Pan-European Quantum Internet Hackathon Technical Instructions

PEQI Technical Committee

October 2019

Introduction

Welcome to the Pan-European Quantum Internet Hackathon! The goal of this hackathon is to explore novel applications that can be realized with a Quantum Internet. You will have the opportunity to work with peers that share your interest in this field and will get a chance to work together in realizing such applications. The purpose of this document is to provide some additional detail into the technical side of things for the hackathon. This includes advice for developing your projects locally and some information regarding an additional simulated quantum network that you may use for your project.

Useful links

- `https://github.com/SoftwareQuTech/SimulaQron`. The SimulaQron code.
- `www.simulaqron.org`. The SimulaQron website.
- `https://arxiv.org/abs/1712.08032`. The publication on SimulaQron.
- `https://softwarequtech.github.io/SimulaQron/html/index.html`. Documentation.

Simulating Quantum Networks

For your projects in the Hackathon you will have the opportunity to make use of a simulated quantum network. The tool we will be using to fulfill this end will be SimulaQron, you can find documentation for this software at `https://softwarequtech.github.io/SimulaQron/html/index.html`. SimulaQron is a simulator for a quantum internet. Such an internet consists of nodes, which, on top of sending around bits (classical information), also send *quantum bits*, or *qubits*, which are the basic units of quantum information. Qubits have properties that are very different from classical bits; for example, depending on the way we choose to read-out a quantum bit (referred to as a *measurement basis*), we may read a completely random bit and also change the qubit's state. We will be sure, however, which value we will read if the measurement basis is 'compatible' with the qubit's state. The latter fact is useful for cryptographic applications, since a sender and receiver can encrypt information and read out using a secret measurement basis, only known to the two parties involved, whereas an eavesdropper would read out in a basis which returns random information (see also *Quantum Key Distribution* in the reference).

Two or more qubits can be *entangled*, which means that their physical properties cannot be described separately. A "strongest" form of entanglement between two qubits (called *maximal entanglement*) has two favorable properties:

- maximal correlation: if the two qubits are read out in the same basis, they are bound to yield two maximally correlated[1] read out bits, one for each qubit;
- maximal privacy: no-one else than the parties with access to the two qubits at the time of measurement can even have the slightest knowledge of these two read-out bits.

[1] In case you want to be really precise: the two bits are either maximally correlated or maximally *anti*correlated.

In the context of a quantum internet, entanglement can be used for many things: for encrypting data, for transferring qubits by a process called *quantum teleportation* and for protecting quantum data against decoherence of the state of the qubits (a form of quantum error correction), among many other things.

Finally, it is useful to know that manipulating a quantum bit can be done through a *quantum gate*, whose use is similar to the use of a logic gate in a classical circuit (although the underlying math is very different).

Dealing with quantum data is different from dealing with classical information, not in the least because of the fact that qubits can be entangled. Although researchers all over the world are developing increasingly more accurate quantum processers, these devices are not yet sufficiently reliable for deployment in a real network. But why wait with developing software for the quantum internet in the meantime? SimulaQron has been developed for precisely this goal: it is a simulator with software development as main purpose. What follows in the next section is an overview of the context in which SimulaQron was developed: as a stand-in for part of the quantum network stack.

Software can be independent from low-level hardware

To understand where in the overall system your programs will be running, it is useful to have a brief look at the bigger picture sketched in Figure 1. At the very bottom of this lies the actual quantum hardware. In Delft, we have two networked quantum hardware systems available: One very sophisticated system based on so-called NV centers in diamond. This is a small quantum computer, capable of executing quantum gates and measurements. Depending on the exact chip, these currently have between 1 and 10 qubits at QuTech. These have presently been entangled over a distance of 1.3kms at QuTech, which currently remains the world record in producing long lived entanglement. Such processors with a smaller number of qubits (2-5) are also planned to be deployed in the Dutch demonstration network at the end of 2020, with one such quantum computer in different cities. In addition, we also have a much simpler - but faster - system capable only of preparing and measuring one qubit at a time, for example for running quantum key distribution.

Within the European Quantum Internet Alliance, we also work with another quantum computing system, namely Ion Traps. For the purpose of this Hackathon, however, this system can be considered equivalent as a small quantum computer on which you can execute gates and make measurements! For networking, you want quantum computers that can (1) have an optical interface to talk to remote quantum computers (2) have good quantum memories, that is, the qubits live for a long amount of time allowing you to wait for communication (less than a second at present, but this is very long in the quantum world). Both NV in diamond as well as Ion Traps satisfy both of these demands. Other platforms that you may have read about online (superconducting qubits) presently do not.

At the quantum physical layer, such quantum hardware is connected via fiber. Using frequency conversion to the telecom band, standard telecom fibers can be used. In the demonstration network, we will for simplicity use a dedicated fiber for the actual quantum data, and send classical (control) information using a separate fiber. It is possible however in principle to transmit quantum and classical data over the same fiber, using technology at QuTech.

Above the quantum hardware sits a low level classical control system. This low level classical control is platform dependent, that is, it depends on what kind of quantum hardware (NV in diamond, Ion Traps, Photonic,...) sits underneath. It is responsible for controlling this exact physical system. Entanglement generating between neighbouring nodes is also dealt with on the low level controller, implementing our link and physical layer protocols.

Above, sits a high level control system (dubbed QNodeOS within the Quantum Internet Alliance). This is where applications are run, as well as higher level functionality such as network layer protocols sit. This higher level control system is platform independent, meaning that it can be programmed independently of what the underlying platform actually is. That is, for example applications you write there should in principle run on all platforms, and only be limited by the set of functionality they offer (such as the number of qubits, supported quantum gates, etc). The platform independent system can talk to the platform dependent one over a common interface called CQC (classical-quantum-combiner) which we are presently working on to implement as an interface to our hardware platforms. This is an extended API/instruction set that supports the executing of quantum gates and measuements on quantum processors, but also the creation of entanglement.

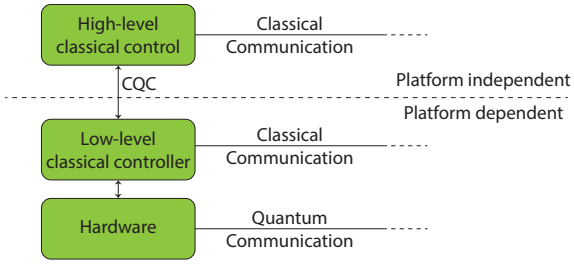

Figure 1: Very high level overview of the relation between the quantum hardware system, the low level control, and the high level classical control system on which applications are run. The clasical communication lines drawn here are logical ones and may be done over the same medium.

Relation to SimulaQron

SimulaQron provides a stand-in for the platform dependent part in a distributed simulation. You can talk to it over the same interface, namely CQC. Since this is quite cumbersome, a higher level python library allows you to program quite abstractly, and will then talk to the platform dependent system over CQC in order to execute quantum commands. Evidently, the simulation works differently than the real system in order to realize such behaviour: it is a distributed simulation giving you the illusion of talking to real quantum processors and being able to entangle them at a distance. This sounds very complicated, but the real system is also extremely complex, requiring a lot of nitty gritty which we will not need to consider in this hackathon!

Installation & getting started

For installing SimulaQron and how to set up the virtual servers, we refer to the documentation: `https://softwarequtech.github.io/SimulaQron/html/GettingStarted.html`.

During Development

While working on your projects we advise that you set up simulated networks on your local machine for debugging purposes and you may follow the SimulaQron documentation at `https://softwarequtech.github.io/SimulaQron/html/ConfNodes.html` to read more about how to configure and manage simulated networks. Simulating a test network locally will provide you a way to inspect how resources are being distributed and troubleshoot any unexpected behavior you may see from the quantum network. Examples of using SimulaQron may be found at `https://softwarequtech.github.io/SimulaQron/html/ExamplesNodeOS.html`.

After Development

We will additionally set up a simulated quantum network that will be hosted on AWS which you may use for your projects as well. To stick to the spirit of the Pan-European Hackathon, the simulated quantum network will represent a network formed by the nodes participating in the hackathon! Figure 2 shows what this looks like. In short, the quantum network is a complete graph where any pair of nodes may exchange qubits with one another.

Once you have played around with locally simulated quantum networks you will be able to test out your projects on our AWS-hosted simulated quantum network. To do so, you may speak to your local technical contact to request a configuration file for use with your project that will make all quantum network actions take place on this simulated network. You may place this configuration file in the correct location on your machine by opening a terminal and placing the configuration file in the location specified by the command simulaqron get network-config-file. This should swap any communications you expected to occur over your locally simulated network to communications between the AWS-hosted network.

If your project involves a client/server model we may also allocate an additional AWS instance for your project to host the server-side code should you wish. If this server-side code also uses resources in the quantum network make sure to set up SimulaQron and set the configuration files on the instance appropriately.

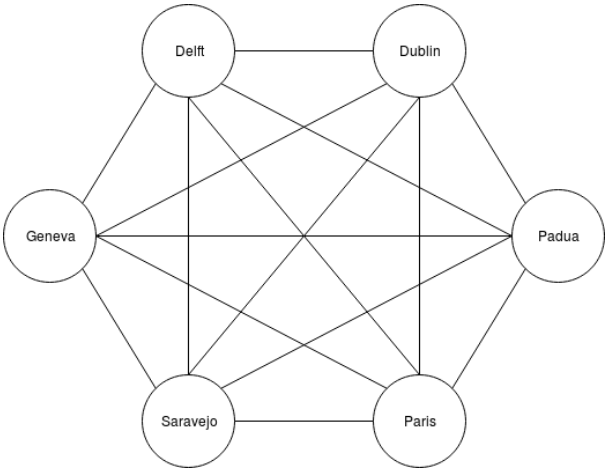

Figure 2: Fully connected simulated quantum network.

Quantum Information Reference

This portion of the document will provide a small reference for quantum information topics that may be useful when considering the design of your application.

Concept	**Meaning**
Bell pair	Pair of entangled qubits.
Entanglement between two qubits	Joint property of two qubits: the quantum state of these qubits cannot be described separatedly. It thus only makes sense to talk about the "joint state" of the two qubits.
EPR-pair	Pair of entangled qubits. Name comes from one of the initial papers on the topic, by Einstein, Podolsky and Rosen. Ironically, this paper argued against the existence of such entangled pairs.
Measurement/read-out of a qubit	Yields a single-bit. In general, this single bit is the outcome of a random biased coin flip, where the bias depends on the degree to which the chosen measurement basis and the quantum state of the qubit 'align'.
Measurement basis (Read-out basis)	"Measurement settings" using which one decides to measure/read-out the information that a quantum bit holds. Choice of measurement basis might strongly influence the outcome.
Qubit (quantum bit)	Equivalent of a classical bit. Can be in the 0-state or 1-state, or in a superposition of these.
Quantum bit error rate (QBER) in a given measurement basis	Probability that measuring two (entangled) qubits in the given measurement basis gives fully (anti)correlated outcomes. See also 'QBER' in the documentation `https://softwarequtech.github.io/SimulaQron/html/ExamplespythonLibQBER.html?highlight=qber`
Quantum gate	Operation that manipulates a quantum state. Equivalent of a logical gate in classical circuits.
Quantum Key Distribution (QKD)	Class of protocols for generating secure shared classical key using quantum bits. Famous QKD protocols include BB84 (Bennett-Brassard-1984) and Ekert91 (Ekert, 1991).
Quantum teleportation from node A to node B	Transferring a single qubit from node A to node B by consuming a shared entangled pair of qubits between them. See also sec. **??**
Tomography	Process of measuring many copies of the same qubit or set of qubits to find out its quantum state. See also 'tomography' in the documentation `https://softwarequtech.github.io/SimulaQron/html/SimulaQron.cqc.pythonLib.html?highlight=tomography#SimulaQron.cqc.pythonLib.cqc.CQCConnection.tomography`