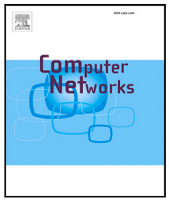
Computer Networks 254 (2024) 110672

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Distributed quantum computing: A survey

Marcello Caleffi [a,b,*], Michele Amoretti [c], Davide Ferrari [c], Jessica Illiano [a], Antonio Manzalini [d], Angela Sara Cacciapuoti [a,b]

[a] FLY: Future Communications Laboratory, Department of Electrical Engineering and Information Technology (DIETI), University of Naples Federico II, Naples, 80125, Italy[1]
[b] Laboratorio Nazionale di Comunicazioni Multimediali, National Inter-University Consortium for Telecommunications (CNIT), Naples, 80126, Italy
[c] QSLab: Quantum Software Laboratory, Department of Engineering and Architecture (DIA), University of Parma, Parma, 43124, Italy [2]
[d] TIM, Turin, 10148, Italy

ARTICLE INFO

Keywords:
Quantum internet
Quantum networks
Quantum communications
Quantum computing
Quantum computation
Distributed quantum computing
Quantum algorithms
Quantum compiler
Quantum compiling
Simulator

ABSTRACT

Nowadays, quantum computing has reached the engineering phase, with fully-functional quantum processors integrating hundreds of noisy qubits. Yet – to fully unveil the potential of quantum computing out of the labs into the business reality – the challenge ahead is to substantially scale the qubit number, reaching orders of magnitude exceeding thousands of fault-tolerant qubits. To this aim, the *distributed quantum computing* paradigm is recognized as the key solution for scaling the number of qubits. Indeed, accordingly to such a paradigm, multiple small-to-moderate-scale quantum processors communicate and cooperate for executing computational tasks exceeding the computational power of single processing devices. 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 from a computer and communications engineering perspective. Furthermore, this survey provides an easy access and guide towards the relevant literature and the prominent results in the field.

1. Introduction

Quantum computing has finally reached the engineering phase, with fully-functional quantum processors integrating hundreds of noisy qubits [1,2]. And it has the potential to completely change markets and industries, since a quantum computer can, in principle, tackle classes of problems that choke classical machines [3,4].

However, to fully unlock the potentialities of quantum computing, thousands of fault-tolerant interconnected qubits are required [1]. And quantum technologies are still far away from this ambitious goal, since there still exist hard technological limitations on the number of qubits that can be embedded in a single quantum chip [5]. Indeed, we are in the noisy intermediate-scale quantum (NISQ) processors age [6–8].

In this context, the consensus of both academic and industry communities for realizing large-scale quantum processors is to adopt the *distributed quantum computing* (DQC) paradigm, which relays on a quantum network infrastructure for clustering together modular and small quantum chips in order to scale the number of qubits [3,4,9–11].

Indeed, accordingly to the DQC paradigm, individual quantum processors, limited in the number of qubits, work together to solve computational tasks exceeding the computational power of single processing devices [3,5,12–16]. And, differently from distributed classical computing, a linear increase in the number of interconnected quantum processors unlocks an exponential increase of the quantum computing power [3,4,12,16].

DQC architectures are expected to be realized, in a very near future, in the form of local quantum server farms [4,10] whereas, on a longer time-horizon, geographically-distributed server farms are envisioned to be interconnected around the globe [10,16]. Indeed, Rigetti already developed high fidelity, low-latency quantum interconnects between modules, providing technological foundations for modular quantum computers [17]. 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 [18]. Metropolitan-area and wide-area quantum networks are also under research and development [19–23], which

* Corresponding author at: FLY: Future Communications Laboratory, Department of Electrical Engineering and Information Technology (DIETI), University of Naples Federico II, Naples, 80125, Italy.
E-mail addresses: marcello.caleffi@unina.it (M. Caleffi), michele.amoretti@unipr.it (M. Amoretti), davide.ferrari1@unipr.it (D. Ferrari), jessica.illiano@unina.it (J. Illiano), antonio.manzalini@telecomitalia.it (A. Manzalini), angelasara.cacciapuoti@unina.it (A.S. Cacciapuoti).
[1] Web: www.quantuminternet.it
[2] Web: www.qis.unipr.it/quantumsoftware

https://doi.org/10.1016/j.comnet.2024.110672
Received 14 May 2024; Received in revised form 28 June 2024; Accepted 22 July 2024
Available online 8 August 2024

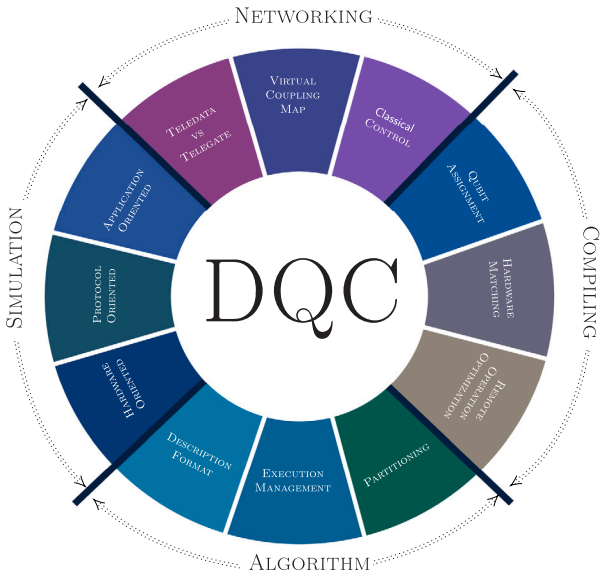


Fig. 1. The Distributed Quantum Computing ecosystem, represented by highlighting the four different pillars overviewed within the survey: *Algorithms*, *Compiling*, *Networking* and *Simulation*.

would enable DQC among geographically-distributed quantum farms and/or devices.

Unfortunately, the existing literature on DQC 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 astonishing and intriguing properties of distributed quantum computing, with the objective of allowing the reader:

(i) to own the implications of the novel and distinct characteristics of quantum information, for understanding the differences between distributed classical computing and 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.

Due to the fast growth of this research field, such an understanding serves the computer science and the communications engineering communities to have an easy access and guide towards the relevant literature and the prominent results, which is of paramount importance for advancing the state-of-the-art.

The survey provides perspectives – including state-of-the-art and challenges – on four different area related to DQC, namely: *algorithms*, *networking*, *compiling*, and *simulation*, as detailed in Section 1.1.

1.1. Outline

As illustrated in Fig. 1, for each of the aforementioned four pillars, the most relevant aspects are analyzed and discussed.

A quantum network infrastructure is a fundamental pre-requisite for any form of DQC. Therefore, through the survey we shed the light on the communication primitives required for inter-networking different quantum processors. We discuss the main challenges arising with this inter-networking, by introducing the reader to the fundamental differences between interconnecting remote *classical* processors versus interconnecting remote *quantum* processors. Regarding algorithms, the focus is on the crucial and specific challenges arising when moving from monolithic to distributed quantum computing, ranging from appropriate description formats through quantum algorithm partitioning to execution management. Quantum Compiling, instead, deals with translating a hardware-agnostic description of the algorithm into a functionally equivalent description that takes into account the physical constraints of the underlying computing architecture [13,15]. Indeed, within the context of DQC, the compiling must account also for the network constraints, which impact on the strategy adopted for splitting the algorithm into "portions" to be concurrently executed on the individual

Fig. 2. Paper Structure.

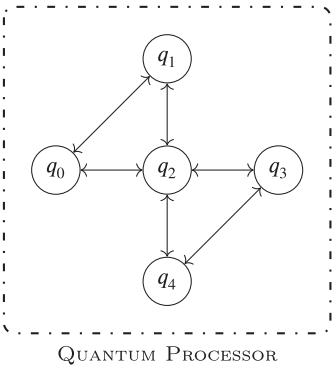


Fig. 3. Coupling map of a superconducting quantum processor [24,25]. The five physical qubits stored within the processor are represented by circles. The arrows denote the possibility to realize a two-qubit CNOT gate between the five qubits. As an example, a CNOT between qubits q_0 and q_1 can be directly executed by the quantum processor, whereas a CNOT between qubits q_0 and q_2 cannot.

quantum processing units (QPUs).[3] 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.

The paper is structured as depicted in Fig. 2. Specifically, in Section 2, we introduce some preliminaries by highlighting the differences between monolithic and distributed computing. Then, in Section 3, we provide the reader with the networking functionalities required by the distributed computing paradigm, by detailing the pivotal role played by communication infrastructure to enable distributed quantum computing. In Section 4, we focus on quantum algorithms as well as their execution management in the light of the distributed paradigm. In Section 5, we describe some relevant approaches to the problem of compiling quantum algorithms for distributed execution. In Section 6, we provide an overview of the most advanced simulation tools, by discussing their suitability for the design and analysis of distributed quantum computing architectures. In Section 7, we discuss the open issues and the research directions for each of the four pillars of this survey. Finally, we conclude our survey in Section 8, by first providing a discussion about the main differences between distributed classical and quantum computing, and then by providing an industrial perspective on DQC.

2. Distributed quantum computing

The purpose of this section is to briefly introduce the main differences between monolithic and distributed quantum computing. To this aim, intra-chip connectivity is discussed first, as it plays a key role in both the two paradigms.

2.1. Monolithic quantum computing

Monolithic Quantum Computing refers to the execution of a quantum algorithm on a single quantum processor.

As briefly described in the box enclosed in the next page, a quantum algorithm is commonly modeled by a quantum circuit. Quantum circuits are made of quantum gates and, in general, the set of gates that can be executed on a certain quantum processor is finite, i.e., constituted by few quantum gates, as a consequence of the constraints imposed by the underlying qubit technology [26–45]. Thus, any gate outside the aforementioned *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, as mentioned before. And any logical gate that can be run on current IBM quantum platform is built from a gate set composed by the CNOT gate and four single-qubit gates, namely, I, $\mathtt{R}_z$, SX, and X gate.

It must be observed that a discrete set of gates cannot be used to implement any arbitrary unitary operation exactly, since the set of unitary operations is continuous [46]. In other words, for any finite set of gates there exist unitary transformations that cannot be realized as a combination of these gates. However, there exist finite sets of gates – referred to as *universal* gate sets – that can approximate any unitary transformation to arbitrary accuracy [47]. And indeed, for any level of accuracy, this approximation can be done efficiently accordingly to the Solovay–Kitaev theorem [46,47].

From the above, one can safely assume that a quantum algorithm can be executed on a given quantum processor, either in the form described by the original quantum circuit or by properly replacing unavailable gates with equivalent sequences of the available ones. To do this, in the context of monolithic quantum computing, it is necessary that the number of physical qubits within the processor is at least equal to the *circuit width*, i.e., to the number of logical qubits. In fact, each logical qubit within the quantum circuit must be assigned to a physical qubit of the quantum processor.

However, in the NISQ age, a logical qubit has to be usually mapped onto several physical qubits for implementing proper fault-tolerant techniques [48]. Hence, the number of physical qubits available in a single processor may be not sufficient to execute the quantum algorithm. As a consequence, the consensus of both academic and industry communities for realizing large-scale quantum processors is to adopt the DQC paradigm, discussed below.

2.2. Archetypes for distributed quantum computing

In *Distributed Quantum Computing*, quantum processors, limited in the number of qubits, work together for solving the computation associated with a quantum algorithm. Hence and as illustrated with the toy model in Fig. 4, a distributed quantum computation involves non-local gates, i.e., it involves operations between qubits belonging to different processors. Also (classical) distributed computing involves operations between bits stored at different processors. And these non-local operations are executed through data replications, namely, by simply copying and sending the bits from one processor to another. So one might be tempted to believe that the same strategy can be adopted when it comes to DQC. Unfortunately, this is not true due to the laws of quantum mechanics, such as quantum measurement postulate and non-cloning theorem. Hence DQC requires a paradigm shift for dealing with inter-processor communications, as deeply discussed in the remaining part of the manuscript.

Regardless of the challenges connected to DQC, one must notice that DQC can be realized according to different archetypes, related to the development of the underlying network infrastructure and maturity of the quantum technologies, as represented in Fig. 5. In the following we introduce these archetypes, while in Section 7 we describe the challenges and open problems connected to them.

2.2.1. Multi-core quantum architectures

The first archetype for DQC is the one exploiting the interconnection of multiple QPUs within a single quantum computer. This results in an architecture known as *multi-chip* [49] or *multi-core* [50,51]. 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 requirements between qubits devoted to store quantum states, referred to as *memory qubits*, and qubits devoted to computational/processing

[3] Throughout the manuscript, the two terms quantum processor and quantum processing unit are used as synonyms.

Intra-chip Connectivity

Quantum computing requires quantum states to be manipulated not only via single-qubit gates, but also through multi-qubit gates, mainly two-qubit gates such as CNOT or CZ gates[a]. This implies that, within a quantum chip, physical qubits must be able to interact in a controlled fashion. Indeed, the underlying quantum technology influences the interaction features of the physical qubits. Specifically, there exists a large class of different quantum computing platforms where two-qubit gates cannot be applied to any physical qubit pair of a quantum processor, but they are instead restricted to certain pairs. These limitations arise as a consequence of both the: (i) noise effects induced by qubit-interactions, and (ii) physical-space constraints within a single processor [13]. In this class of quantum computing platform, the quantum devices are characterized by a *coupling graph* that specifies which qubits may interact. More into details, in the coupling graph[b], vertices denote qubits and arrows denote the possibility of realizing a two-qubit gate between the connected qubits – as illustrated in Fig. 3. Among the technologies in the aforementioned category, we can mention superconducting qubits, utilized for example by Google [26], IBM [27], Rigetti [28], Alice & Bob [29], Anyon [30], IQ [31], and OQC [32]. But also quantum dots – utilized for example by Intel [33], C12 [34], and Quobly [35] – and color-center qubits – such as NV centers in diamonds utilized by Quantum Brilliance [36] – belong to this category.

From the above, it becomes clear that any quantum computation executed on a quantum processor belonging to this category requires that each multi-qubit operation between non-adjacent (within the coupling graph) physical qubits is mapped into a sequence of operations between adjacent physical qubits [13]. This mapping process, known as *quantum compilation* and described in details in Section 5, must be properly optimized so that the overhead for satisfying all the constraints imposed by the coupling graph is minimized.

Quantum compilation must be performed regardless of the adopted quantum computing paradigm – i.e., monolithic vs distributed – since in both the paradigms multi-qubit operations are restricted to act only on adjacent physical qubits. However, the overall optimization process underlying quantum compilation is more challenging in DQC, since multi-qubit operations can involve qubits belonging to different chips.

Another category of quantum technologies, instead, does not constraint the interactions among qubits, i.e., their coupling graph is fully connected. It is the case of quantum devices based on trapped ions – such as the ones developed by AQT [37], IonQ [38], Quantinuum [39], and Oxford Ionics [40] – or neutral atoms, such as the ones developed by PASQAL [41], QuEra [42], Atom Computing [43], and Infleqtion [44]. Complete connectivity among physical qubits constitutes an advantage over the partially connected topologies exhibited by the first category of qubit technologies, since extra gates are not needed for moving quantum states to nearest-neighbor qubits. Yet, these technologies, although they can operate at room temperature, exhibit different weaknesses [2]. Regarding trapped ions, scaling the number of particles to large numbers is challenging. With neutral atoms, the repetition rate – i.e., the "computing clock" – is currently lower than other platforms, mainly limited by the time required for the preparation of the qubit array and a destructive readout. Furthermore, the interfacing with classical electronic hardware is generally more complex when compared with other technologies.

[a] See the box in the following page for a concise description of controlled gates such as CNOT or CZ.

[b] We highlight that, in literature, coupling graphs are also referred to as coupling maps. And in the remaining part of the manuscript we adopt such a widely-adopted terminology.

tasks. Indeed, memory qubits likely require coherence times several order of magnitude larger than computational qubits.

Multi-core quantum architectures are also called *Quantum Networks-on-Chip* (QNoC). The rationale for this naming is to highlight that some sort of chip-scale network is employed for interconnecting different quantum computing modules [52–55]. In a QNoC architecture, the physical distance between *remote qubits* – i.e., qubits belonging to different computing modules – is very short, ranging from same-rack/same-refrigerator to same-optical table distances. Accordingly, the degrees of freedom in scaling the number of cores to be interconnected within such a space are lower than the degrees available in Multi-Computer architectures, as analyzed in the next subsection. This, in turn, implies that the number of physical qubits that can be clustered together in a Multi-Core quantum architecture [48] is limited as well.

We further observe that this DQC archetype is less demanding than the other two archetypes in terms of maturity of the underlying quantum computing technologies. In fact, several issues characterizing the other two archetypes – such as quantum transduction and medium-to-long-range quantum communications – are not present in the multi-core architecture. This, in turn, implies that multi-core DQC architectures are also the most investigated in literature, whereas the state-of-the-art of the other architectures is still at its infancy.

2.2.2. Multi-computer quantum architectures

In the second DQC archetype, the distributed computation is performed collectively by multiple quantum computers located within the same farm and interconnected via some sort of Quantum LAN (QLAN) [56]. Thus, some sort of hardware heterogeneity might arise, given that different quantum computers are involved in the computation. Such a heterogeneity must be taken into consideration by the distributed quantum computing ecosystem [12,16], as analyzed in Section 7. As represented in Fig. 5, the physical distance between remote qubits in a multi-computer quantum architecture increases with respect to the multi-core architectures, since the qubits that may interact could belong to different computers. Accordingly, the distances among remote qubits are between room-wide and building-wide, and the number of physical qubits that can be clustered together is bounded by the number of computers that can be interconnected within a server farm.

From the above, it is evident that, although more demanding in terms of quantum technologies maturity than QNoC, this type of DQC archetype provides more degree of freedoms in optimizing the distributed computation.

2.2.3. Multi-farm quantum architectures

In the third archetype of DQC, the distributed computation exploits multiple geographically-distributed quantum farms. Hence, the hardware heterogeneity is significant, given that the different quantum farms are likely operated by different companies. Furthermore, the interconnection of geographically-distributed quantum farms requires a wide-scale network infrastructure, likely achieved through the Quantum Internet [4,16,57].

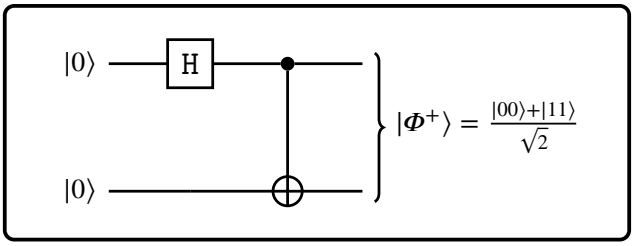
From the above, one can conclude that the main features of this last DQC archetype are the increasing number of quantum devices to be

Quantum Circuit Model

The *quantum circuit* [46] 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* [13]. 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 the figure below, 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 are described by *unitary matrices* relative to some basis, i.e., matrix U such that $U^{\dagger}U = I$. It follows that ideal (or noisy-free) quantum computation is *reversible*: it is always possible to invert a quantum computation.



As a matter of fact, every unitary operator U on a single qubit can be formulated as:

$$U = e^{i\theta_1}\mathrm{R}_x(\theta_2)\mathrm{R}_y(\theta_3)\mathrm{R}_z(\theta_4), \quad \theta_i \in \mathbb{R} \tag{1}$$

with R_i denoting the i-axis rotation operator. 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 1.

Among two-qubit gates, highly relevant are the *controlled* ones. The generic `Controlled-U` two-qubit gate operates on two qubits, namely on a *control qubit* (controlling the operation) and on 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:

$$\begin{cases} \mathtt{I}|\varphi_t\rangle & \text{if } |\varphi_c\rangle = |0\rangle \\ \mathtt{U}|\varphi_t\rangle & \text{if } |\varphi_c\rangle = |1\rangle. \end{cases} \tag{2}$$

with `I` denoting the identity operation. The `CNOT` gate is a `Controlled-U` gate, where `U` is the `Pauli-X` gate. The `CNOT` gate can be used to create or destroy entanglement among the qubits. Specifically, as depicted in the circuit model figure, to obtain an entangled state we may start from the separable input $|00\rangle$ and, by applying `H` to the first qubit, we obtain $(|00\rangle+|10\rangle)/\sqrt{2}$. Finally, by applying a `CNOT` gate (where the first qubit acts as control qubit), the resulting state is the Bell state $|\Psi^+\rangle$:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big). \tag{3}$$

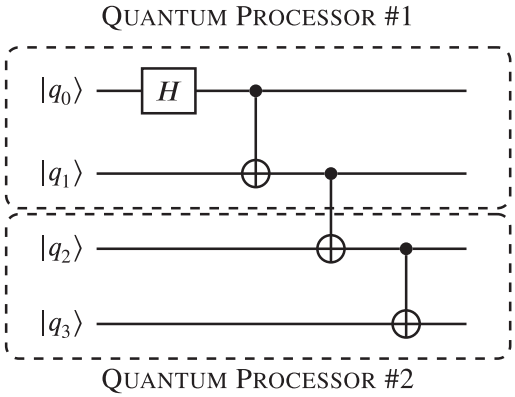


Fig. 4. Toy model for distributed quantum computation. The quantum circuit is composed by three two-qubit gates, i.e, `CNOT`s. First and last gates operate *locally*, namely, between qubits stored within the same QPU, whereas the intermediate gate operates *remotely*, namely, between qubits stored within different QPUs.

wired and the heterogeneity of the environments hosting the quantum computers.

Regardless of the considered DQC archetype, it is worthwhile to mention that networking primitives with no counterpart in the classical world are needed for enabling the data transfer among remote qubits required by a distributed quantum computation. These primitives are introduced and overviewed in the following section.

3. Quantum networking: Enabling remote operations

As mentioned in the previous sections – regardless of the specific DQC archetype – when it comes to distributed quantum computing, qubits are distributed among multiple devices, interconnected by some sort of quantum network infrastructure.

Accordingly, whenever a quantum gate must operate on *remote qubits*, some sort of *communication primitive* must be available for performing inter-processor operations. Unfortunately, this communication primitive cannot be accomplished through classical protocols. Indeed, the physical phenomena underlying quantum communications with no classical counterpart impose a paradigm shift.

To better substantiate the above statement, in Section 3.1 we introduce the communication primitives required by DQC, by also discussing the role played in the DQC by the classical communication infrastructure. Then, in Section 3.2, we present a key strategy – referred to as *entanglement swapping* – for artificially augmenting the connectivity among different quantum processors, by exploiting entangled states.

3.1. Communication primitives

Here, we discuss the two main strategies – namely, *TeleGate* and *TeleData* – for implementing quantum gates between remote qubits.

3.1.1. Direct qubit transmission

Distributed classical computing extensively relies on the possibility of freely duplicating information. But this basic assumption does not hold when it comes to DQC [58,59] accordingly to the *no-cloning* theorem. 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 communication primitives underlying DQC [4]. 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

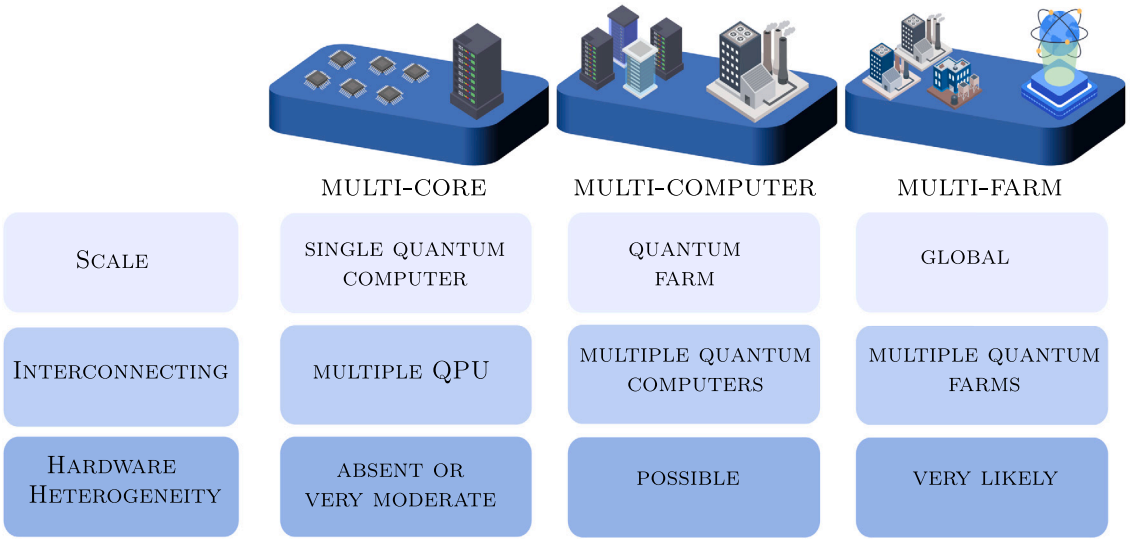


Fig. 5. Archetypes for Distributed Quantum Computing, with three different dimensions – i.e., scale, interconnection and heterogeneity – highlighted for the sake of comparison.

this qubit to a remote processor, e.g., via a fiber link or free space.[4] However, if the traveling photon is lost due to attenuation or it is corrupted by *decoherence*, the associated quantum information cannot be recovered via a measuring process or by re-transmitting a copy of the original information. Specifically, any 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 [60]. 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. As a consequence of the peculiarities of quantum decoherence, the techniques for mitigating the imperfections introduced by qubit transmission cannot be directly borrowed from classical communications [61], and the direct transmission of qubits remains, at the time of writing, limited to special cases characterized by relatively short distances and tolerance to losses and low transmission success rate, such as Quantum Key Distribution (QKD) networks [4]. Clearly, DQC cannot be considered as an application tolerating losses or low transmission success rate, since reliable computations need fault-tolerance to errors.

Thankfully, quantum entanglement [62] can be exploited as the key communication resource to avoid the issues arising with the direct transmission of data qubits. Indeed, entanglement enables a communication technique, known as *quantum teleportation* [4], for transmitting an unknown qubit without the physical transfer of the particle storing the qubit, as described in the following.

3.1.2. TeleData primitive

Whenever two qubits are entangled – as for the Bell state $|\Psi^+\rangle$ given in 3 – they exist in a shared state, such that any action on a qubit affects instantaneously the other qubit as well, regardless of the distance [4,58]. This unconventional correlation is exploited by the *quantum teleportation protocol* [61], which enables the possibility of "transmitting" – namely, *teleporting* – an unknown qubit without the physical transfer of the particle storing the qubit, by exploiting a pair of maximally entangled qubits (such as $|\Psi^+\rangle$) shared between source and destination (see Fig. 6).

More into details, the source performs a pre-processing, namely, a *Bell State Measurement* (BSM) on both the unknown qubit encoding the information to be transmitted – say $|\psi\rangle$ – and the entangled qubit. As represented in the gray box in the figure, the BSM consists of a `CNOT` gate – with $|\psi\rangle$ acting as control and the entangled qubit acting as target – followed by an Hadamard gate on $|\psi\rangle$ 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 1.

Table 1
Quantum teleportation: post processing operations to be performed at the destination for recovering the original quantum state, stemming from the measurement output.

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

Briefly, pre-sharing a maximally-entangled pair of qubits,[5] two nodes can reliably exchange quantum information through the teleportation process [63], by overcoming the limitations of direct data transmissions. Hence, quantum teleportation constitutes the fundamental communication protocol underlying the communication paradigms known as `TeleData` and `TeleGate` [64], which generalize the concept of moving quantum states among remote devices in DQC.

To provide concrete insights on the `TeleData` and `TeleGate` concepts, we must first 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 storing entangled states enabling inter-processor communications. And we refer to these qubits

[4] If the first DQC archetype, i.e., the multi-core one, is analyzed this consideration still holds, but a conversion – aka quantum transduction as analyzed in Section 7 – between quantum states may be un-necessary.

[5] We may observe that direct transmission of qubits is still needed to distribute entangled states among the network nodes. However and as deeply clarified in [58], differently from unknown qubits, entangled states can be repeatedly prepared for facing with losses and/or noise corruptions.

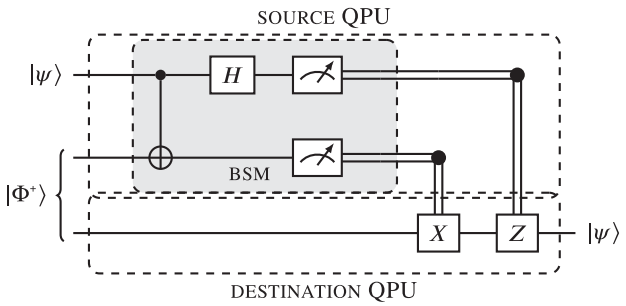


Fig. 6. Circuital representation of the quantum teleportation process. The first two wires belong to the source node, whereas the bottom wire belongs to the destination node. A generic qubit $|\psi\rangle$ is initially stored at the source, and a Bell state such as $|\Phi^+\rangle$ given in 3 must be distributed through a quantum link so that one entangled particle is stored at the source and the other at the destination. Once the Bell state is available, the teleportation is obtained with some processing on $|\psi\rangle$ and on the entangled qubit at the source, followed by two conditional gates on the entangled qubit 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 1, for determining whether the two conditional gates X and Z must be applied to recover the original state $|\psi\rangle$ from the entangled qubit available at the destination.

as communication qubits [57], to distinguish them from the remaining qubits within the device devoted to processing/storage, which we refer either as data or memory qubits as pointed out in Section 2.2. Specifically, at least one qubit at each processor must be a communication qubit, i.e., a qubit reserved for the generation and distribution of the entangled state [57]. The more communication qubits are available within a quantum processor, the more entanglement resources are available at that processor, with an obvious positive effect on the achievable entanglement rate [58]. But the more communication qubits are available, the less data qubits are available for quantum computing.

As instance, let us consider two quantum processors interconnected via a quantum network as depicted in Fig. 7. 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 q_4 in Fig. 7(a) – is teleported into a communication qubit – say q'_0 in the same figure – of the second processor. Once the quantum state 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 Fig. 7(b). 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.

3.1.3. TeleGate primitive

`TeleData` is not the only available option for implementing remote quantum operations in DQC. Indeed, `TeleGate` represents another option, which exploits a variation of the teleportation process to overcome the limitations of direct data transmissions. Specifically, `TeleGate` enables a direct gate between remote physical qubits stored at different processors without the need of moving the data qubits between the processors, as long as a Bell state such as $|\Phi^+\rangle$ is distributed between the two processors. For the sake of exemplification, let us assume that a remote `CNOT` between data qubit q_4 and q'_1 in Fig. 7(a), with q_4 acting as control and q'_1 as target, must be performed. According to the `TeleGate` primitive, this remote `CNOT` can be implemented with two local `CNOT`s 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, as shown with the quantum circuit in Fig. 7(c). As a consequence, `TeleGate` performs the remote operation mimicking a direct gate – i.e., as the involved qubits are directly connected on the same processor – by avoiding to "move" data qubits between the processors.

From a communication resource perspective, `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 to be executed, (ii) the characteristics of the network interconnecting the remote quantum processors, and (iii) the ratio between data and communication qubits [13,58,64].

With reference to the latter factor, a fundamental trade-off arises [13]. 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 operations can be executed in parallel, reducing the communication overhead induced by the distributed computation. But the more are the communication qubits, the less data qubits are available for computing in each processor.

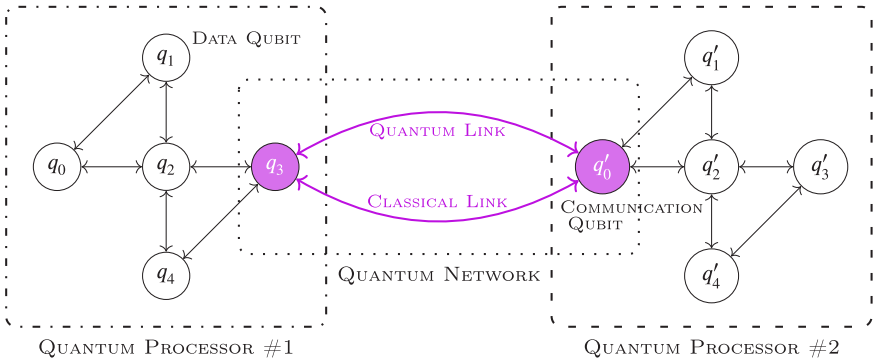
Accordingly to the above reasoning, the selection of the set of communication qubits is a crucial task for DQC, with profound effects on the overall performance of the distributed computation as analyzed in the next sections.

3.1.4. Classical control and communications

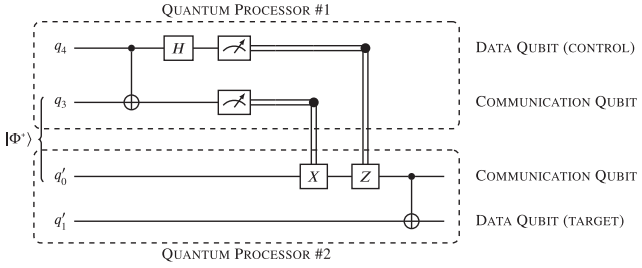
It is worthwhile to mention that other communication primitives required by DQC are the ones provided by the classical network. Specifically, as highlighted above and regardless of the technology and the qubit archetype, DQC depends on the availability of classical communication and network functionalities for managing the classical signaling. As a matter of fact, classical signaling is required by both `TeleData` and `TeleGate`. However, classical signaling is not limited to the 2-bits required by quantum teleportation: it rather constitutes a requirement widespread within different DQC tasks, ranging from entanglement generation through distillation to swapping (discussed in the next section). And, indeed, it is fairly reasonable to assume these classical services as provided by existing classical network infrastructures [59, 65].

3.2. Augmented coupling map

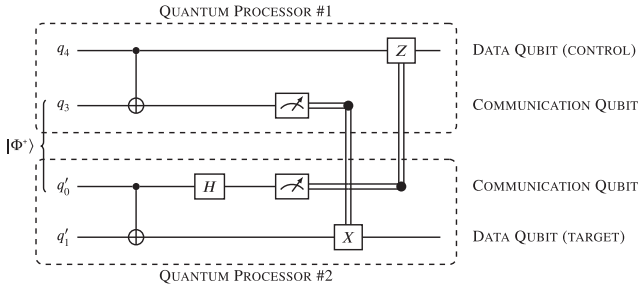
From Fig. 7, it may seem that DQC requires a fully-connected network topology, where each quantum processor must be *directly* inter-connected with all the other processors. This would, in turn, imply that the communication primitives would heavily depend on the availability of a *direct* (one-hop) entanglement generation and distribution architecture. However, the reality is quite the opposite. Specifically, DQC can exploit a strategy known as *entanglement swapping* [45], as summarized in Fig. 8, to implement a remote `CNOT` between qubits stored at remote processors, even if the processors are not directly connected through a quantum link.



(a) Two quantum processors given in Figure 3 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 – purple-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 7a 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 represented by 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 7a can be implemented with two local `CNOT`s 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.

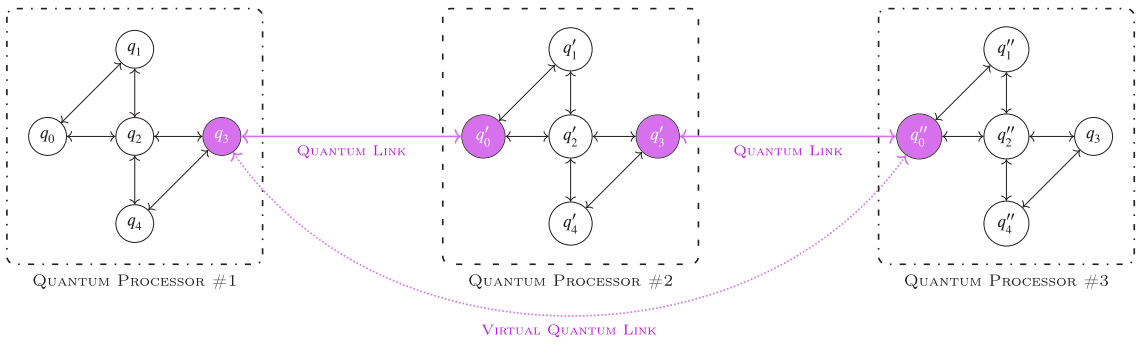
Fig. 7. Remote quantum operations through either `TeleData` or `TeleGate`. Fig. 7(a) shows the network topology along with the processors coupling maps, whereas Figs. 7(b) and 7(c) illustrate the quantum circuit detailing the classical (2 bits) and the quantum (the Bell state) resources needed to execute a `TeleData` and a `TeleGate`, respectively. *Source:* Figure reproduced from [13].

For the sake of exemplification, to distribute a Bell state between remote processors – say quantum processor #1 and #3 in Fig. 8(a) – two Bell states must be first distributed so that one Bell state is shared between the first processor and an intermediate processor/node – usually referred as quantum repeater [66] – 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 Fig. 8(b) – a Bell state is obtained at the remote communication qubits q_3 and q_0'' in Fig. 8(b) – by applying some local processing at the remote processors depending on the (classical) output of the Bell state measurement.

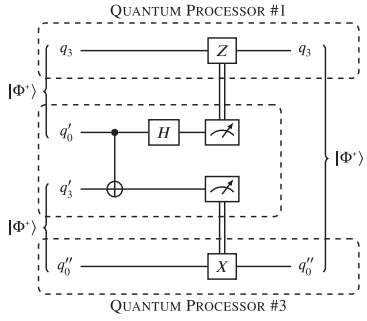
Once remote processors share Bell states, they may operate as they were neighbors in the physical topology [67], since remote operations can be promptly performed. In other words, entanglement enables half-duplex unicast links between any pairs of processors sharing it, regardless of their relative positions within the underlying physical network topology, by redefining so the very same concept of topological neighborhood, with no counterpart in the classical world [67,68]. As a consequence, the entanglement-enabled connectivity allows to augment the neighbor set, by creating "additional" links, referred as artificial quantum links [56,69], toward remote processors.

From the above, it becomes clear that entanglement swapping significantly increases the degrees of freedom in performing remote quantum operations, through the artificial quantum links, by enabling a dynamic coupling map, which includes the physical links as well, as represented in Fig. 8. The higher is the number of available quantum processors, the higher is the number of possible artificial links. Indeed, this number 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 artificial links via entanglement swapping scales more than linearly.

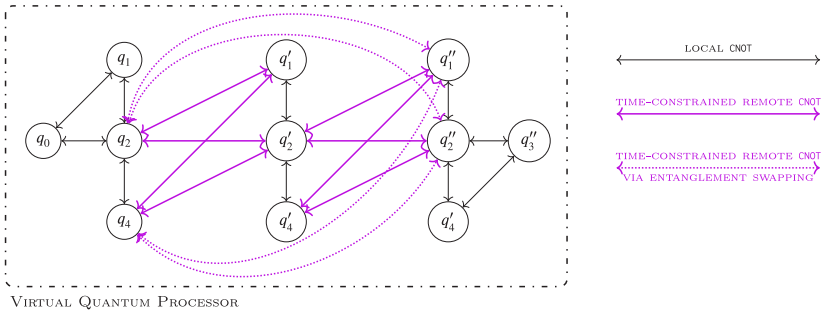
It must be acknowledged that such an augmented connectivity does not come for free. Entanglement swapping consumes Bell states at each intermediate processor. And the longer is the path between the two processors involved in the remote operation, the higher is the number of consumed Bell states. The 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 [13], and the impact of this trade-off on the overall performance of distributed quantum computing must be carefully accounted for.



(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 8a – through local processing and classical communication.



(c) Dynamic coupling map for the network topology shown in Figure 8a. The solid purple lines denote remote CNOTs between adjacent processors, whereas the dotted purple lines denote remote CNOTs between distant processors achievable via entanglement swapping.

Fig. 8. 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.
Source: Figure reproduced from [13].

3.3. More on communication protocols for DQC

From the above subsections, the key role played in DQC by maximally-entangled qubit pairs is evident. However, entanglement is affected by decoherence as well. This, in turn, affects the fidelity of the remote operations. In [70] an extensively review of entanglement purification protocols and quantum error correction is carried out. Indeed, entanglement purification protocols process entangled states in order to improve their fidelity. Among these protocols, *recurrence* entanglement purification protocols, process N disjoint entangled pairs through iterative purification steps, for extracting $M < N$ entangled pairs characterized by higher fidelity [70]. From a communication perspective such protocols exhibit the advantage of being iterative and they easily adapt to different input states. However, they introduce additional delay arising from the need of processing multiple entangled pairs.

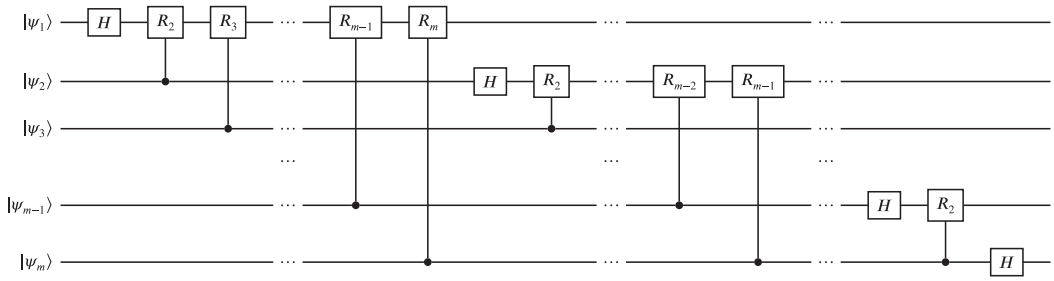
Additional communication protocols in DQC concern the entanglement generation and distribution functionalities. In [71] these functionalities are achieved by leveraging two main protocols referred to as Midpoint Heralding Protocol (MHP) and Quantum Entanglement Generation Protocol (QEGP). Specifically, the MHP protocol acts in a time-slotted environment and is responsible of the generation of entangled pairs at a given time-slot through the activation of dedicated entanglement generation devices. Furthermore, such protocol is able to perform different operations on the generated entangled pairs, such as measurement or storing. Differently, the QEGP protocol is responsible for managing the entanglement requests through a scheduler. Besides, it is in charge of carefully addressing some key needs of the entity exploiting the entanglement such as the quality of the generated states, the number of the entangled pairs and some specifics of the operations to be performed, such as the measurement basis. The inter-operation of the two protocols is achieved through the definition of dedicated messages to be exchanged between the network entities.

Differently, the so-called *entanglement routing* protocols aim at the distribution of entangled pairs between network nodes interconnected by a multi-hop path of physical links. In this regard several contributions have been provided in literature, each aiming at engineering the swapping operations, described in the previous subsection, to be performed at the intermediate nodes in order to optimize some network metrics such as the entanglement throughput or the fidelity of the distributed pairs [72–74].

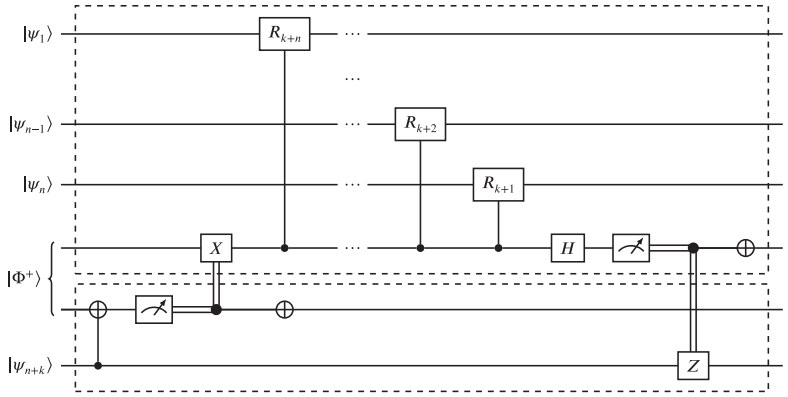
We refer the reader with insights on the challenges related to the design of network protocols and architecture for DQC in Section 7.

4. Quantum algorithms

There exist 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 [75]. Remarkably, most known quantum algorithms use a combination of algorithmic paradigms – actually,



(a) Monolithic QFT circuit. The i-th qubit is obtained through an Hadamard gate followed by $m-i$ controlled $\mathtt{R}_m$ operations – with $\mathtt{R}_i = \mathtt{P}_{2\pi/i}$ denoting the phase gate mapping $|1\rangle \rightarrow e^{j2\pi/i}|1\rangle$ while leaving $|0\rangle$ unchanged – with the controlled operations controlled by the $m-i$ higher-order qubits.



(b) Part of a distributed QFT as shown in [92]. Here, n non-local operations on n qubits are performed using a single shared entangled state. The control is $|\psi_{n+k}\rangle$, the targets are $|\psi_1\rangle, ..., |\psi_n\rangle$. With respect to Figure 9, a subset of the m qubits is considered (i.e., $n+k<m$).

Fig. 9. Quantum Fourier Transform (QFT) circuit compilation for DQC.

sub-routines – specific to quantum computing [76]. These paradigms include the Quantum Fourier Transform (QFT) [77], the Grover Operator (GO) [78], the Harrow/Hassidim/Lloyd (HHL) method for linear systems [79], Variational Quantum Algorithms (VQA) [80], and direct Hamiltonian simulation (SIM). A prominent example is Shor's algorithm for integer factorization [77], which is based on the QFT, illustrated by the quantum circuit in Fig. 9.

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 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 algorithm.[6]

In the following, we discuss three topics that concern quantum algorithms in a DQC context. First, we review the state of the art of description formats for quantum circuits. Second, we discuss the suitability of certain quantum algorithm to be partitioned. Third, we focus on execution management.

4.1. Description formats

As illustrated in the box titled *Quantum Circuit Model*, in the quantum circuit model of computation, quantum states are manipulated by means of quantum gates. Therefore, quantum algorithms correspond to sequential layers of quantum operators applied to the quantum states. Each layer comprises operators that can be executed simultaneously. Long time ago, Knill [82] introduced a few conventions for thinking about and writing quantum pseudocode. Subsequently, several languages have been proposed to describe quantum algorithms in a user-friendly and high-level fashion.

Nowadays, the vast majority of Software Development Kits (SDKs) for quantum algorithm implementation and testing refers to the classical Python language. Major examples are Qiskit [83] by IBM, Cirq [84] by Google, and PennyLane [85] by Xanadu.

Python is very convenient to write software that includes not only quantum circuit descriptions, but also instructions for executing the quantum programs on simulated or real quantum hardware. Sticking to circuit description, a few quantum assembly (QASM) languages have emerged, such as OpenQASM [86,87] and NetQASM [88]. These languages are characterized by a simple, hardware-agnostic but still precise syntax for describing atomic gate-level operations. To facilitate the compilation process, intermediate representations between QASM and hardware-specific control instructions have been designed, such as QSSA [89], QIRO [90] and InQuIR [91].

OpenQASM [86] was proposed as an imperative programming language for quantum circuits based on earlier QASM dialects. Current version 3 encompasses a broader set of circuits beyond the language of qubits and gates, focusing on real-time classical computations that must be performed within the coherence times of the qubits.

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

[6] Factoring $L = 2048$ bit primes – for breaking current RSA implementations – requires about $3L = 6144$ noise-free qubits [46]. It is worth noting that merely increasing the number of physical qubits is not sufficient, as some sort of quantum error correction [81] is also required to guarantee high-quality – namely, noise-free – computations.

send/recv primitives) of multiple nodes is the responsibility of the application programmer.

QSSA [89] is based on static single assignment (SSA), and it models quantum operations as being side-effect-free. The inputs and outputs of the operation are in one-to-one correspondence; qubits cannot be created or destroyed. As a result, QSSA supports a static analysis pass that verifies no-cloning at compile-time. The quantum circuit is fully encoded within the def-use chain of the intermediate representation, allowing the compiler developer to leverage existing optimization passes on SSA representations such as redundancy elimination and dead-code elimination. In practice, QSSA enables decades of research in compiler optimizations to be applied to quantum compilation.

QIRO [90] is an intermediate representation for quantum computing that directly exposes quantum and classical data dependencies for the purpose of optimization. QIRO consists of two dialects, one input dialect and one that is specifically tailored to enable quantum–classical co-optimization. The first dialect employs memory-semantics (quantum operations act on qubits via side-effects), while the second one uses value-semantics (operations consume and produce states) to integrate quantum dataflow in the intermediate representation's SSA graph. This allows for a number of optimizations that leverage dataflow analysis.

Last but not least, InQuIR [91] is a DQC-specialized intermediate representation, allowing the use of classical and quantum communication instructions between different QPUs. InQuIR is provided with a formal semantics that has enough instructions to describe complicated behaviors of distributed quantum programs. In particular, it is able to cope with runtime errors such as qubit memory exhaustion and deadlock in intercommunication between QPUs.

4.2. Partitioning

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, many relevant quantum algorithms are characterized by intricate structures and multi-qubit gates, which move them away from perfect distributability. As an example, let us consider the QFT algorithm, whose circuit is given in Fig. 9(a), notably used as subroutine in many quantum algorithms – e.g., Shor's algorithm and the quantum phase estimation algorithm – as mentioned above. From Fig. 9(a), it is easy to assess that QFT requires each qubit to strongly interact with all the other qubits through controlled $\mathtt{R}_m$ gates [92]. Hence, QFT cannot be considered as perfectly distributable. A portion of the compiled QFT circuit, encompassing two QPUs, is illustrated in Fig. 9(b).

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 5. 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 [93], the authors present two distribution schemes for the *quantum phase estimation* algorithm, they give the resource requirements for both and they show that using less noisy shared entangled states results in a higher overall fidelity. Introduced by Kitaev [94], 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 [77]. The solution proposed in [93] is based on the distributed version of the QFT circuit, obtained by means of non-local controlled U-gates.[7]

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 [65], the authors 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 way.[8] 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. An example of a portion of distributed three-qubits VQA over two QPUs is depicted in Fig. 10.

In [97], 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.

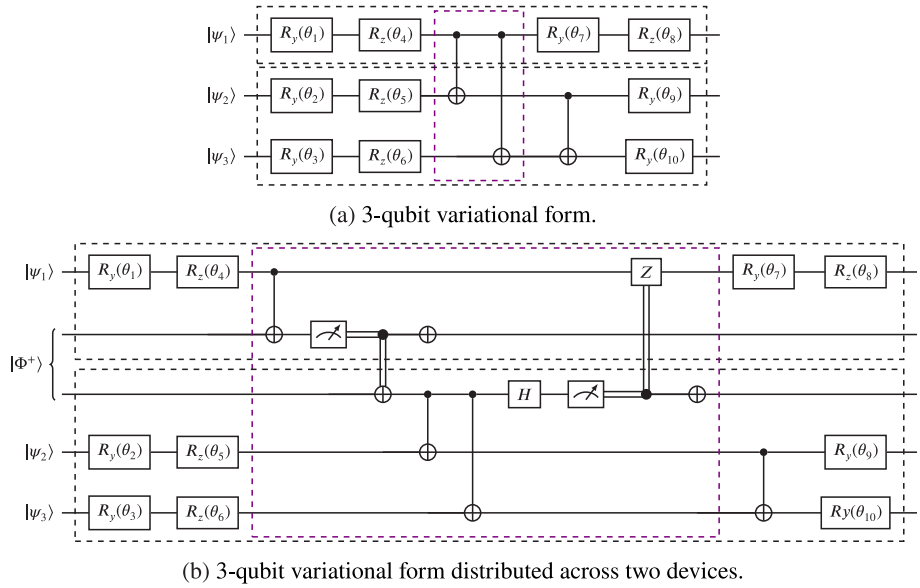
4.3. Execution management

Another relevant aspect is 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 Fig. 11. A QPU scheduling algorithm that partially address this service was proposed in [11]. 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 reasonable to assume that the QPU scheduling plane should be separated from the networking plane, because of the *separation of concerns* principle. This means that entanglement routing must be provided by the network infrastructure to support the execution of the DQC jobs, whose allocation to the QPUs is decided previously. 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 were discussed in Section 3. Here we recall a recent work [98], 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. More recently, the

[7] Non-local controlled U-gate generalizes the TeleGate operation discussed in Section 3.1 to arbitrary unitary U [95].

[8] By using the *cat-entangling* method by Yimsiriwattana et al. [96], which is substantially equivalent to TeleGate introduced in Section 3.1.



(a) 3-qubit variational form.

(b) 3-qubit variational form distributed across two devices.

Fig. 10. Variational Quantum Algorithm (VQA) circuit compilation for DQC.

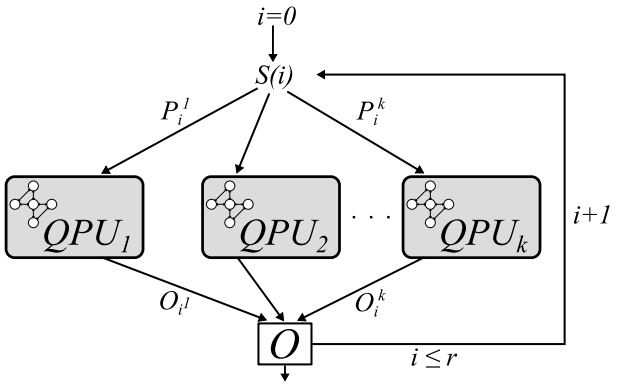


Fig. 11. Execution of multiple quantum circuit instances with k QPUs. For each execution round i, a schedule $S(i)$ maps some quantum circuit instances to the quantum network, with each QPU receiving a quantum circuit P_i^j that is either a monolithic one or a sub-circuit of a monolithic one. The classical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

same authors investigated the issue of service differentiation in the DQC environment [99]. They defined the problem of how to select which computation nodes should participate in each pool, so as to achieve a fair share of the quantum network resources available.

Recently, two frameworks with similar names have been proposed almost at the same time, namely Quantum Network Utility Maximization (QNUM) [100] and Quantum Network Utility (U_{QN}) [101]. While QNUM is specifically tailored to the evaluation of entanglement routing schemes in quantum networks (see Section 3 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 [101] 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 Volume[9] proposed in [102] 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.

In a recent work [16], the authors observed that DQC execution management deals with the *parallel job scheduling* problem, a widely investigated optimization problem in which a set of jobs of varying processing times need to be scheduled on multiple machines while trying to minimize the makespan, i.e., the length of the schedule. Each job has a processing time (in the DQC domain, it can be approximated with the number of layers of computations of the distributed circuit), and requires the simultaneous use of multiple machines. In general, the problem is NP-hard. In [16], two novel metrics are introduced, to the purpose of evaluating QPU utilization and quantum network utilization with different parallel job scheduling strategies. Using two well-known parallel job scheduling algorithms – namely, FIFO and List-Scheduling – it is demonstrated that high QPU utilization may involve also high quantum network utilization. In a classical computing setting, optimal makespan and full resource utilization would be highly appreciated. In DQC, the story is quite different. Indeed, makespan optimality needs highly effective and efficient entanglement routing between QPUs, in order to guarantee timely execution of non-local gates that are all concentrated in a short time frame. The conclusion is that searching for a reasonable tradeoff between QPU utilization and quantum network utilization is crucial.

5. Quantum compiling

For quantum devices characterized by constrained connectivity among qubits, the monolithic execution of a quantum algorithm on a single quantum processor requires a circuit pre-processing known as *quantum compiling* [13,15,103–105]. Specifically, compiling a quantum circuit is a two-step[10] process where:

(i) each logical qubit of the quantum circuit must be mapped onto one (or more, when adopting fault-tolerant techniques [106]) physical qubit of the quantum processor, and

[9] Quantum 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.

[10] With the two steps being inter-dependent, affecting each others.

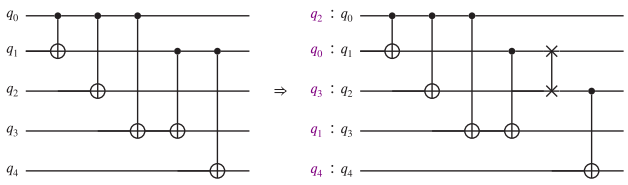


Fig. 12. Pictorial representation of quantum compiling. The circuit on the left is translated into the circuit on the right, in order to cope with the coupling map provided in Fig. 3. Within the rightest figure, the q_i with purple font denotes the physical qubits assigned to the logical qubits q_j with 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.

(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 Fig. 12.

The overall process must be optimized to account for the key performance metrics affecting quantum computation [107–109]. 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.

An example of quantum compilation is provided with Fig. 12, where the original quantum circuit is translated into the compiled one to account for the coupling characteristics of the quantum processor shown in Fig. 3. Indeed, 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, 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. 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 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 3. 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 [103,110], 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 the Intra-chip connectivity box, the actual connectivity degree is usually low, to mitigate the noise caused by cross-talking phenomena [111]. *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 Fig. 13), then find the best split for the abstract circuit, i.e., the split that minimizes the overall communication cost required to execute the distributed 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 [112] 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 2, 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.

5.1. Hardware matching

A fundamental step in every quantum compiler is translating general-purpose quantum gates instructions into instructions specific to the underlying quantum hardware. This translation can go down to the level of analog signals for the control hardware [125–127] or remain at the same abstraction level of the input instructions albeit using the specific gate set supported by the target quantum computer. In this section, we focus on the latter compilation case, which is commonly denoted as *transpiling*.

Depending on the technology used to manufacture a quantum computer, the set of natively supported gates that can be executed varies. Even inside the same "family" of quantum devices, the native gate set may vary. This is the case, for example, of superconducting qubits. Among IBM quantum computers, there are three types of superconducting devices, each supporting a different flavor of single-qubit rotations and two-qubit gates. Furthermore, Google's superconducting devices have a different gate set. Regarding ions trap devices, the native two-qubit gate is the RXX gate, meaning that a CNOT must be decomposed into multiple single qubit rotations and an RXX gate [128]. Finally, in NV centers [88], the only two-qubit gate available is the CNOT, and it can only be applied between the central electron and the surrounding carbon atoms.

To support different flavors of native gate sets, the compiler usually employs a collection of decomposition rules to translate each non native gate into a sequence of native gates. This procedure usually produces a more complex circuit than the input one. Therefore, other circuit optimization techniques may be adopted to reduce the number of native gates used and the depth of the circuit.

5.2. 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.[11] Whether it is better to tackle it dynamically –

[11] One 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 for the sake of simplicity.

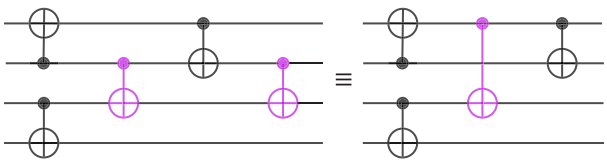


Fig. 13. Example of ebit optimization: the left part of the equivalence can be optimized to the right one, which reduces the number of non-local gates.

Table 2
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.

Compiler	Language	Network Topologies	Qubit Assignment	Non-local Gate Handling	Open Source
[113]	Haskell	hypergraph	minimum k-cut	TeleGate and TeleData	YES
[114]	unknown	hypergraph	minimum k-cut	TeleGate	NO
[115]	unknown	any	Tabu search	TeleGate and TeleData	NO
[116]	MATLAB	/	heuristic	TeleData	NO
[117]	MATLAB	/	dynamic programming	TeleData	NO
[15]	Python	any	minimum k-cut	TeleGate and TeleData	NO
[118]	C++ and CPLEX	n.a.	minimum k-cut	TeleGate and TeleData	NO
[13]	Python	LLN	sorting	TeleGate and TeleData	NO
[119]	pseudo-code	any	integer linear programming	TeleGate	/
[120]	MATLAB	n.a.	genetic alg.	TeleData	NO
[121]	/	any	sorting	TeleData	/
[122]	/	hypercube	sorting	TeleData	/
[123]	Python	any	minimum k-cut	TeleGate	YES
[124]	Python	any	reinforcement learning	TeleGate and TeleData	YES

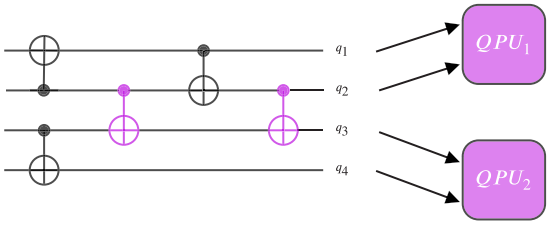


Fig. 14. Toy example of qubit assignment. Once the logical qubits composing the quantum circuit have been assigned to the different QPUs, the CNOTs between remote qubits – highlighted in violet – becomes non-local.

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 subsets of logical qubits to processors, while minimizing the number of required interactions among different sub-sets, as shown in Fig. 14.Several authors investigate this research direction [13,15,113–115]. 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 need 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 [115].

The authors in [113] 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. In [114], the authors 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*[8]. The problem is then solved as a minimum k-cut, where partitions have roughly the same size. In [115], 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 [116], the circuit becomes an undirected graph with qubits as vertices, while edge weights correspond to the number of two-qubit gates between them. In [117], 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. In [15], the authors devise a two step process for the qubit assignment. First, the circuit is translated into a weighted graph and partitioned, using an efficient k-way graph partitioning algorithm, into k (where k equals the number of available QPUs) partitions of roughly equal size. Finally, the authors employ an heuristic algorithm to improve over this initial solution, as equal partitions may not be optimal.

When qubit assignment is dynamic, new challenges – as well as new possibilities – arise. In [118] the authors 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 [13], the authors consider the worst-case scenario of QPUs interconnected through an LNN topology.[12] 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

[12] The Linear Nearest Neighbor (LNN) topology [129] 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.

qubits. In [119], the authors model the compilation problem with an Integer Linear Programming formulation. The formulation is inspired to the vast theory on 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.

5.3. Remote operations optimization

As described in Section 3, 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 3, there are two main approaches for implementing non-local gates, namely TeleData and TeleGate.

The TeleData approach is considered, for example, in [116,117,120–122]. In [121], the authors 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. In [122], the authors 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 $\pi : [1,n] \to [1,n]$. 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. In [116], the authors 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 [130], so that the number of edges between partitions is minimized. Finally, each graph partition is converted to a quantum circuit. In [117], the authors 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, in [120], the authors 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.

Conversely, the TeleGate direction is pursued, for example, in [15,113,114,119,123]. In [113], the authors use cat-entanglement[8] 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 [114], 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 in [119] adopts the TeleGate approach. The authors of [15] use an heuristic approach where the compiler can be set to use only TeleGates or TeleGates and TeleDatas. The results show that the best choice is highly dependent on the type of circuit. The authors in [123] focus on the concept of gate embedding, which make it possible to use the same EPR pair for multiple TeleGates. The authors combine this embedding techniques with different qubit assignment and non-local gate handling techniques from the literature to provide a flexible compilation workflow for heterogeneous quantum networks. Finally, in [124] the authors proposes an MDP formulation that models non-local gate optimization and local compilation on each QPU. Given the prohibitive complexity of such a model, they propose a relaxation of the MDP formulation and tackle the problem with a reinforcement learning approach. The authors show that the RL approach performs well with small random circuits and could be scaled up to bigger circuits by sacrificing some degree of optimality for the solution. Besides [15], this is the only other work considering local compilation.

6. 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 3, 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.

6.1. 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 [131] and NetSquid [132], 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) [131] 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, 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

Table 3
Comparison of simulation tools that can be used for designing and evaluating DQC systems.

Simulation Tool	Language	Multiprocessing	Multithreading	Noise Models	Open Source	Class
SQUANCH [131]	Python	NO	NO	YES	YES	HW
NetSquid [132]	Python	NO	NO	YES	NO	HW
SimulaQron [133]	Python	YES	NO	NO	YES	PR
SeQUeNCe [134]	C++/Python	YES	NO	YES	YES	PR
QuiSP [135]	C++	NO	NO	YES	YES	PR
QuNetSim [136]	Python	NO	YES	NO	YES	PR
NetQASM SDK [88]	C++/Python	NO	YES	YES	YES	AP
QNE-ADK [137]	C++/Python	NO	NO	YES	NO	AP

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 [132] 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 [138], the evaluation of end-to-end entanglement generation strategies in terms of capacity bounds and impact on Quantum Key Distribution (QKD) [139,140], and the performance evaluation of request scheduling algorithms for quantum networks [141].

6.2. 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. [142] – with the possibility to model hardware-agnostic networked quantum processors, with very limited (if not missing) support for noise modeling. Relevant examples are SimulaQron [133], SeQUeNCe [134], QuiSP [135] and QuNetSim [136]. 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 [133] 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 [134] 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 [135] 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 [143]. 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 [135] 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 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 [135].

QuNetSim [136] 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 [133], ProjectQ [144] and EQSN [145] (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.

6.3. 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 [88] relying on NetSquid [132]. 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. [146] 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 [132] 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 [147] language.

NetQASM SDK [88] 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 [132] or SimulaQron [133], 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.

The Quantum Network Explorer Application Development Kit (QNE-ADK) [137] 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 [88]. 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 [132]. 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.

7. Open issues and research directions

In this section we discuss the open issues and research directions related to DQC, by focusing on the four pillars around which this survey is organized on.

7.1. Quantum networking

The open issues and research direction for the quantum networking pillar are first discussed around the three DQC archetypes presented in Section 2, as depicted in Fig. 15. Then, issues and directions crossing a single archetype are gathered at the end of the subsection.

7.1.1. Multi-core quantum architectures

In this type of DQC architecture, the physical distance between remote qubits is very short. Hence, it is reasonable to assume that the underlying communication infrastructure exploits short-range communication links, such as micro-wave links in case of superconducting computing technology. The network topology is likely static, so that only simple quantum network functionalities are required. 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.

7.1.2. Multi-computer quantum architectures

In this type of DQC architecture, as said, the computation is performed collectively by multiple quantum computers located within the same farm. Hence, entanglement distribution still benefits from a tightly controlled environment – reasonable to assume available within a single quantum farm – and the relatively short distances. For the sake of exemplification, the communication infrastructure can still be composed by cold microwave links [148] for superconducting-based qubit technology, although optical links would greatly simplify the hardware requirements albeit at the price of significant technological advances in the microwave-optical conversion.

More into details, an interface – aka quantum transducer – between the processing unit and the inter-computer communication infrastructure is eventually needed, but it represents still an open problem. For instance, super-conducting technologies demand for the so-called *matter-flying interface* [4,149], namely, a device able to convert a qubit belonging to the QPU to a qubit suitable for the transmission over a quantum physical channel [149–151]. In multi-computer architectures, such an interface represents a technology challenge which comprises the major complexity source from a networking perspective.

Delay imposed by classical and quantum communication times is slightly longer, when compared to Multi-Core architectures. 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, in this type of architecture, the execution management problem discussed in Section 4.3, arises, with multiple users performing concurrent access to the resources.

7.1.3. Multi-farm quantum architectures

This last archetype for DQC architectures involves interconnecting multiple geographically-distributed quantum farms. Two are the key challenges here. First, as mentioned in Section 2, there exists a likely spread heterogeneity, which requires significant efforts in terms of standardization and interoperability [57]. Furthermore, the heterogeneity among quantum links – e.g., optical vs terrestrial free-space vs satellite free-space – will arise. And efficient quantum transducers are now mandatory [4,149,151].

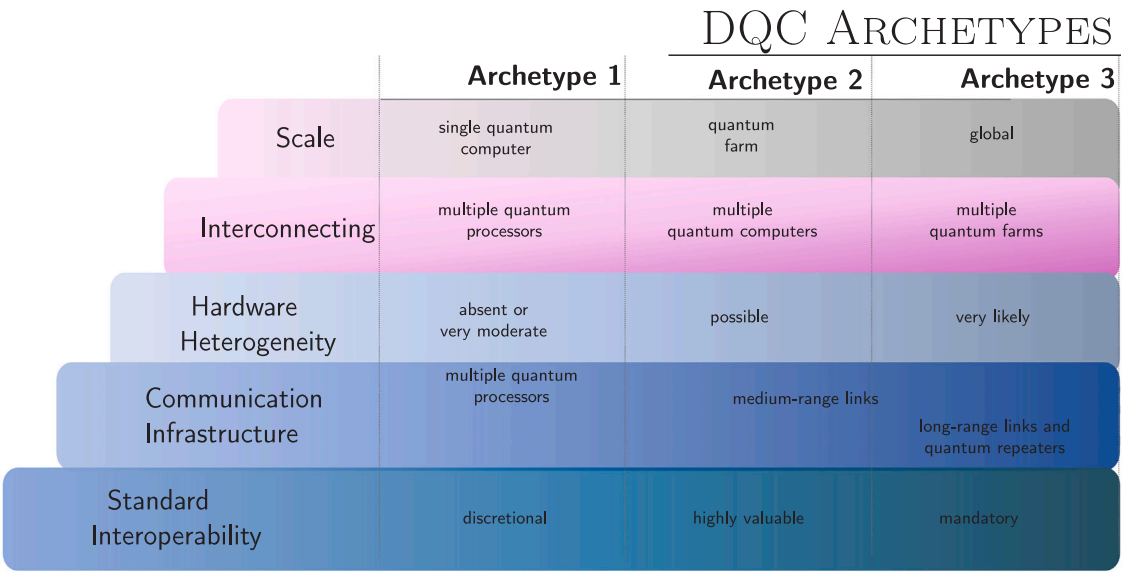


Fig. 15. Networking challenges for distributed quantum computing. It is reasonable to assume that the underlying hardware complexity scales proportional with the three DQC archetypes in terms of: (i) extension of the communication infrastructure, (ii) number of interconnected quantum devices, (iii) and hardware heterogeneity among the quantum devices.

The delays induced by the distances introduce severe challenges on the entanglement generation and distribution. And effective routing techniques are required.

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 are even more complex, demanding for specific network services to be integrated with those of the classical Internet (such as DNS, DHCP, etc.).

We emphasize that, although each type of architecture is characterized by an increasing amount of interconnected quantum resources, the actual deployment evolution of DQC towards the multi-farm architecture is strongly dependent on the technological advances and the experimental implementations of the different entities composing a distributed quantum computing ecosystem [4,12].

7.1.4. Cross-architecture challenges

Another fundamental issue arising with networking remote quantum processors, regardless the specific DQC archetype, is represented by noise and imperfections affecting the *quality* of the distributed entangled states. The noisier is the distributed entangled 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*) [70,152–157]. 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 requiring at least two communication qubits at each processor. 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.

Furthermore, entanglement does not limit to Bell pairs. In fact, multipartite entanglement – i.e., entanglement shared between more than two parties – has the potentiality to be a powerful resource for DQC [69,158–160]. More into details, multipartite entangled states are exploited in the so-called measurement-based quantum computing, which gives rise to a computational model referred to as "*one-way computing*" [161,162]. The one-way computing is different from the circuit model, as it relays on sequences of Pauli measurements on a multipartite entangled state to perform the computation. Remarkably, in such a computing model, the process of generating entanglement is decoupled from the process of consuming it for computation. This, in principle, could be exploited for the design of the network functionalities. Indeed, one can proactively distribute the multipartite resource for one-way computing among the remote processors, and then reactively proceed with the computation, i.e., the Pauli measurements. In this context, from a network perspective, it becomes crucial to efficiently distribute the multipartite entangled resource. Even more importantly, this approach demands for a tight coordination among the networked processors, for the local corrections to be made after the measurements [69,163].

However, multipartite entanglement is still a widely unexplored research area and it is an open research direction to fully unleash its potentiality in DQC.

7.2. Quantum algorithms

Future directions are both theoretical and practical. Despite a considerable amount of work on the fundamentals of distributed quantum computing [121,122,154], 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 [121], 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 parallel job scheduling for 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. Furthermore, to compare different deployments and schedules, DQC-specific key performance indicators must be defined [100,101].

7.3. Quantum compiling

As described in Section 3, the entanglement generation and distribution functionality plays a key role in quantum compiling. Although there exists no standard model conferring to specific entities the responsibility of entanglement generation, distribution and managing, we can identify two possible approaches, namely, network-centric and computing-centric.

In the network-centric approach, the communication infrastructure and its protocols are responsible for the entanglement generation distribution and managing, (seen as both functionality and actuation), while the compiler exploits the entanglement information gathered by the network to perform distributed algorithms. The other approach can be identified as computing-centric. Specifically, the compiler uses as input the physical topology and instructs the quantum communication infrastructure, through requests about the entangled states to be generated and distributed for performing a distributed algorithm.

Although these two approaches are yet to be completely investigated, some general considerations can be made.

In multi-computer and multi-core architectures, it is likely that network operations, such as entanglement generation and distribution, can be engineered and optimized according to the specific algorithm to be performed. In such a case, ad-hoc augmented coupling map can be likely generated in advance (proactively) – prior to the algorithm execution and at compilation time – and the compiler can issue requests to the quantum network before performing the algorithm. This approach entails an additional set-up time to be added to the time interval required for compiling a distributed algorithm. Differently, in large-scale networks as in multi-farm architectures, this can be an hard task due to the large number of computer involved and remote operations.

As a consequence, there exist a deep bi-directional impact between network and compiler design for DQC, which, in turn, is affected by the considered variant of DQC.

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 [105,164–167]. 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. [146] – allowing for automated workflows that would allow for faster comparative evaluation of compiling strategies. Finally, the problem of combining compilation for DQC, i.e., partitioning and non-local gates optimization, with the local compilation of each circuit partition on the QPUs is rarely studied in the literature.

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, 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 [168].

7.4. Quantum simulation

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. Yet, on the other hand, a simulation tool allowing for full-stack simulation of large networks is still missing. Such a tool should be 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. Discussion and future perspective

8.1. Discussion

In order to further highlight the peculiarities of DQC, we summarized in Table 4 the main differences between distributed classical and quantum computing, by focusing on the different archetypes analyzed in Section 2. To this aim, we start from single-core computation.

From an architectural perspective and by oversimplifying, we can identify the classical computing unit as the system comprising: (i) an Arithmetic and Logic Unit (ALU), responsible of performing logic operations; (ii) a hierarchy of memories, L1 cache L2 cache at least; (iii) and a Control Unit (CU). Accordingly, a single-core is identified as a computing system composed by the CU, the ALU and a L1 cache and the single-core paradigm uses such computing unit to process information. Differently, the architectural definition for quantum computing unit is not well-established due to the differences among the underlying technologies. Indeed, there exist no single hardware unit responsible for logic operations on qubits. As instance, superconducting technology requires the physical qubits to be confined within a cryostat and logic operations are implemented through pulses properly shaped to change the electrodynamics and the energy levels of the qubit archetype. Differently, the photonic technology requires an optical table as support and the logic operations are performed through optical components acting on a single photon, which acts as physical qubit archetype [169].

Nevertheless, by performing a strong abstraction effort, in a way resembling of a single-core classical computing unit, a single QPU can be identified as the system comprising: (i) one entity responsible for the quantum information processing; (ii) another entity responsible for storing such information – a quantum memory; (iii) one or more devices in charge of measurement (accessing the information), (iv) and finally a control system which mainly comprises classical ad-hoc resources, both software and hardware, in charge of the management of such entities. It is worthwhile to clarify that a unified architectural model that can be commonly identified as "*quantum single-core*" is still missing.

The step from single-core to multi-core in the classical domain was simple and effective. More into details, a distributed classical multi-core computing architecture comprises a single control unit in charge of multiple cores, which, in turn, share an L2 cache. The cores are usually interconnected via a BUS and placed within the same chip. Hence, for classical multi-core architectures we can clearly identify the shared resources, namely, L2 cache and control unit, alongside with the BUS for accessing the shared cache.

Differently, the evolution from single-core to multi-core architecture is complex in the quantum domain and it appears as deeply technology-dependant. By accounting for the early-stage technology readiness and for the high abstraction used in the concept of QPU, the key observation here lies within the additional hardware and software resources required for the interconnection of multiple QPUs. Specifically,

Table 4
A schematic summary of the differences arising with quantum and classical distributed computing paradigms.

Quantum Domain				
Feature	**Multi-farm Computing**	**Multi-computer Computing**	**Multi-core Computing**	**Single-core Computing**
Architecture	multiple quantum farms interconnected via Quantum Internet	multiple quantum computers interconnected via medium-range communication links	multiple refrigerators or optical tables interconnected via dedicated or short-range communication links	single quantum computing unit
Geographical Distribution	global/ wide area (QWAN)	same room (QLAN)/ same building (QLANs)	same refrigerator / same optical table	
Physical Qubit Scalability	**yes**: exponential scaling of the computing power with linear increasing of interconnected entities	**limited**: number of physical qubits		limited due to crosstalk effects or complex optical setting
Shared Resources	entangled states communication infrastructure			classical control system entanglement generation device measurement and readout system
Resource Sharing	communication infrastructure			dedicated
Programming model	quantum circuit partitioning + TeleData/TeleGate			monolithic execution
Entangled topology coherence	mandatory delegated to classical communications			Not applicable
Hardware Cost	increased as network hardware and infrastructure are required		limited increase as some hardware resources are shared	fixed by technology
Hardware Heterogeneity	very likely	possible and required for some computing unit technologies		absent or very moderate
Interoperability	mandatory	highly suggested		discretional
Standards	mandatory and to be defined	to be defined		
Classical Domain				
Feature	**Distributed Computing**	**Multi-processor Computing**	**Multi-core Computing**	**Single-core Computing**
Architecture	multiple computer connected via network	multiple processors connected via network or BUS	multiple processing cores connected via BUS	Single processing core
Geographical Distribution	Global/WLAN	LAN same rack	limited to single physical location (chip)	
Bit Scalability	horizontally scalable by connecting computers to the network	bounded by the number of processors	bounded by the number of cores	Not applicable
Shared Resources	network resources	BUS,level 3 cache	BUS,level 2 cache, CU	level 1 cache, CU
Resource Sharing	network	system	dedicated	
Programming model	message passing	shared memory	shared memory	
Cache Coherence	Not applicable as no shared cache	fundamental for data consistency and coherence among nodes		Not applicable
Hardware Cost	increased, network requires additional hardware	definitely improved, cost-effective solution		can be improved only at design stage
Hardware Heterogeneity	can be both homogeneous and heterogeneous likely and easily managed	typically no but allowed		no
Interoperability	required	limited		Not applicable
Standards	network standards and protocols	specific parallel programming models and libraries		
Examples	TCP/IP, HTTP- REST, SOAP	OpenMP,MPI,CUDA		

differently from the classical domain, a multi-core quantum architecture requires some specificity, which at least includes: (i) dedicated hardware for the entanglement generation and distribution unit; (ii) a classical control unit enabling classical communications between the cores needed by the quantum communication primitives and by the entanglement generation and distribution unit. So, while in the classical domain parallel processing is enabled by a shared memory, in the quantum domain parallelism is enabled by shared entangled states.

Despite this increase in the realization complexity, the interconnection of quantum multi-cores is worthwhile to be pursued. Indeed, while in classical multi-core architectures a linear increase of the number of cores roughly corresponds to a linear increase of the computational power, in the quantum domain the increasing of the computational power is, in principle, exponential.

The differences between classical and quantum multi-core architectures are not limited to architectural aspects. Indeed, while in the classical domain we have the issue of cache coherence, which concerns the consistency of data stored and processed by multiple entities, in the quantum domain, we have *entanglement coherence*, concerning the consistency of entangled states shared among the quantum computers plays a fundamental role [170]. This, in turn, changes the design principles. Indeed, any action on one entangled qubit affects the overall augmented coupling map. Hence it impacts the remote operation that can be performed [58]. Additionally, differently from the classically stored data, entanglement changes its state over time.

As mentioned in Section 2, in this type of DQC architectures, the physical distance between remote qubits is very short. And the number of cores that can be interconnected within a limited space cannot scale boundless. As a consequence, also the number of physical qubits that can be clustered together is limited. The physical qubit scalability can be enhanced by moving to multi-computer quantum architectures. However, we cannot adopt the reasoning adopted in the classical domain for moving from multi-core to multi-computer architectures. Indeed the specificity analyzed above for the multi-core quantum architectures, become even more peculiar. As instance, the dedicated hardware for the entanglement generation and distribution unit(s) must be more effective and efficient for covering the longer distances involved in this type of architecture. Additional hardware with no-counterpart in the classical architectures, such as the quantum transducer described in Section 7, may arise.

Similar considerations hold for the step from multi-computer to multi-farm architectures, with a clear worsening of the challenges due to the scale (both in terms of distances and nodes) of the distributed quantum computing architecture.

8.2. 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.

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, CEN/CENELEC [171] 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.

CRediT authorship contribution statement

Marcello Caleffi: Writing – review & editing, Writing – original draft, Supervision. **Michele Amoretti:** Writing – review & editing, Writing – original draft. **Davide Ferrari:** Writing – review & editing, Writing – original draft. **Jessica Illiano:** Writing – review & editing, Writing – original draft. **Antonio Manzalini:** Writing – review & editing, Writing – original draft. **Angela Sara Cacciapuoti:** Writing – review & editing, Writing – original draft, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgment

Angela Sara Cacciapuoti and Michele Amoretti acknowledge financial support from the European Union – NextGenerationEU, PNRR MUR project PE0000023-NQSTI.

References

[1] K. Nemoto, Our future with quantum computers, JSAP Rev. 2023 (2023) 230212.
[2] CSA QUCATS, Strategic Reasearch and Industry Agenda, European Commission, 2024.
[3] M. Caleffi, A.S. Cacciapuoti, G. Bianchi, Quantum internet: From communication to distributed computing!, in: Proc. of ACM NANOCOM '18, Association for Computing Machinery, 2018, pp. 1–4, http://dx.doi.org/10.1145/3233188.3233224.
[4] A.S. Cacciapuoti, M. Caleffi, F. Tafuri, F.S. Cataliotti, S. Gherardini, G. Bianchi, Quantum internet: Networking challenges in distributed quantum computing, IEEE Netw. 34 (1) (2020) 137–143, http://dx.doi.org/10.1109/MNET.001.1900092.
[5] R. Van Meter, S.J. Devitt, The Path to Scalable Distributed Quantum Computing, Computer 49 (9) (2016) 31–42, http://dx.doi.org/10.1109/MC.2016.291.
[6] J. Preskill, Quantum Computing in the NISQ era and beyond, Quantum 2 (79) (2018).

[7] Y. Kim, A. Eddins, S. Anand, K.X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, A. Kandala, Evidence for the utility of quantum computing before fault tolerance, Nature 618 (7965) (2023) 500–505.

[8] Y. Kim, C.J. Wood, T.J. Yoder, S.T. Merkel, J.M. Gambetta, K. Temme, A. Kandala, Scalable error mitigation for noisy quantum circuits produces competitive expectation values, Nat. Phys. 19 (5) (2023) 752–759.

[9] S. Wehner, D. Elkouss, R. Hanson, Quantum Internet: a Vision for the Road Ahead, Science 362 (6412) (2018).

[10] M. Caleffi, D. Chandra, D. Cuomo, S. Hasaanpour, A.S. Cacciapuoti, The Rise of the Quantum Internet., IEEE Comput. (2020).

[11] R. Parekh, A. Ricciardi, A. Darwish, S. DiAdamo, Quantum algorithms and simulation for parallel and distributed quantum computing, in: 2021 IEEE/ACM Second International Workshop on Quantum Computing Software, QCS, IEEE Computer Society, Los Alamitos, CA, USA, 2021, pp. 9–19, http://dx.doi.org/10.1109/QCS54837.2021.00005, URL https://doi.ieeecomputersociety.org/10.1109/QCS54837.2021.00005.

[12] D. Cuomo, M. Caleffi, A.S. Cacciapuoti, Towards a distributed quantum computing ecosystem, IET Quantum Commun. 1 (2020) 3–8(5), http://dx.doi.org/10.1049/iet-qtc.2020.0002.

[13] D. Ferrari, A.S. Cacciapuoti, M. Amoretti, M. Caleffi, Compiler design for distributed quantum computing, IEEE Trans. Quantum Eng. 2 (2021) 1–20, http://dx.doi.org/10.1109/TQE.2021.3053921.

[14] J. Avron, O. Casper, I. Rozen, Quantum advantage and noise reduction in distributed quantum computing, Phys. Rev. A 104 (2021) 052404, http://dx.doi.org/10.1103/PhysRevA.104.052404.

[15] D. Ferrari, S. Carretta, M. Amoretti, A Modular Quantum Compilation Framework for Distributed Quantum Computing, IEEE Trans. Quantum Eng. 4 (2023) 1–13.

[16] D. Ferrari, M. Amoretti, A design framework for the simulation of distributed quantum computing, in: HPQCI Workshop in Conjunction with the 33rd ACM International Symposium on High-Performance Parallel and Distributed Computing, 2024.

[17] A. Gold, J.P. Paquette, A. Stockklauser, M.J. Reagor, M.S. Alam, A. Bestwick, N. Didier, A. Nersisyan, F. Oruc, A. Razavi, B. Scharmann, E.A. Sete, B. Sur, D. Venturelli, C.J. Winkleblack, F. Wudarski, M. Harburn, C. Rigetti, Entanglement across separate silicon dies in a modular superconducting qubit device, npj Quantum Inf. 7 (1) (2021) 142.

[18] IBM, Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing, URL https://research.ibm.com/blog/ibm-quantum-roadmap-2025.

[19] Y. Zhong, H.-S. Chang, A. Bienfait, et al., Deterministic multi-qubit entanglement in a quantum network, Nature 590 (7847) (2021) 571–575.

[20] M. Pompili, S.L.N. Hermans, S. Baier, et al., Realization of a multinode quantum network of remote solid-state qubits, Science 372 (6539) (2021) 259–264.

[21] 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) (2022) 663–668.

[22] J.V. Rakonjac, S. Grandi, S. Wengerowsky, D. Lago-Rivera, F. Appas, H. de Riedmatten, Transmission of light–matter entanglement over a metropolitan network, Optica Quantum 1 (2) (2023) 94–102.

[23] V. Krutyanskiy, M. Canteri, M. Meraner, V. Krcmarsky, B. Lanyon, Multimode ion-photon entanglement over 101 kilometers, PRX Quantum 5 (2024) 020308, http://dx.doi.org/10.1103/PRXQuantum.5.020308, URL https://link.aps.org/doi/10.1103/PRXQuantum.5.020308.

[24] N.M. Linke, D. Maslov, M. Roetteler, et al., Experimental comparison of two quantum computing architectures, Proc. Natl. Acad. Sci. 114 (13) (2017) 3305–3310.

[25] A. Kandala, K. Temme, A.D. Córcoles, et al., Error mitigation extends the computational reach of a noisy quantum processor, Nature 567 (7749) (2019) 491–495.

[26] Google Quantum AI, Official web site, https://quantumai.google/.

[27] IBM Quantum, Official web site, https://www.ibm.com/quantum.

[28] Rigetti, Official web site, https://www.rigetti.com/.

[29] Alice, Bob, Official web site, https://www.alice-bob.com/.

[30] Anyon, Official web site, https://anyonsys.com/.

[31] IQM, Official web site, https://www.meetiqm.com/.

[32] OQC, Official web site, https://oxfordquantumcircuits.com/.

[33] Intel, Intel–s New Chip to Advance Silicon Spin Qubit Research for Quantum Computing, https://rb.gy/3kz9ih.

[34] C12, Official web site, https://www.c12qe.com/.

[35] Quobly, Official web site, https://www.quobly.io/.

[36] Quantum Brilliance, Official web site, https://quantumbrilliance.com/.

[37] Alpine Quantum Computing, Official web site, https://www.aqt.eu/.

[38] IonQ, Official web site, https://ionq.com/.

[39] Quantinuum, Official web site, https://www.quantinuum.com/.

[40] Oxford Ionics, Official web site, https://www.oxionics.com/.

[41] PASQAL, Official web site, https://www.pasqal.com/.

[42] Quera, Official web site, https://www.quera.com/.

[43] Atom Computing, Official web site, https://atom-computing.com/.

[44] Infleqtion, Official web site, https://www.infleqtion.com/.

[45] R. Van Meter, S. Devitt, The path to scalable distributed quantum computing., Computer 49 (9) (2016) 31–42, http://dx.doi.org/10.1109/MC.2016.291.

[46] M.A. Nielsen, I.L. Chuang, Quantum computation and quantum information, Cambridge University Press, 2011.

[47] E. Rieffel, W. Polak, Quantum Computing: A Gentle Introduction, The MIT Press, 2011.

[48] J. Kim, D. Min, J. Cho, H. Jeong, I. Byun, J. Choi, J. Hong, J. Kim, A fault-tolerant million qubit-scale distributed quantum computer, in: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, 2024, pp. 1–19.

[49] J. Gambetta, Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing, https://www.ibm.com/quantum/blog/ibm-quantum-roadmap-2025.

[50] A. Ovide, S. Rodrigo, M. Bandic, H. Van Someren, S. Feld, S. Abadal, E. Alarcon, C.G. Almudever, Mapping quantum algorithms to multi-core quantum computing architectures, in: 2023 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2023, pp. 1–5.

[51] P. Escofet, S.B. Rached, S. Rodrigo, C.G. Almudever, E. Alarcón, S. Abadal, Interconnect fabrics for multi-core quantum processors: A context analysis, in: Proceedings of the 16th International Workshop on Network on Chip Architectures, NoCArc '23, ACM, 2023, pp. 34–39.

[52] S. Rodrigo, S. Abadal, E. Alarcon, M. Bandic, H. Van Someren, C.G. Almudéver, On double full-stack communication-enabled architectures for multicore quantum computers, IEEE Micro 41 (5) (2021) 48–56.

[53] H. Jnane, B. Undseth, Z. Cai, S.C. Benjamin, B. Koczor, Multicore quantum computing, Phys. Rev. Appl. 18 (2022) 044064, http://dx.doi.org/10.1103/PhysRevApplied.18.044064, URL https://link.aps.org/doi/10.1103/PhysRevApplied.18.044064.

[54] P. Escofet, S.B. Rached, S. Rodrigo, C.G. Almudever, E. Alarcón, S. Abadal, Interconnect fabrics for multi-core quantum processors: A context analysis, in: Proceedings of the 16th International Workshop on Network on Chip Architectures, 2023, pp. 34–39.

[55] P. Escofet, A. Ovide, M. Bandic, L. Prielinger, H. van Someren, S. Feld, E. Alarcón, S. Abadal, C.G. Almudéver, Revisiting the mapping of quantum circuits: Entering the multi-core era, ACM Trans. Quantum Comput. (2024).

[56] F. Mazza, M. Caleffi, A.S. Cacciapuoti, Intra-QLAN connectivity: beyond the physical topology, 2024, arXiv preprint arXiv:2406.09963.

[57] W. Kozlowski, S. Wehner, R. Van Meter, B. Rijsman, A.S. Cacciapuoti, M. Caleffi, S. Nagayama, Architectural principles for a quantum internet, 2023, http://dx.doi.org/10.17487/RFC9340, RFC 9340, URL https://www.rfc-editor.org/info/rfc9340.

[58] J. Illiano, M. Caleffi, A. Manzalini, A.S. Cacciapuoti, Quantum internet protocol stack: a comprehensive survey, Comput. Netw. 213 (2022) 109092.

[59] A.S. Cacciapuoti, J. Illiano, S. Koudia, K. Simonov, M. Caleffi, The quantum internet: Enhancing classical services one qubit at a time, IEEE Netw. 36 (5) (2022) 6–12.

[60] A.S. Cacciapuoti, M. Caleffi, Toward the quantum internet: A directional-dependent noise model for quantum signal processing, in: IEEE ICASSP '19, 2019, pp. 7978–7982, http://dx.doi.org/10.1109/ICASSP.2019.8683195.

[61] A.S. Cacciapuoti, M. Caleffi, R. Van Meter, L. Hanzo, When entanglement meets classical communications: Quantum teleportation for the quantum internet, IEEE Trans. Commun. 68 (6) (2020) 3808–3833, invited paper.

[62] R. Horodecki, P. Horodecki, M. Horodecki, K. Horodecki, Quantum entanglement, Rev. Modern Phys. 81 (2) (2009) 865.

[63] A. Unnikrishnan, D. Markham, Authenticated teleportation and verification in a noisy network, Phys. Rev. A 102 (2020) 042401.

[64] R. Van Meter, K. Nemoto, W. Munro, K. Itoh, Distributed arithmetic on a quantum multicomputer, in: 33rd International Symposium on Computer Architecture, ISCA'06, 2006, pp. 354–365.

[65] S. DiAdamo, M. Ghibaudi, J. Cruise, Distributed Quantum Computing and Network Control for Accelerated VQE, IEEE Trans. Quantum Eng. 2 (2021) 1–21, http://dx.doi.org/10.1109/TQE.2021.3057908.

[66] K. Azuma, S.E. Economou, D. Elkouss, P. Hilaire, L. Jiang, H.-K. Lo, I. Tzitrin, Quantum repeaters: From quantum networks to the quantum internet, Rev. Modern Phys. 95 (2023) 045006.

[67] J. Illiano, A.S. Cacciapuoti, A. Manzalini, M. Caleffi, The impact of the quantum data plane overhead on the throughput, in: Proc. of ACM NANOCOM '21, 2021, pp. 1–6, http://dx.doi.org/10.1145/3477206.3477448.

[68] A.S. Cacciapuoti, J. Illiano, M. Caleffi, Quantum internet addressing, IEEE Netw. 38 (1) (2024) 104–111, http://dx.doi.org/10.1109/MNET.2023.3328393.

[69] S.-Y. Chen, J. Illiano, A.S. Cacciapuoti, M. Caleffi, Entanglement-based artificial topology: Neighboring remote network nodes, 2024, arXiv preprint arXiv:2404.16204.

[70] W. Dür, H.J. Briegel, Entanglement purification and quantum error correction, Rep. Progr. Phys. 70 (8) (2007) 1381.

[71] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, et al., A link layer protocol for quantum networks, in: Proceedings of the ACM Special Interest Group on Data Communication, 2019, pp. 159–173.

[72] R. Van Meter, Distributed digital computation and communication, in: Quantum Networking, John Wiley & Sons, Ltd, 2014, pp. 113–130.

[73] S. Shi, C. Qian, Concurrent entanglement routing for quantum networks: Model and designs, in: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, 2020, pp. 62–75.

[74] F. Dupuy, C. Goursaud, F. Guillemin, A survey of quantum entanglement routing protocols–challenges for wide-area networks, Adv. Quantum Technol. 6 (5) (2023) 2200180.

[75] A. Montanaro, Quantum algorithms: An overview, npj Quantum Inf. 2 (1) (2016) 15023, http://dx.doi.org/10.1038/npjqi.2015.23.

[76] A. J., A. Adedoyin, J. Ambrosiano, et al., Quantum algorithm implementations for beginners, ACM Trans. Quantum Comput. 3 (4) (2022).

[77] P.W. Shor, Polynomial time algorithms for discrete logarithms and factoring on a quantum computer, in: Algorithmic Number Theory, Springer Berlin Heidelberg, 1994, p. 289.

[78] L.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, 1996, pp. 212–219.

[79] A.W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103 (2009) 150502.

[80] M. Cerezo, A. Arrasmith, R. Babbush, et al., Variational quantum algorithms, Nat. Rev. Phys. 3 (9) (2021) 625–644.

[81] B.M. Terhal, Quantum error correction for quantum memories, Rev. Modern Phys. 87 (2015) 307–346.

[82] E. Knill, Conventions for quantum pseudocode, in: Tech. rep, Los Alamos National Lab, United States, 1996, URL https://www.osti.gov/biblio/366453.

[83] H. Abraham, I.Y. Akhalwaya, G. Aleksandrowicz, et al., Qiskit: An open-source framework for quantum computing, 2019.

[84] Google Quantum AI, Cirq, myehosthttps://quantumai.google/cirq.

[85] Xanadu, PennyLane, myehosthttps://pennylane.ai/.

[86] A.W. Cross, L.S. Bishop, J.A. Smolin, J.M. Gambetta, Open quantum assembly language, 2017, arXiv:1707.03429.

[87] A. Cross, A. Javadi-Abhari, T. Alexander, N. De Beaudrap, L.S. Bishop, S. Heidel, C.A. Ryan, P. Sivarajah, J. Smolin, J.M. Gambetta, B.R. Johnson, Openqasm 3: A broader and deeper quantum assembly language, ACM Trans. Quantum Comput. 3 (3) (2022) 1–50.

[88] A. Dahlberg, B. van der Vecht, C.D. Donne, et al., NetQASM - a low-level instruction set architecture for hybrid quantum–classical programs in a quantum internet, Quantum Sci. Technol. 7 (3) (2022) 035023.

[89] A. Peduri, S. Bhat, T. Grosser, QSSA: an SSA-based IR for Quantum computing, in: Proceedings of the 31st ACM SIGPLAN International Conference on Compiler Construction, 2022, pp. 2–14.

[90] D. Ittah, T. Häner, V. Kliuchnikov, T. Hoefler, QIRO: A static single assignment-based quantum program representation for optimization, ACM Trans. Quantum Comput. 3 (3) (2022).

[91] S. Nishio, R. Wakizaka, Inquir: Intermediate representation for interconnected quantum computers, 2023, arXiv:2302.00267.

[92] R. Cleve, J. Watrous, Fast parallel circuits for the quantum Fourier transform, in: Proceedings 41st Annual Symposium on Foundations of Computer Science, 2000, pp. 526–536.

[93] N.M.P. Neumann, R. van Houte, T. Attema, Imperfect Distributed Quantum Phase Estimation, in: Computational Science – ICCS 2020, in: Lecture Notes in Computer Science, Springer International Publishing, 2020, pp. 605–615.

[94] A. Kitaev, Quantum computations: algorithms and error correction, Russian Math. Surveys 52 (6) (1997) 1191–1249.

[95] J. Eisert, K. Jacobs, P. Papadopoulos, M.B. Plenio, Optimal local implementation of nonlocal quantum gates, Phys. Rev. A 62 (2000) 052317.

[96] A. Yimsiriwattana, S. Lomonaco, Generalized GHZ states and distributed quantum computing, Contemp. Math. 381 (2005) http://dx.doi.org/10.1090/conm/381.

[97] N.M.P. Neumann, R.S. Wezeman, Distributed quantum machine learning, in: Innovations for Community Services, Springer International Publishing, 2022, pp. 281–293.

[98] C. Cicconetti, M. Conti, A. Passarella, Resource allocation in quantum networks for distributed quantum computing, in: 2022 IEEE International Conference on Smart Computing, SMARTCOMP, 2022, pp. 124–132.

[99] C. Cicconetti, M. Conti, A. Passarella, Service differentiation and fair sharing in distributed quantum computing, Pervasive Mob. Comput. 90 (2023) 101758.

[100] G. Vardoyan, S. Wehner, Quantum Network Utility Maximization, 2022, arXiv e-prints, arXiv:2210.08135v1.

[101] Y. Lee, W. Dai, D. Towsley, D. Englund, Quantum Network Utility: A Framework for Benchmarking Quantum Networks, 2022, arXiv e-prints, arXiv:2210.10752v1.

[102] A.W. Cross, L.S. Bishop, S. Sheldon, P.D. Nation, J.M. Gambetta, Validating quantum computers using randomized model circuits, Phys. Rev. A 100 (2019) 032328, http://dx.doi.org/10.1103/PhysRevA.100.032328.

[103] A. Botea, A. Kishimoto, R. Marinescu, On the Complexity of Quantum Circuit Compilation, in: The Eleventh International Symposium on Combinatorial Search (SOCS 2018), 2018.

[104] J. Kusyk, S.M. Saeed, M.U. Uyar, Survey on quantum circuit compilation for noisy intermediate-scale quantum computers: Artificial intelligence to heuristics, IEEE Trans. Quantum Eng. 2 (2021) 1–16, http://dx.doi.org/10.1109/TQE.2021.3068355.

[105] S. Sivarajah, S. Dilkes, A. Cowtan, et al., T|ket⟩: a retargetable compiler for NISQ devices, Quantum Sci. Technol. 6 (1) (2020) 014003.

[106] A.D. Carcoles, A. Kandala, A. Javadi-Abhari, et al., Challenges and opportunities of near-term quantum computing systems, Proc. of the IEEE (2020) 1–15, in press.

[107] D. Ferrari, M. Amoretti, Efficient and effective quantum compiling for entanglement-based machine learning on IBM Q devices, Int. J. Quantum Inf. 16 (08) (2018) 1840006.

[108] L. Cincio, Y. Subaşı, A.T. Sornborger, P.J. Coles, Learning the quantum algorithm for state overlap, New J. Phys. 20 (11) (2018) 113022.

[109] A. Zulehner, A. Paler, R. Wille, An efficient methodology for mapping quantum circuits to the IBM qx architectures, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 38 (7) (2019) 1226–1236.

[110] M. Soeken, G. Meuli, B. Schmitt, et al., Boolean satisfiability in quantum compilation, Phil. Trans. Royal Soc. A 378 (2164) (2019) 1–16, http://dx.doi.org/10.1098/rsta.2019.0161.

[111] C. Chamberland, G. Zhu, T.J. Yoder, et al., Topological and Subsystem Codes on Low-Degree Graphs with Flag Qubits, Phys. Rev. X 10 (011022) (2020).

[112] IBM Q, Transpiler, https://qiskit.org/documentation/apidoc/transpiler.html.

[113] P. Andrés-Martínez, C. Heunen, Automated distribution of quantum circuits via hypergraph partitioning, Phys. Rev. A 100 (2019) 032308, http://dx.doi.org/10.1103/PhysRevA.100.032308.

[114] R. G. Sundaram, H. Gupta, C.R. Ramakrishnan, Efficient Distribution of Quantum Circuits, in: 35th International Symposium on Distributed Computing (DISC 2021), 2021.

[115] R.G. Sundaram, H. Gupta, C.R. Ramakrishnan, Distribution of Quantum Circuits Over General Quantum Networks, in: 2022 IEEE International Conference on Quantum Computing and Engineering, QCE, 2022, pp. 415–425.

[116] O. Daei, K. Navi, M. Zomorodi-Moghadam, Optimized quantum circuit partitioning, Internat. J. Theoret. Phys. 59 (12) (2020) 3804–3820, http://dx.doi.org/10.1007/s10773-020-04633-8.

[117] Z. Davarzani, M. Zomorodi-Moghadam, M. Houshmand, M. Nouri-baygi, A dynamic programming approach for distributing quantum circuits by bipartite graphs, Quantum Inf. Process. 19 (2020) http://dx.doi.org/10.1007/s11128-020-02871-7.

[118] E. Nikahd, N. Mohammadzadeh, M. Sedighi, M.S. Zamani, Automated window-based partitioning of quantum circuits, Phys. Scr. 96 (3) (2021) 035102, http://dx.doi.org/10.1088/1402-4896/abd57c.

[119] D. Cuomo, M. Caleffi, K. Krsulich, F. Tramonto, G. Agliardi, E. Prati, A.S. Cacciapuoti, Optimized compiler for distributed quantum computing, ACM Trans. Quantum Comput. 4 (2) (2023) 1–29.

[120] D. Dadkhah, M. Zomorodi, S.E. Hosseini, A New Approach for Optimization of Distributed Quantum Circuits, Internat. J. Theoret. Phys. 60 (9) (2021) 3271–3285, http://dx.doi.org/10.1007/s10773-021-04904-y.

[121] R. Beals, S. Brierley, O. Gray, et al., Efficient distributed quantum computing, Proc. R. Soc. A Math. Phys. Eng. Sci. 469 (2153) (2013) 20120686.

[122] S. Brierley, Efficient implementation of quantum circuits with limited qubit interactions, Quantum Info. Comput. 17 (13–14) (2017) 1096–1104.

[123] P. Andres-Martinez, T. Forrer, D. Mills, J. Wu, L. Henaut, K. Yamamoto, M. Murao, R. Duncan, Distributing circuits over heterogeneous, modular quantum computing network architectures, Quantum Science and Technology (2024) http://dx.doi.org/10.1088/2058-9565/ad6734.

[124] P. Promponas, A. Mudvari, L. Della Chiesa, P. Polakos, L. Samuel, L. Tassiulas, Compiler for distributed quantum computing: a reinforcement learning approach, 2024, arXiv preprint arXiv:2404.17077.

[125] Y. Shi, N. Leung, P. Gokhale, Z. Rossi, D.I. Schuster, H. Hoffmann, F.T. Chong, Optimized compilation of aggregated instructions for realistic quantum computers, in: Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1031–1044, http://dx.doi.org/10.1145/3297858.3304018, URL https://doi.org/10.1145/3297858.3304018.

[126] P. Gokhale, A. Javadi-Abhari, N. Earnest, Y. Shi, F.T. Chong, Optimized quantum compilation for near-term algorithms with OpenPulse, in: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO, 2020, pp. 186–200, http://dx.doi.org/10.1109/MICRO50266.2020.00027.

[127] J. Cheng, H. Deng, X. Qia, Accqoc: Accelerating quantum optimal control based pulse generation, in: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture, ISCA, 2020, pp. 543–555, http://dx.doi.org/10.1109/ISCA45697.2020.00052.

[128] S. Debnath, N.M. Linke, C. Figgatt, K.A. Landsman, K. Wright, C. Monroe, Demonstration of a small programmable quantum computer with atomic qubits, Nature 536 (7614) (2016) 63–66, http://dx.doi.org/10.1038/nature18648.

[129] A.G. Fowler, S.J. Devitt, L.C.L. Hollenberg, Implementation of Shor's algorithm on a linear nearest neighbor qubit array, Quantum Inf. Process. 4 (2004) 237–251, http://dx.doi.org/10.26421/QIC4.4.

[130] B.W. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs, Bell Syst. Tech. J. 49 (2) (1970) 291–307, http://dx.doi.org/10.1002/j.1538-7305.1970.tb01770.x.
[131] B. Bartlett, A distributed simulation framework for quantum networks and channels, 2018, arXiv e-prints, arXiv:1808.07047.
[132] T. Coopmans, R. Knegjens, A. Dahlberg, et al., NetSquid, a NETwork Simulator for QUantum Information using Discrete events, Commun. Phys. 4 (1) (2021) 164.
[133] A. Dahlberg, S. Wehner, SimulaQron - a simulator for developing quantum internet software, Quantum Sci. Technol. 4 (1) (2018) 015001.
[134] X. Wu, A. Kolar, J. Chung, et al., SeQUeNCe: a customizable discrete-event simulator of quantum networks, Quantum Sci. Technol. 6 (4) (2021) 045027.
[135] T. Matsuo, Simulation of a dynamic, RuleSet-based quantum network, 2021, arXiv e-prints, arXiv:1908.10758.
[136] S. DiAdamo, J. Notzel, B. Zanger, M.M. Bese, QuNetSim: A Software Framework for Quantum Networks, IEEE Trans. Quantum Eng. 2 (2021) 1–12.
[137] QuTech, Quantum Network Explorer ADK, 2022, URL https://github.com/QuTech-Delft/qne-adk.
[138] C.-T. Liao, S. Bahrani, F.F. da Silva, E. Kashefi, Benchmarking of quantum protocols, Sci. Rep. 12 (1) (2022) 5298, http://dx.doi.org/10.1038/s41598-022-08901-x.
[139] M. Mehic, M. Niemiec, S. Rass, et al., Quantum key distribution: A networking perspective, ACM Comput. Surv. 53 (5) (2020).
[140] A. Manzalini, M. Amoretti, End-to-end entanglement generation strategies: Capacity bounds and impact on quantum key distribution, Quantum Rep. 4 (3) (2022) 251–263.
[141] C. Cicconetti, M. Conti, A. Passarella, Request scheduling in quantum networks, IEEE Trans. Quantum Eng. 2 (2021) 2–17.
[142] Various Authors, Quantum protocol zoo, 2022, URL https://wiki.veriqloud.fr/index.php.
[143] S. Devitt, W. Munro, K. Nemoto, Quantum error correction for beginners, Rep. Progr. Phys. 76 (7) (2013).
[144] D.S. Steiger, T. Häner, M. Troyer, ProjectQ: An Open Source Software Framework for Quantum Computing, Quantum 2 (2018) 49, http://dx.doi.org/10.22331/q-2018-01-31-49, arXiv:1612.08091.
[145] B. Zanger, S. DiAdamo, EQSN: Effective Quantum Simulator for Networks, 2020, URL https://github.com/tqsd/EQSN_python.
[146] D. Ferrari, S. Nasturzio, M. 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.
[147] A.W. Cross, L.S. Bishop, J.A. Smolin, J.M. Gambetta, Open quantum assembly language, 2017, arXiv e-prints, arXiv:1707.03429.
[148] P. Magnard, S. Storz, P. Kurpiers, et al., Microwave quantum link between superconducting circuits housed in spatially separated cryogenic systems, Phys. Rev. Lett. 125 (2020) 260502.
[149] L. d'Avossa, M. Caleffi, C. Wang, J. Illiano, S. Zorzetti, A.S. Cacciapuoti, Towards the quantum internet: entanglement rate analysis of high-efficiency electro-optic transducer, in: 2023 IEEE International Conference on Quantum Computing and Engineering, QCE, 1, IEEE, 2023, pp. 1325–1334.
[150] N. Lauk, N. Sinclair, S. Barzanjeh, J.P. Covey, M. Saffman, M. Spiropulu, C. Simon, Perspectives on quantum transduction, Quantum Sci. Technol. 5 (2) (2020) 020501.
[151] L. d'Avossa, A.S. Cacciapuoti, M. Caleffi, Quantum transduction models for multipartite entanglement distribution, IEEE QCNC24 (2024).
[152] C.H. Bennett, G. Brassard, S. Popescu, et al., Purification of noisy entanglement and faithful teleportation via noisy channels, Phys. Rev. Lett. 76 (5) (1996) 722.
[153] C.H. Bennett, D.P. DiVincenzo, J.A. Smolin, W.K. Wootters, Mixed-state entanglement and quantum error correction, Phys. Rev. A 54 (5) (1996) 3824.
[154] J.I. Cirac, A.K. Ekert, S.F. Huelga, C. Macchiavello, Distributed quantum computation over noisy channels, Phys. Rev. A 59 (1999) 4249–4254.
[155] L. Ruan, W. Dai, M.Z. Win, Adaptive recurrence quantum entanglement distillation for two-kraus-operator channels, Phys. Rev. A 97 (5) (2018) 052332.
[156] F. Rozpędek, T. Schiet, D. Elkouss, et al., Optimizing practical entanglement distillation, Phys. Rev. A 97 (6) (2018) 062333.
[157] L. Ruan, B.T. Kirby, M. Brodsky, M.Z. Win, Efficient entanglement distillation for quantum channels with polarization mode dispersion, Phys. Rev. A 103 (3) (2021) 032425.
[158] F. Mazza, M. Caleffi, A.S. Cacciapuoti, Quantum LAN: On-demand network topology via two-colorable graph states, IEEE QCNC24 (2024).
[159] K. Simonov, M. Caleffi, J. Illiano, A.S. Cacciapuoti, Universal quantum computation via superposed orders of single-qubit gates, 2023, arXiv preprint arXiv:2311.13654.
[160] F. Riera-Sàbat, W. Dür, A modular entanglement-based quantum computer architecture, 2024, arXiv preprint arXiv:2406.05735.
[161] R. Raussendorf, H. Briegel, A one-way quantum computer, Phys. Rev. Lett. 86 (2001) 5188–5191.
[162] R. Raussendorf, H. Briegel, Computational model underlying the one-way quantum computer, 2001, arXiv preprint quant-ph/0108067.
[163] M. Hein, J. Eisert, H.J. Briegel, Multiparty entanglement in graph states, Phys. Rev. A 69 (6) (2004) 062311.
[164] P. Murali, J.M. Baker, A.J. Abhari, et al., Noise-adaptive compiler mappings for noisy intermediate-scale quantum computers, 2019, arXiv e-prints, arXiv:1901.11054.
[165] S. Nishio, Y. Pan, T. Satoh, et al., Extracting success from ibm–s 20-qubit machines using error-aware compilation, J. Emerg. Technol. Comput. Syst. 16 (3) (2020).
[166] S. Niu, A. Suau, G. Staffelbach, A. Todri-Sanial, A hardware-aware heuristic for the qubit mapping problem in the nisq era, IEEE Trans. Quantum Eng. 1 (2020) 1–14.
[167] D. Ferrari, M. Amoretti, Noise-adaptive quantum compilation strategies evaluated with application-motivated benchmarks, in: Proceedings of the 19th ACM International Conference on Computing Frontiers, CF '22, 2022, pp. 237–243, http://dx.doi.org/10.1145/3528416.3530250.
[168] J. Wang, G. Guo, Z. Shan, Sok: Benchmarking the performance of a quantum computer, Entropy 24 (10) (2022) http://dx.doi.org/10.3390/e24101467.
[169] F. Bouchard, K. Fenwick, K. Bonsma-Fisher, D. England, P.J. Bustard, K. Heshami, B. Sussman, Programmable photonic quantum circuits with ultrafast time-bin encoding, 2024, arXiv preprint arXiv:2404.17657.
[170] M. Amoretti, S. Carretta, Entanglement verification in quantum networks with tampered nodes, IEEE J. Sel. Areas Commun. 38 (3) (2020) 598–604.
[171] O. van Deventer, N. Spethmann, M. Loeffler, M. Amoretti, et al., Towards European Standards for Quantum Technologies, EPJ Quantum Technol. 9 (3) (2022).

Marcello Caleffi is currently Professor with the DIETI Department, University of Naples Federico II, where he co-lead the Quantum Internet Research Group. He is also with the National Laboratory of Multimedia Communications, National Inter- University Consortium for Telecommunications. From 2010 to 2011, he was with the Broadband Wireless Networking Laboratory with the Georgia Institute of Technology, as a Visiting Researcher. In 2011, he was also with the NaNoNetworking Center in Catalunya (N3Cat) with the Universitat Politecnica de Catalunya, as a Visiting Researcher. Since July 2018, he held the Italian National Habilitation as a Full Professor of Telecommunications Engineering. His work appeared in several premier IEEE Transactions and Journals, and he received multiple awards, including the "*2024 IEEE Communications Society Award for Advances in Communication* and the "*2022 IEEE Communications Society Best Tutorial Paper Award*". He currently serves as an Editor/Associate Editor for IEEE Trans. On Wireless Communications, IEEE Trans. on Communications, IEEE Transactions On Quantum Engineering, IEEE Open Journal of the Communications Society and IEEE Internet Computing. He has served as the chair, the TPC chair, and a TPC member for several premier IEEE conferences. In 2017, he has been appointed as Distinguished Visitor Speaker from the IEEE Computer Society and he has been elected treasurer of the IEEE ComSoc/VT Italy Chapter. In 2019, he has been also appointed as a member of the IEEE New Initiatives Committee from the IEEE Board of Directors and, in 2023, he has been appointed as ComSoc Distinguished Lecturer.

Michele Amoretti received his Ph.D. in Information Technologies in 2006 from the University of Parma, Parma, Italy. He is an Associate Professor of Computer Engineering at the University of Parma (Italy), where he leads the Quantum Software Laboratory (QSLab) in the Department of Engineering and Architecture. He authored or co-authored over 130 research papers in refereed international journals, conference proceedings, and books. He serves as *Associate Editor* for the journal IEEE Trans. on Quantum Engineering. He is the Principal Investigator of the University of Parma's research unit involved in the "Quantum Internet Alliance" project funded by the European Union – Horizon Europe – Quantum Flagship initiative. He is also involved as a researcher in the "National Quantum Science and Technology Institute (NQSTI)", funded by the European Union – Next Generation EU. He is a member of the Italian delegation in CEN/CENELEC's JTC 22 "Quantum Technologies".

Davide Ferrari his Ph.D. in Information Technologies at the Department of Engineering and Architecture of the University of Parma, Italy, in 2023. During the Ph.D. he worked on quantum compiling, quantum optimization and distributed quantum computing. He has been a research scholar at Future Technology Lab of the University of Parma, working on the design of efficient algorithms for quantum compiling. He is now a research fellow at the Department of Engineering and Architecture of the University of Parma. He is involved in the Quantum Information Science (QIS) research initiative at the University of Parma, where he is a member of the Quantum Software Laboratory. In 2020, he won the 'IBM Quantum Awards Circuit Optimization Developer Challenge'. His research focuses on quantum optimization applications and efficient quantum compiling for local and distributed quantum computing.

Jessica Illiano received the B.Sc degree in 2018 and then the M.Sc degree in 2020 both (summa cum laude) in Telecommunications Engineering from University of Naples Federico II (Italy). In 2020 she was winner of the scholarship "Quantum Communication Protocols for Quantum Security and Quantum Internet" fully funded by TIM S.p.A. and in 2024 she received her Ph.D. degree in Information Technologies and Electrical Engineering at University of Naples Federico II. Since 2017, she is a member of the Quantum Internet Research Group, FLY: Future Communications Laboratory at the University of Naples Federico II where she currently is Assistant Professor. Currently, she is website co-chair of N2Women and student Associate Editor for IET Quantum Communication. Her research interests include quantum communications, quantum networks and quantum information processing.

Antonio Manzalini received the M. Sc. Degree in Electronic Engineering from the Politecnico of Turin (Italy) and the Ph.D. (cum Laude) from Sorbonne Universités (France). In 1990 he joined Telecom Italia (formerly CSELT) where he was involved in innovation activities on technologies and architectures for optical networks. He actively participated in standardization, mainly in ETSI and ITU-T, and he was involved in several EURESCOM and European Project playing responsibility roles. He was Chair of the IEEE initiative on Software Defined Networks (SDN), and he was General Chair of the several IEEE Conferences. He owns several patents on methods and systems for networks and services. His results were published in more than 130 of technical papers and publications. Currently, he is working in TIM (Telecom Italia) Innovation, addressing Cloud-Edge Computing, Beyond-5G Networks and Quantum Communications and Computing. He is currently Chair in GSMA of a group on Quantum Networking and Services.

Angela Sara Cacciapuoti (www.quantuminternet.it) is a Professor of Quantum Communications and Networks at the University of Naples Federico II (Italy). Her work has appeared in first tier IEEE journals and she received different awards, including the "2024 IEEE ComSoc Award for Advances in Communication", the "2022 IEEE ComSoc Best Tutorial Paper Award", the "2022 WICE Outstanding Achievement Award" for her contributions in the quantum communication and network fields, and "2021 N2Women: Stars in Networking and Communications". Lately, she also received the IEEE ComSoc Distinguished Service Award for EMEA 2023, assigned for the outstanding service to IEEE ComSoc in the EMEA Region. Currently, she is an IEEE ComSoc Distinguished Lecturer with lecture topics on the Quantum Internet design and Quantum Communications. And she serves also as Member of the TC on SPCOM within the IEEE Signal Processing Society. Moreover, she serves as *Area Editor* for IEEE Trans. on Communications and as Editor/Associate Editor for the journals: IEEE Trans. on Quantum Engineering, IEEE Network and IEEE Communications Surveys & Tutorials. She served as Area Editor for IEEE Communications Letters(2019 - 2023), and she was the recipient of the 2017 Exemplary Editor Award of the IEEE Communications Letters. In 2023, she also served as Lead Guest Editor for IEEE JSAC special issue "The Quantum Internet: Principles, Protocols, and Architectures". From 2020 to 2021, Angela Sara was the Vice-Chair of the IEEE ComSoc Women in Communications Engineering. Previously, she has been appointed as Publicity Chair of WICE. From 2017 to 2020, she has been the Treasurer of the IEEE Women in Engineering (WIE) Affinity Group of the IEEE Italy Section. Her research interests are in Quantum Information Processing, Quantum Communications and Quantum Networks.