QUANTUM INTERNET ALLIANCE

D4.2 Quantum Applications - Software Library Implementation Report

Document History

Revision Nr	Description	Author	Review	Date
1	Final Version	Marc Kaplan		17/04/2021

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 820445.

Index

1. Abstract

This document presents the Software Library of functions for quantum communication developed by Sorbonne University. This library provides atomic functions implementations for easing the development of quantum networking protocols. It provides functions in a simulation backend agnostic way thanks to a thin wrapping of their basic functionalities. The various atomic function implementations are dispatched in several submodules depending on their anticipated use.

2. Keyword list

Quantum network simulation, SimulaQron, Netsquid

3. Acronyms & Abbreviations

DoA	Description of Action
EC	European Commission
WP	Work Package

4. Introduction

Simulation of quantum networks allows to validate and benchmark quantum communication protocols. It is a crucial task for designing new applications for future use-cases of quantum networks. For this purpose, TU Delft has proposed two simulators with different advantages.

SimulaQron, available at www.simulaqron.org, can be used to validate protocols correctness quickly. Using its python interface, it allows fast development of distributed tasks with built-in functionalities for classical and quantum communication.

NetSquid, on the other hand, uses discrete event simulation to model physical parameters of the network. Its use is more involved, requiring to design modules for each physical component of the network. This complexity leads to simulations that are much closer to real-life situations. The simulator is available at netsquid.org.

In order to speed up the development of new protocols, we have built a library of quantum communication protocols elementary tasks. The basic idea is to implement a list of building blocks that can be easily combined to define new protocols. Furthermore, this library abstracts the operations, in the sense that it can be used with various backends, i.e. quantum network simulators.

This document presents the library in its current status. It is designed as an evolutive project with contributors potentially adding new features to the package. It can be installed following the instructions available at https://pypi.org/project/qpz-atomics/

5. Description of the library

We give an overview of the library as exposed on its website.

5.1. Module overview

The current submodules classification is:

- mapping: the thing wrapper around a simulation backend
- gate: everything that applies a fixed quantum unitary over 1 or 2 qubits
- prep: operations related to prepare given sets of quantum states
- meas: operations related to measure quantum subsystems according to some fixed POVM
- test: operations related to testing that one or several quantum subsystems have a given property
- util: useful operations that might not be directly related to a specific quantum operation but which are good to have
- tran: transport layer protocols which might not be readily available for all backends or which would need to include more robust implementations

Functionality tests are being developed for the atomic functions relying on the hypothesis package. The tests tend to follow the cryptographic approach of “correctness”. The “security” part is usually not performed as this most of the times involves checking indistinguishability of probability distributions.

5.2. Design principles

There exist many different quantum computing backends. The idea with this library was to abstract them away so that code running written using the library could be run on other backends, provided that the rest of the code not composed of functions defined by the library is not backend specific.

To do this, we instantiate the library by giving it a mapping and a node. The mapping is the translation of the backend specific way of calling elementary quantum operations, while the node is the actual quantum registers that are available to perform the computation. The node usually contains also some additional functions such as sending qubits to other nodes, receiving and sending entanglement etc. The differences have been abstracted away with the mappings for simulaqron and qunetsim . Other mappings have been considered and used but not yet made available most notably for Netsquid.

Users can add functions, or code new mappings by forking and pull-requesting insertion of new additions.

5.3. Usage

Examples can be found in the file examples/examples.py. The library is instantiated for each node (as if the nodes were independent computers, each loading its version of the library).

Other sources of inspirations are the tests defined in the tests directory

New atomic functions will be added following the list established by extracting atomic functions from the Quantum Protocol Zoo (wiki.veriqloud.com).

5.4. Testing

Tests can be run using python setup.py test at the root of the repository.

The repository includes a tests directory that contains the file test_qpz_atomics.py which gathers all the tests implemented. It is using the pytest package to launch the tests and gather statistics, while being based on hypothesis for generating examples.

For the tests to run, users need to have a quatum network simulator available and running. We have chosen to implement the tests using simulaqron as a backend, hence requiring a running simulaqron instance. This can be done typing the following:

simulaqron set max-qubits 100
simulaqron start

Other backends could be used provided the tests are rewritten and the required backend is available and properly mapped in the library.

6. Conclusion

We have presented the implementation of the software library. Its main goal is to fasten the development of new quantum communication tasks by providing elementary building blocks that can be combined together. Moreover, the library is design to be agnostic to the backend, which means it can be used with various quantum network simulators.

The library is open-source and designed to be evolutive. In the spirit of collaborative development, users are free to develop their own functionalities which can be merged to the library through a standard pull-request workflow.

In parallel to the construction of this library, VeriQloud has initiated a work on benchmarking quantum communication protocols. Currently, the use of NetSquid prevents using the atomic tasks since there is not yet a mapping to this backend. However, in the future, the software library will fasten the bencharking of new protocols.

VeriQloud's library can be found at https://github.com/LiaoChinTe/netsquid-simulation. The results of the benchmarking work will be presented in D4.3.

7. Appendix: list of functions

We present the list of atomic functions with their current status and planning of future deployment.

Name	Implementation	Doc string	Test	Submodule	Comment
Sending qubit (given by simulation backend)	DNE	OK	NA	simulaqron mapping	
Receive qubit (given by simulation backend)	DNE	OK	NA	simulaqron mapping	
Local Clifford Gates (CNOT, H, Pauli's)	DNE	OK	NA	simulaqron mapping	
Local non-Clifford Gates (T, Tinv)	DNE	OK	NA	simulaqron mapping	
CZ gate	DNE	OK	TDO	gate	self inverse, and corresponds to classically controlled Z for comp. basis control
CSWAP gate	DNE	OK	TDO	gate	
Creation Pauli eigenstates	DNE	OK	DNE	prep	
Creation and broadcast of n-party GHZ state	DNE	OK	DNE	prep	
Single qubit equatorial plane preparation	DNE	OK	TDO	prep	
Creation and broadcast of n-party stabilizer states	NXT			prep	
QFactory	LTR			prep	

Projections onto Pauli eigenstates	DNE	OK	OK	meas	
Single qubit equatorial plane measurement	TDO			meas	
Multi qubit POVM	LTR			meas	
Quantum One-Time-Pad encode / decode	DNE	OK	OK	pres	
BB84 encode / decode is a subset of QOTP encode / decode	NA	NA	NA	NA	
Collective BB84 encoding	DNG			pres	
Swap Test	DNE	OK	TDO	test	
Verification of stabilizer state				test	
Quantum RNG	DNE	Check	TDO	util	
Information reconciliation	LTR			util	
Classical error correction	LTR			util	
Quantum error correction	LTR			util	
Privacy amplification	LTR			util	
Quantum one-way function	NXT			util	
Weak string erasure	NXT				
Teleportation send	TDO			tran	
Teleportation receive	TDO			tran	
Quantum authenticated channel	LTR				
Secure classical broadcast channel	LTR				

Classical authenticated channel	LTR				