A Modular Compilation Framework for Distributed Quantum Computing

Michele Amoretti

Quantum Software Laboratory
Department of Engineering and Architecture
University of Parma

michele.amoretti@unipr.it

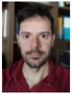
Summary

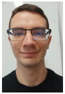
Quantum Software Laboratory

QSLab
QUANTUM SOFTWARE LABORATORY

Team

- Michele Amoretti (PhD, Associate Professor, QSLab Director)
- Davide Ferrari (PostDoc)
- Michele Bandini (Research Fellow and PhD Student)
- Giacomo Belli (Research Fellow)
- Marco Mordacci (PhD Student)





Main research interests

- Quantum algorithms
- Quantum compiling and gate synthesis
- Quantum protocols and quantum network applications
- Distributed quantum computing
- Quantum machine learning
- HPC/quantum integration and benchmarking

Projects

- Quantum Internet Alliance (QIA)
- National Quantum Science and Technology Institute (NQSTI)
- Advanced QUAntum MAchine learNing (AQUAMAN)

Preliminary Concepts

Entangled States

Sometimes it is not possible to decompose the state of an *n* qubit quantum register in the tensor product of the component states. Such states are denoted as **entangled states** (opposed to **separable states**). Their measurement outcomes are correlated.

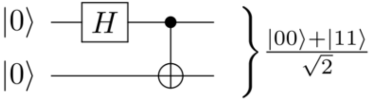
Example: **Bell states** (EPR pairs)

$$|\beta_{00}\rangle = |\phi^{+}\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$$

$$|\beta_{01}\rangle = |\psi^{+}\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}}$$

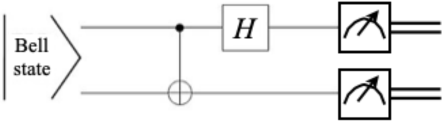
$$|\beta_{10}\rangle = |\phi^{-}\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

$$|\beta_{11}\rangle = |\psi^{-}\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$



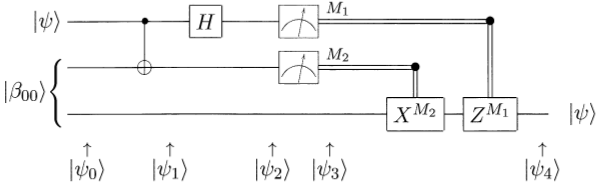
BSM is a crucial operation in quantum networking. The input of the corresponding quantum circuit is a Bell state, while the output is a pair of classical bits that reveal the nature of the input.

E.g., if the input is $|\beta_{00}\rangle$, the output is 00.



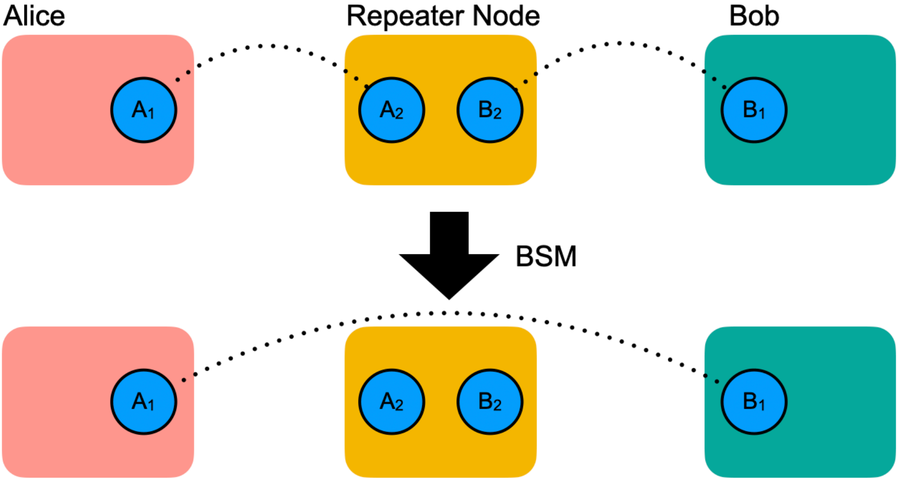
Quantum State Teleportation

"Once you disembody the state of one of particle, you can then recreate the particle in remote copy."
Charles Bennett, co-author of the first paper on quantum teleportation

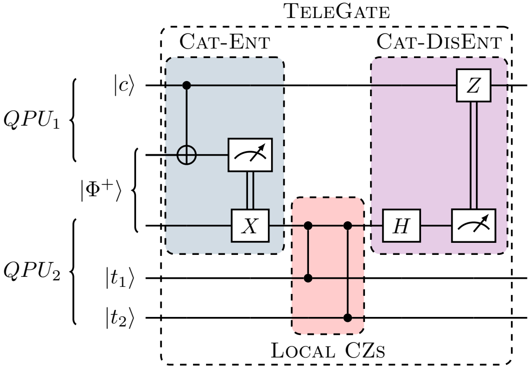


The two top lines of the quantum circuit represent Alice's system.
The bottom line represents Bob's system.

Entanglement Swapping



Quantum gate teleportation (also known as **TeleGate**) enables a direct gate between physical qubits stored at different processors without the need of quantum state teleportation, as long as a Bell state is distributed through the quantum link.

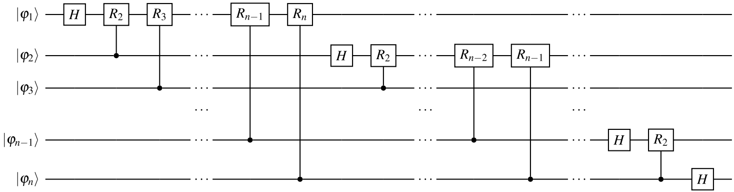


Distributed Quantum Computing

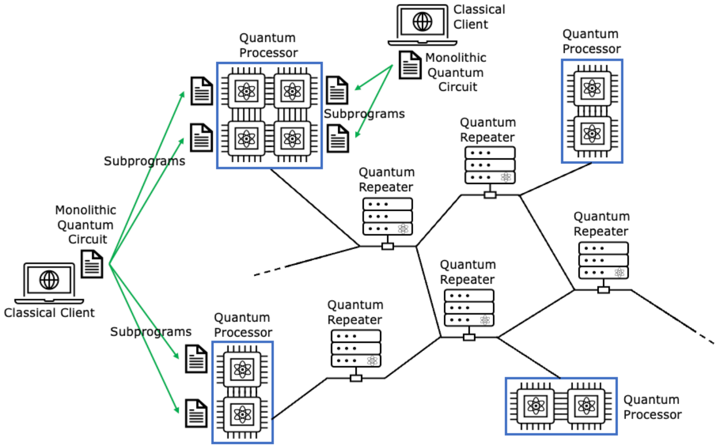
For most practical applications, **quantum algorithms** require large quantum computing resources.

For example, Shor's algorithm for integer factorization:

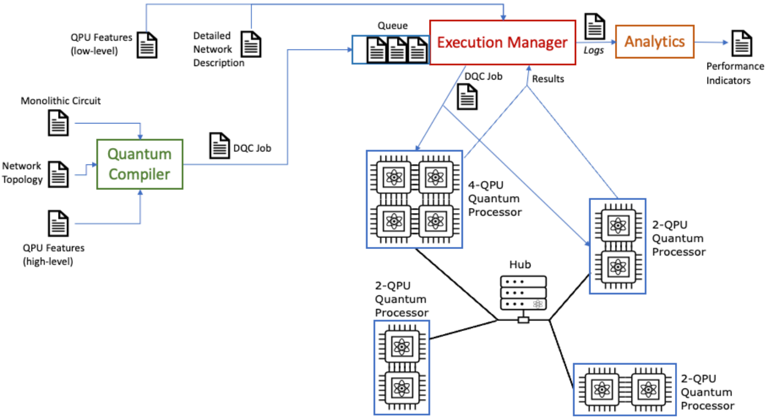
- based on the Quantum Fourier Transform (QFT)
- factoring $L = 2048$ bit primes, requires about $3L = 6144$ noise-free qubits



The growing demand for large-scale quantum computers is motivating research on distributed quantum computing (DQC) architectures.



In this talk, we consider the following **DQC workflow**:



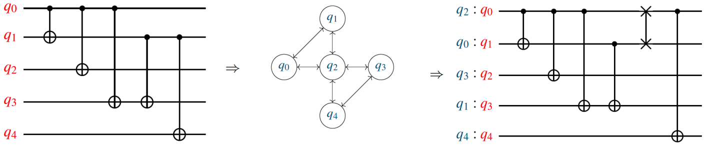
Quantum Compiling: translating an input quantum circuit into the most efficient equivalent of itself, taking into account the characteristics of the device that will execute the computation.

- A quantum algorithm designer focuses on the logic of the quantum circuit expressing the computation, regardless of the particulars of the quantum hardware that will execute the circuit.
- The **abstract** circuit must be mapped to a circuit to be executed on a specific quantum hardware by means of a suitable compiler.

In general, the quantum compilation problem is NP-Hard.

An abstract circuit is composed by **logical qubits**, while a quantum processor is equipped with a register of **physical qubits**.

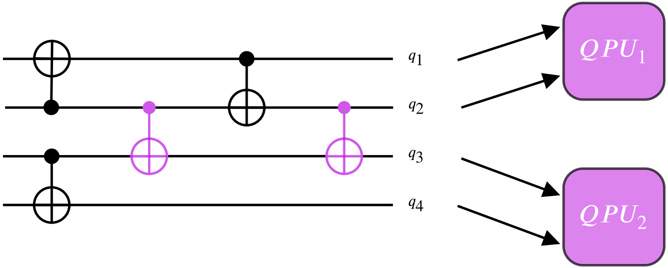
A **qubit assignment**, in its most basic form, is a one-to-one mapping between logical and physical qubits.



D. Ferrari, I. Tavernelli, M. Amoretti, Deterministic algorithms for compiling quantum circuits with recurrent patterns, Quantum Information Processing, vol. 20, no. 6, 2021

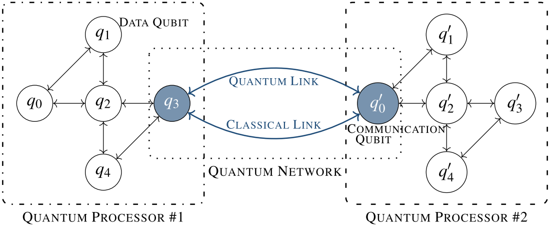
D. Ferrari, M. Amoretti, Noise-Adaptive Quantum Compilation Strategies Evaluated with Application-Motivated Benchmarks, Proc. of the 19th ACM International Conference on Computing Frontiers, Turin, Italy, 2022

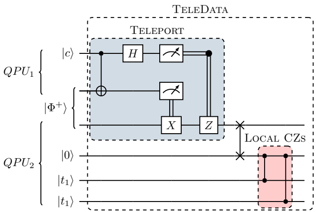
In DQC, for a given set of logical qubits, we need to choose a partition that maps sub-sets of logical qubits to processors, while minimizing the number of required interactions among different sub-sets.

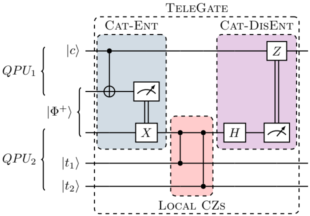


A key requirement for distributed quantum computing is the ability to perform **non-local operations**.

To this purpose, we exploit entanglement, namely **Bell states** (EPR pairs).



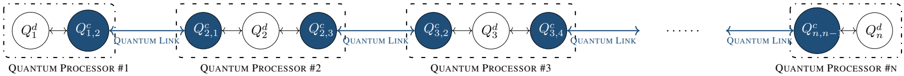




Worst-case Architecture

D. Ferrari, A. S. Cacciapuoti, M. Amoretti and M. Caleffi, Compiler Design for Distributed Quantum Computing, IEEE Transactions on Quantum Engineering, vol. 2, pp. 1-20, 2021

In this work, we proposed an efficient compiler for DQC, considering the **worst-case DQC architecture**.

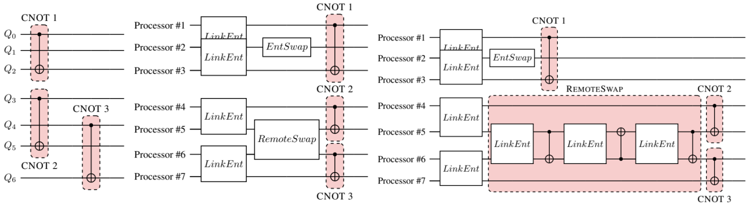


- Only one data qubit is available at each QPU
- QPUs are interconnected through a one-dimensional nearest-neighbor topology
- Overhead induced by any real-world architecture will be upper-bounded by the communication overhead induced by the worst-case architecture

Worst-case Architecture

We proposed a sorting strategy to reduce the depth overhead induced by handling non-local gates.

We proved that **the overhead is upper-bounded by a factor that grows linearly with the number of qubits**.



(a) A layer with three parallel CNOTs.

(b) The layer distributed with the Sort strategy.

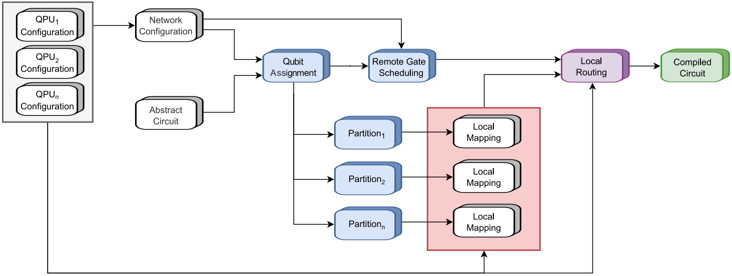
(c) The distributed layer after decomposing the Remote SWAP.

Modular Compilation Framework

D. Ferrari, S. Carretta, M. Amoretti, A Modular Quantum Compilation Framework for Distributed Quantum Computing, IEEE Transactions on Quantum Engineering, vol. 4, 2023

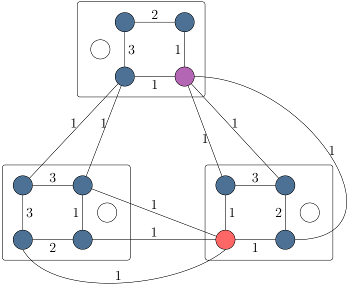
In this other work, we proposed a general purpose framework for compiling quantum circuits to DQC architectures.

- Circuit agnostic
- Bridging the gap between compilation for DQC and local compilation

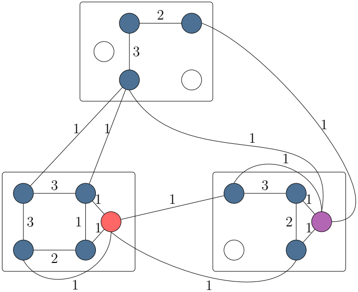


Qubit assignment: the goal is to partition the circuit minimizing the number of required EPR pairs.

The circuit is represented as a weighted graph.



(a)



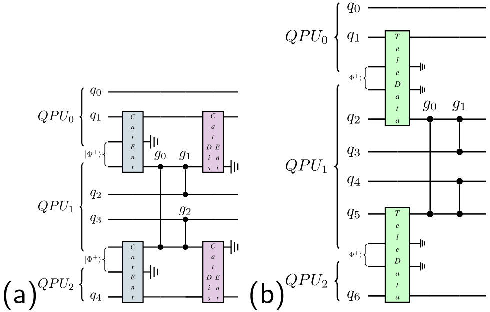
(b)

Solution (a) costs 8 EPR pairs, while solution (b) costs 6 EPR pairs.

Worst-case complexity: $O(n^3 p)$ for n qubits and p partitions

Non-local gate scheduling: the goal is to cover all non-local gates by means of `TeleData` or `TeleGate` operations.

The selection is based on a cost function.

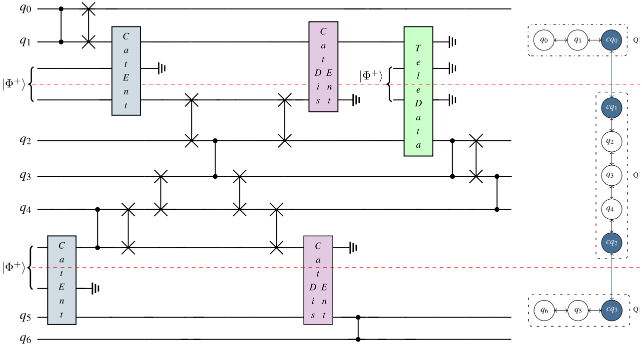


Solution (a) requires less data qubits than solution (b) on QPU_1.

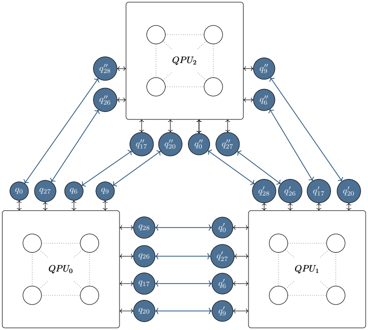
Worst-case complexity: $O(r^3 p)$ for r non-local gates and p partitions

Local routing: the goal is to handle local gates in a way that is compatible with the connectivity graph of the end nodes.

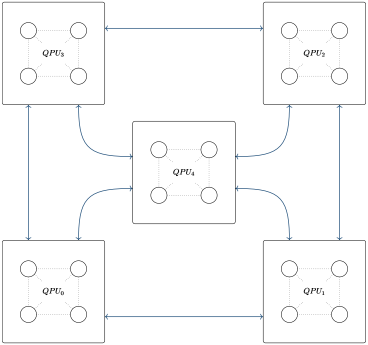
The algorithm scans the local circuit and for every gate that involves qubits not directly connected, computes the shortest sequence of necessary `SWAP` gates.



Considered architectures for experimental evaluation:

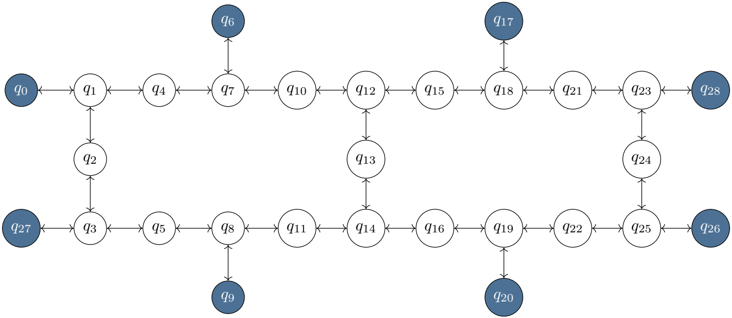


3-QPU architecture



5-QPU architecture

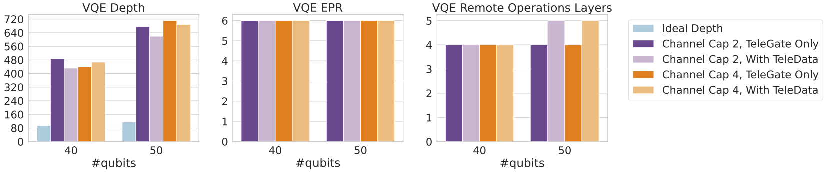
QPU configuration with 21 data qubits and 8 communication qubits, inspired by IBM's heavy hexagon devices:

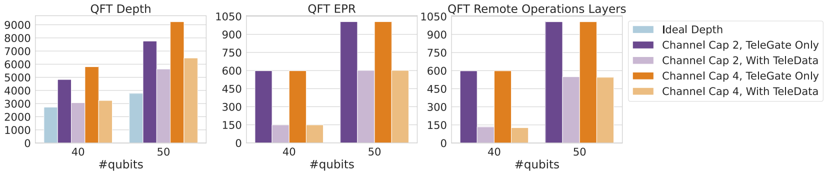


The heavy hexagon configuration can be scaled up in a modular fashion.

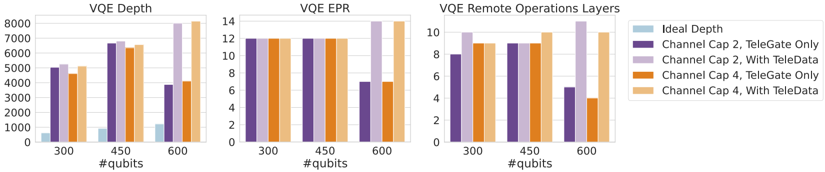
Modular Compilation Framework

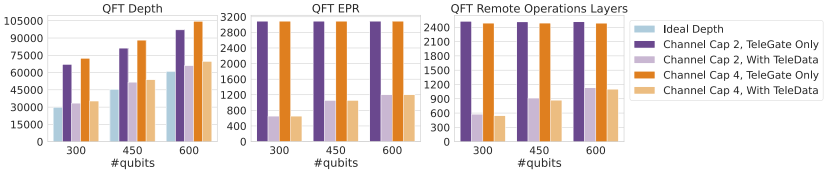
Some results using three 21-qubit QPUs:





Some results using five 125-qubit QPUs:





Current work focuses on:

- Output formats
- Resource (i.e., EPR pairs) optimization
- Experiments with complete network topologies

DQC Execution Management

D. Ferrari, M. Amoretti, A Design Framework for the Simulation of Distributed Quantum Computing, accepted for presentation at the HPQCI workshop, in conjunction with the 33rd ACM International Symposium on High-Performance Parallel and Distributed Computing, Pisa, Italy, 2024

D. Ferrari, M. Bandini, M. Amoretti, A Execution Management of Distributed Quantum Computing Jobs, Distributed Quantum Computing: Algorithms, Networks, Software, and Applications workshop at IEEE QCE, Montreal, Canada, September 2024

DQC execution management deals with the **parallel job scheduling** problem, in which set of jobs of varying processing times need to be scheduled on n_{QPU} machines while trying to minimize the **makespan**, i.e., the length of the schedule.

QPU utilization:

$$U_{\text{QPU}} = \frac{\sum_i p_i q_i}{M n_{\text{QPU}}} \in [0, 1] \tag{1}$$

where

- p_i is the estimated execution time of the i-th job
- q_i is the number of required QPUs of the i-th job
- M is the makespan of the schedule
- n_{QPU} is the number of the system's QPUs

Quantum network utilization:

$$U_{\text{QN}} = \frac{\sum_i N_{\text{R}i}}{\frac{(n_{\text{QPU}}-1)M}{r}} = \frac{r\sum_i N_{\text{R}i}}{(n_{\text{QPU}}-1)M} \in [0,1] \quad (2)$$

where

- r is the estimated execution time of a single non-local gate
- $N_{\text{R}i}$ is number of non-local gates in the i-th job
- $n_{\text{QPU}} - 1$ is the maximum number of remote gates in a layer that spans all the system's QPUs
- M is the makespan of the schedule

DQC Execution Management

First-In First-Out (FIFO) is a simple scheduling algorithm, it assumes that the first job entering the queue is the first job that must be scheduled for execution.

Algorithm FIFO-Scheduling

Input: job queue J, idle QPU set Q

```
function SCHEDULE
    i ← 0
    while Q ≠ ∅ do
        next ← J[i]
        if ∃q ⊆ Q : q = next.q then
            schedule next
            Q ← Q\q
            J ← J\next
        end if
    end while
end function
```

DQC Execution Management

List-scheduling (LS) is an efficient greedy algorithm that guarantees a makespan that is always at most $2 - 1/n_{\text{QPU}}$ times the optimal makespan.

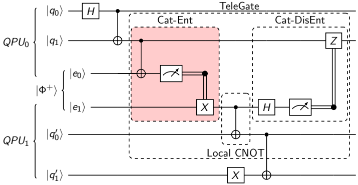
Algorithm List-Scheduling
Input: job queue J, idle QPU set Q

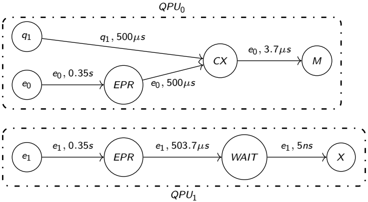
```
function SCHEDULE
    i ← 0
    while Q ≠ ∅ do
        next ← J[i]
        if ∃q ⊆ Q : q = next.q then
            schedule next
            Q ← Q\q
            J ← J\next
        else
            i ← i + 1
        end if
    end while
end function
```

DQC Execution Management

Given the hardware characteristics, the Execution Manager can estimate the **time** required for each job to complete by:

- analyzing the programs that make up a job
- creating directed acyclic graphs, one graph for each program





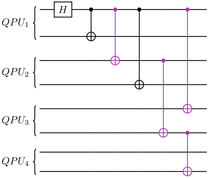
Carbon initialization time	300 μs
Electron initialization time	2 μs
Carbon one-qubit gate duration	20 μs
Electron one-qubit gate duration	5 ns
Electron two-qubits gate duration	500 μs
Electron readout time	3.7 μs
Entanglement generation time (fidelity of 0.8)	0.35 s

M. Pompili et al., Experimental demonstration of entanglement delivery using a quantum network stack, npj Quantum Information, 2022

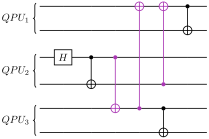
G. Avis et al., Requirements for a processing-node quantum repeater on a real-world fiber grid, npj Quantum Information, 2023

We evaluated the performance of FIFO and LS algorithms:

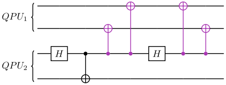
- five different jobs
- a network of six QPUs, each with two data qubits



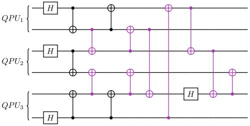
J_1



J_2

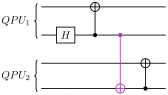


J_3



J_4

Job	Length [s]	n_{QPU}
J_1	1.055	4
J_2	0.708	3
J_3	0.706	2
J_4	1.406	3
J_5	0.357	2



J_5

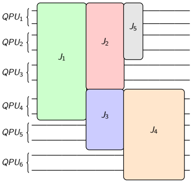
Three different job queues:

Queue		Makespan [s] FIFO	Makespan [s] LS
1	$\{J_5, J_4, J_3, J_2, J_1\}$	3.167	2.470
2	$\{J_1, J_4, J_2, J_5, J_3\}$	2.827	2.461
3	$\{J_5, J_1, J_4, J_2, J_3\}$	2.469	2.461

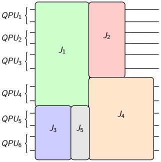
Job	Length [s]	n_{QPU}
J_1	1.055	4
J_2	0.708	3
J_3	0.706	2
J_4	1.406	3
J_5	0.357	2

With queue 1, the LS algorithm produces a schedule noticeably shorter than the one produced by FIFO.

Queue 1

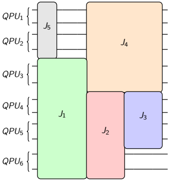


FIFO

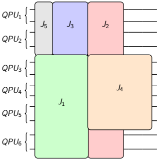


LS

Queue 3



FIFO



LS

DQC Execution Management

With LS, both QPU and Quantum network utilization are higher.

	U_{QPU}		U_{QN}	
Queue	FIFO	LS	FIFO	LS
1	0.668	0.856	0.465	0.597
2	0.748	0.859	0.521	0.599
3	0.856	0.859	0.597	0.599

- High QPU utilization is generally a good feature
- High quantum network utilization could instead be an issue

Quantum Internet Alliance

Quantum Internet Alliance

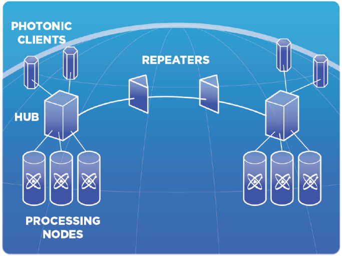
- Horizon Europe's FPA (7 years)
  - SGA1 (10/2022 – 3/2026)
  - SGA2 (4/2026 - 9/2029)
- 42 partners in 9 EU countries
- https://quantuminternetalliance.org/

QUANTUM INTERNET ALLIANCE

QUANTUM FLAGSHIP

This project (QIA) has received funding from the European Union's Horizon Europe programme.

- Fully programmable quantum network prototype connecting two metropolitan scale networks by a long-distance fiber backbone using quantum repeaters
- Two test protocols to inform technical requirements
  - Deterministic Teleportation
  - Blind Quantum Computation
- A world-leading European Platform for Quantum Internet Development that acts as a catalyst for an innovative ecosystem



QUANTUM INTERNET ALLIANCE

QUANTUM FLAGSHIP

This project (QIA) has received funding from the European Union's Horizon Europe programme.

Thank you!

QSLab
QUANTUM SOFTWARE LABORATORY

http://www.qis.unipr.it/quantumsoftware.html