartite graphs. *Quantum Information Processing*, 19, 2020. doi: 10.1007/s11128-020-02871-7.
- [90] Eesa Nikahd, Naser Mohammadzadeh, Mehdi Sedighi, and Morteza Saheb Zamani. Automated window-based partitioning of quantum circuits. *Phys. Scr.*, 96(3):035102, January 2021. ISSN 1402-4896. doi: 10.1088/1402-4896/abd57c.
- [91] Daniele Cuomo, Marcello Caleffi, Kevin Krsulich, Filippo Tramonto, Gabriele Agliardi, Enrico Prati, and Angela Sara Cacciapuoti. Optimized compiler for distributed quantum computing, 2021.
- [92] Davood Dadkhah, Mariam Zomorodi, and Seyed Ebrahim Hosseini. A New Approach for Optimization of Distributed Quantum Circuits. *International Journal of Theoretical Physics*, 60(9):3271–3285, September 2021. ISSN 0020-7748, 1572-9575. doi: 10.1007/s10773-021-04904-y.
- [93] A. G. Fowler, S. J. Devitt, and L. C. L. Hollenberg. Implementation of Shor’s algorithm on a linear nearest neighbor qubit array. *Quantum Information and Computation*, 4:237–251, 2004. doi: 10.26421/QIC4.4.
- [94] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307, 1970. doi: 10.1002/j.1538-7305.1970.tb01770.x.

- [95] Prakash Murali, Jonathan M. Baker, Ali Javadi Abhari, et al. Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers. *arXiv e-prints*, 2019. arXiv:1901.11054.
- [96] Shin Nishio, Yulu Pan, Takahiko Satoh, et al. Extracting success from ibm's 20-qubit machines using error-aware compilation. *J. Emerg. Technol. Comput. Syst.*, 16(3), may 2020.
- [97] Siyuan Niu, Adrien Suau, Gabriel Staffelbach, and Aida Todri-Sanial. A hardware-aware heuristic for the qubit mapping problem in the nisq era. *IEEE Transactions on Quantum Engineering*, 1:1–14, 2020.
- [98] Davide Ferrari and Michele Amoretti. Noise-adaptive quantum compilation strategies evaluated with application-motivated benchmarks. In *Proceedings of the 19th ACM International Conference on Computing Frontiers*, CF '22, page 237–243, 2022. doi: 10.1145/3528416.3530250.
- [99] Davide Ferrari, Saverio Nasturzio, and Michele Amoretti. A software tool for mapping and executing distributed quantum computations on a network simulator, 2021. URL <https://2021.qcrypt.net/speakers/#list-of-accepted-posters>.
- [100] Junchao Wang, Guoping Guo, and Zheng Shan. Sok: Benchmarking the performance of a quantum computer. *Entropy*, 24(10), 2022. doi: 10.3390/e24101467.
- [101] Ben Bartlett. A distributed simulation framework for quantum networks and channels. *arXiv e-prints*, 2018. arXiv:1808.07047.
- [102] Tim Coopmans, Robert Knegjens, Axel Dahlberg, et al. NetSquid, a NETwork Simulator for QUantum Information using Discrete events. *Communications Physics*, 4(1):164, December 2021.
- [103] Axel Dahlberg and Stephanie Wehner. SimulaQron - a simulator for developing quantum internet software. *Quantum Science and Technology*, 4(1):015001, Sep 2018.
- [104] Xiaoliang Wu, Alexander Kolar, Joaquin Chung, et al. Sequence: a customizable discrete-event simulator of quantum networks. *Quantum Science and Technology*, 6(4):045027, 2021.
- [105] T. Matsuo. Simulation of a Dynamic, RuleSet-based Quantum Network. *arXiv e-prints*, 2021. arXiv:1908.10758.
- [106] Stephen Diadamo, Janis Notzel, Benjamin Zanger, and Mehmet Mert Bese. QuNetSim: A Software Framework for Quantum Networks. *IEEE Transactions on Quantum Engineering*, 2:1–12, 2021.
- [107] Axel Dahlberg, Bart van der Vecht, Carlo Delle Donne, et al. Netqasm - a low-level instruction set architecture for hybrid quantum-classical programs in a quantum internet. *Quantum Science and Technology*, 7(3):035023, jun 2022.
- [108] QuTech. Quantum Network Explorer ADK, 2022. URL <https://github.com/QuTech-Delft/qne-adk>.
- [109] Chin-Te Liao, Sima Bahrani, Francisco Ferreira da Silva, and Elham Kashefi. Benchmarking of quantum protocols. *Scientific Reports*, 12(1):5298, March 2022. ISSN 2045-2322. doi: 10.1038/s41598-022-08901-x.
- [110] Miraleme Mehic, Marcin Niemiec, Stefan Rass, et al. Quantum key distribution: A networking perspective. *ACM Comput. Surv.*, 53(5), sep 2020.
- [111] Antonio Manzalini and Michele Amoretti. End-to-end entanglement generation strategies: Capacity bounds and impact on quantum key distribution. *Quantum Reports*, 4(3):251–263, 2022.
- [112] Claudio Cicconetti, Marco Conti, and Andrea Passarella. Request scheduling in quantum networks. *IEEE Transactions on Quantum Engineering*, 2:2–17, 2021.
- [113] Various Authors. Quantum Protocol Zoo, 2022. URL <https://wiki.veriqcloud.fr/index.php>.
- [114] S.J. Devitt, W.J. Munro, and K. Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7), 2013.
- [115] Damian S. Steiger, Thomas Häner, and Matthias Troyer. ProjectQ: An Open Source Software Framework for Quantum Computing. *Quantum*, 2:49, January 2018. ISSN 2521-327X. doi: 10.22331/q-2018-01-31-49.
- [116] S. Zanger, B. and DiAdamo. EQSN: Effective Quantum Simulator for Networks, 2020. URL https://github.com/tqsd/EQSN_python.
- [117] Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta. Open quantum assembly language. *arXiv e-prints*, July 2017. arXiv:1707.03429.
- [118] P. Magnard, S. Storz, P. Kurpiers, et al. Microwave quantum link between superconducting circuits housed in spatially separated cryogenic systems. *Phys. Rev. Lett.*, 125:260502, Dec 2020.