修士論文

東京大学大学院情報理工学系研究科電子情報学専攻

Design and Evaluation of Communication Protocols for Quantum Repeater Networks

量子リピーターネットワーク向け通信プロトコルの設計と評価

指導教員: 江崎浩 教授

Aparicio Luciano
アパリシオ・ルシアノ

学生証番号: 48-096448

２０１１年８月１７日

Abstract

When built, quantum repeater networks will require classical network protocols to control the quantum operations. However, existing work on repeaters has focused on the quantum operations themselves, with less attention paid to the contents, semantics, ordering and reliability of the classical control messages. In this work we define and describe our implementation of the classical control protocols. The state machines and packet sequences for the three protocol layers are presented, and operation confirmed by running the protocols over simulations of the physical network. We also show that proper management of the resources in a bottleneck link allows the aggregate throughput of two end-to-end flows to substantially exceed that of a single flow. Our layered architectural framework will support independent evolution of the separate protocol layers.

Networks of quantum repeaters utilize three concepts to execute a distributed algorithm that creates *entangled* quantum states between nodes that are far apart: a basic *entanglement mechanism* which depends on the physical implementation, error management (in this work, we study a method known as *purification*), and finally a quantum state propagation layer (here we implement *entanglement swapping*, which builds multi-hop connections from single-hop connections). Some researchers are investigating approaches that are substantially different from entanglement swapping [30, 24, 19]. Here we focus on swapping, but the layered architecture approach is broadly applicable, allowing other implementations to replace only a single layer in the protocol stack.

Previous work primarily focused on the physical and mathematical tools for building repeaters. Classical information is also needed to enable *teleportation* and swapping, as many quantum operations are not deterministic, and results of quantum measurements need to be reported to distant partners before further operations can proceed. Also, operations in the middle of the network must be coordinated to route and swap properly. This requires classical messages to make operations robust, but message propagation times penalize performance. Even though this delay is usually included in repeater simulations, prior work has not defined the protocols in detail, especially with respect to how all of the nodes make consistent decisions in a timely fashion.

In this work, we introduce a protocol stack for networks of quantum repeaters that considers all the necessary classical messages and which can be easily adapted for different approaches at all three protocol layers. We run simulations of competing flows on a dumbbell topology in order to increase our confidence in the behavior of our network protocols. By adjusting the fidelity thresholds required for entanglement swapping, we show that some configurations boost the aggregate throughput for multiple flows significantly above the maximum for a single flow, taking advantage of resources that would otherwise sit idle. The operation of such complex networks and such delicate tuning of the system without formal protocol definitions would not be possible.

Previous work has also concentrated almost exclusively on the dedicated use of a single line or chain of repeaters, delivering Bell pairs only to the two nodes at the ends of the chain. Networks, however, typically have more than two end points, and allow any pair of these end nodes to communicate. More topologically complex networks, with numerous end nodes, are obviously much more scalable than connecting every possible pair of nodes using a dedicated line of repeaters.

we investigate how classical multiplexing schemes translate to the domain of quantum repeaters, in order to manage shared resources and active communication flows of data from different stations. We simulate four sharing protocols in a complex network with competing traffic. *Circuit switching* gives any individual flow the best performance, but makes poor use of the overall network and is inflexible. We use circuit switching as our baseline case to compare *time division multiplexing*, *buffer space multiplexing*, and *statistical multiplexing*, and show that all multiplexing schemes are better than circuit switching. For the particular network simulated, we find that statistical multiplexing outperforms time division multiplexing by 28% and buffer space multiplexing by 13%. We find that all three multiplexing schemes are *fair*; each flow is penalized a similar percentage of its throughput as the the total number of users in the network increases. Finally, statistical multiplexing requires

no network-wide coordination of the use of quantum memory or channels, and is easier to implement than a robust, scalable scheme for the other protocols. Our current results suggest that the best strategy for quantum repeater networks is statistical multiplexing. (This is in fact the quantum analogue of the basis on which the Internet works.) However, our current simulations are done with no degradation of memory over time, using the assumption of quantum error-corrected memory at each repeater; inclusion of decoherence and a finite qubit lifetime remains as future work.

Contents

Abstract **1**

Acknowledgements **9**

1 Introduction **10**
1.1 Motivation . . . 10
1.2 Contributions . . . 11
1.3 Structure of thesis . . . 11

2 Background on Quantum Information Science **12**
2.1 What is a quantum state? . . . 12
2.2 Quantum properties . . . 12
2.2.1 Superposition . . . 12
2.2.2 Interference . . . 13
2.2.3 Entanglement . . . 13
2.2.4 Measurement . . . 13
2.3 How to describe quantum states? . . . 13
2.3.1 Pure states and Mixed states . . . 13
2.3.2 State Vector . . . 13
2.3.3 Density Matrix Representation . . . 14
2.4 Qubits . . . 14
2.4.1 Physical representations . . . 14
2.4.2 Physical technologies . . . 15
2.5 Manipulating quantum states . . . 15
2.6 Entanglement . . . 17
2.6.1 Fidelity . . . 17
2.6.2 Bell Pairs . . . 18
2.6.3 Bell States . . . 18
2.7 Teleportation . . . 18

3 Background on Quantum Repeaters **23**
3.1 Introduction . . . 23
3.2 Physical entanglement . . . 24
3.3 Purification . . . 24
3.4 Entanglement Swapping . . . 25
3.4.1 Swapping . . . 25
3.4.2 Graphical understanding of Purification and Swapping . . . 26
3.5 Quantum Memory . . . 28
3.6 Basic Repeater Operation . . . 28
3.7 Multi-User Quantum Repeater Networks . . . 29
3.8 Applications . . . 29

4 Quantum Networks **30**
4.1 What do networks do? . . . 30
4.2 Changing network topology . . . 30
4.3 Dynamically changing network state . . . 31
4.4 Fault tolerance . . . 31
4.5 Solving these problems . . . 32
4.5.1 Protocols . . . 32
4.5.2 Resource management . . . 32
4.6 Metrics for Success in Networks . . . 32

5 Protocol Design **33**
5.1 Quantum Network Protocol . . . 33
5.1.1 Comparison with the OSI model . . . 34
5.2 Inter-node Message structure . . . 35
5.3 Physical Layer . . . 35
5.3.1 Introduction . . . 35
5.3.2 Messages . . . 36
5.4 Acknowledged Entanglement Control Layer . . . 37
5.4.1 Finite State Machine . . . 37
5.4.2 Messages . . . 39
5.5 Purification Layer . . . 40
5.5.1 Finite State Machine . . . 41
5.5.2 Messages . . . 41
5.5.3 Purification Policy . . . 43
5.6 Entanglement Swapping Control Layer . . . 43
5.6.1 Finite State Machine . . . 44
5.6.2 Messages . . . 45
5.7 Application Layer . . . 46
5.8 Quantum Repeater Multiplexing . . . 47
5.8.1 Quantum Circuit Switching . . . 47
5.8.2 Time Division Multiplexing (TDM) . . . 48
5.8.3 Buffer Space Multiplexing . . . 49
5.8.4 Statistical Multiplexing . . . 49
5.8.5 Aggressive Use of Resources . . . 50

6 Quantum Network Simulator **51**
6.1 Introduction . . . 51
6.2 Simulating Distributed Quantum States . . . 51
6.3 Why Omnet++? . . . 51
6.3.1 Omnet++ general description . . . 51
6.3.2 Omnet Operation . . . 52
6.4 Organization of Simulator Files . . . 54
6.4.1 Topology Design (NED files) . . . 54
6.4.2 Messages Design (MSG files) . . . 54
6.4.3 Configuration file (INI file) . . . 54
6.5 Layers Implementation . . . 54
6.5.1 Interlayer Messages . . . 54
6.5.2 Physical Layer . . . 56
6.5.3 Acknowledged Entanglement Control Layer . . . 57
6.5.4 Purification Layer . . . 58
6.5.5 Entanglement Swapping Control Layer . . . 58
6.5.6 Application Layer . . . 58
6.6 Main Code Description . . . 58
6.7 Key differences between proposed protocols and actual simulations . . . 59

7 Evaluation **61**
7.1 Multiple hops . . . 61
7.2 Shared resources . . . 61
7.2.1 Circuit Switching . . . 63
7.2.2 Statistical Multiplexing . . . 64
7.2.3 TDM . . . 64
7.2.4 Buffer Space Multiplexing . . . 64
7.2.5 Discussion . . . 65

8 Conclusions **69**

List of Figures

2.1 1-Qubit quantum gates . . . 16
2.2 C-NOT Gate. . . . 17
2.3 Controlled-Z Gate. . . . 17
2.4 Teleportation Circuit . . . 18
2.5 Teleportation Circuit - Qubit we wish to teleport . . . 20
2.6 Teleportation Circuit - Bell pair available . . . 20
2.7 Teleportation Circuit - Interaction of Bell pair and data qubit . . . 20
2.8 Teleportation Circuit - Measurement of qubits . . . 21
2.9 Teleportation Circuit - Send measurement results to Bob . . . 21
2.10 Teleportation Circuit - Apply operations to Bob's qubit . . . 21
2.11 Teleportation Circuit - Bob's qubit turns into the data qubit . . . 22

3.1 Purification. . . . 25
3.2 Swapping. . . . 26
3.3 Purification & Swapping - Sequence 1 . . . 27
3.4 Purification & Swapping - Sequence 2 . . . 27
3.5 The Dumbbell network . . . 29

5.1 Protocol Stack Architecture . . . 33
5.2 Protocol Stack with many stations . . . 34
5.3 Encapsulation header . . . 35
5.4 Data structure defined for the Physical Layer . . . 36
5.5 Messages exchanged in Physical Layer . . . 36
5.6 Message type exchanged in Physical Layer . . . 37
5.7 Finite state machine for Transmitter's ACKed Entanglement Control . . . 38
5.8 Finite state machine for Receiver's ACKed Entanglement Control . . . 38
5.9 Data structure defined for the Entanglement Control Layer . . . 39
5.10 Messages exchanged in AEC Layer . . . 39
5.11 Message type exchanged in Acknowledged Entanglement Control Layer . . . 40
5.12 Handshake of messages between 2 neighbor stations during entanglement attempt . . . 40
5.13 Finite state machine for Purification Control . . . 41
5.14 Message type exchanged in Purification Control Layer . . . 42
5.15 Messages exchanged in Purification Layer . . . 42
5.16 Handshake of messages between 2 neighbor stations during purification . . . 43
5.17 Handshake of messages between two neighboring stations during purification, after entanglement 43
5.18 Finite State Machine for Entanglement Swapping Control for a middle node . . . 44
5.19 Finite state machine for Entanglement Swapping Control for an end node . . . 44
5.20 Message type exchanged in Entanglement Swapping Control Layer . . . 45
5.21 Messages exchanged in Entanglement Swapping Control Layer . . . 46
5.22 Reports sent in Entanglement Swapping Control Layer . . . 46
5.23 Messages exchanged in the Application Layer . . . 47
5.24 Circuit Switching. A to B enabled. . . . 48
5.25 Circuit Switching. C to D enabled. . . . 48

5.26 Time division multiplexing of two communication flows 49
5.27 Buffer Space Multiplexing. 49

6.1 Excerpt of Dumbbell.ned 55
6.2 Dumbbell network in Omnet 56
6.3 QubusPE.msg 56
6.4 Excerpt of Omnet.ini 57
6.5 Message type exchanged betwen layers 57
6.6 Functions of the Physical Layer 57
6.7 Pseudo-code for the function handleMessage 60

7.1 Simulated dumbbell network 61
7.2 Throughput in Bell pairs/sec, one flow versus two flows 62
7.3 End-to-end fidelity of teleported qubits 62
7.4 Simulated network 63
7.5 Throughput of statistical multiplexing 66
7.6 Throughput of TDM 67
7.7 Throughput of buffer space multiplexing compared to uncontested flows. 68

List of Tables

5.1 Proposed protocol type codes 36

6.1 Run-time Omnet configuration files in the simulator 52
6.2 Classes defined in the simulator 52
6.3 Omnet message files in the simulator, used during compilation of simulator. 53
6.4 Qubit states for Qubus and Entanglement Control Layer 53
6.5 Qubit states for Purification Control Layer 53
6.6 Qubit states for Entanglement Swapping Control Layer 54
6.7 Interlayer protocol codes 56

7.1 Hardware-Configuration 63
7.2 Traffic flows 64
7.3 Maximum traffic per flow using circuit switching. 64
7.4 Throughput using statistical multiplexing 65
7.5 Throughput using Time Division Multiplexing 65
7.6 Throughput using buffer space multiplexing 68

Acknowledgements

First and foremost I want to thank my advisor Esaki Sensei for allowing me to do research in quantum repeater networks eventhough it was not a research topic in our laboratory. Since I came to Japan not only has he given me academic advice but also personal and professional guidance.

Secondly, I want to give my sincerest thanks to Dr. Rodney Van Meter who has introduced me to quantum information science, quantum repeaters and quantum networks. He always supported and guided me from the beginning, given me advice on my research and motivated my to publish our results. Without him and his guidance, this thesis would have not been written. I do not only consider him a great advisor but also a very good friend.

In regards to the AQUA (Advancing Quantum Architecture) team, I want to thank Dr. Horsman who has always supported me on the physics of quantum mechanics and quantum information science to write papers and to improve my knowledgement on these fields. Also to Shota Nagayama and Takahiko Satoh who also supported me with Japanese translations, and became good friends.

I want to thank Dr. Bill Munro for his guidance and discussions we had regarding my research.

I am thankful for funding from MEXT which allowed me to afford the tuition fees and the expenses of living in Japan.

I want to thank to Carlos Felipe Santacruz for all his help during my first steps in C++ and LaTeX and his invaluable frienship. Also to Giancarlos Troncoso with whom we shared unforgettable vacations and our attempts to study Japanese together.

During my stay in Japan, I made many friends from Esaki-lab, sharing many experiences and trips. I also received their help to translate some emails or other information from Japanese to English.

I also met many Latin American friends, with whom we shared similar experiences while living in Japan, and they were always supporting me and making me feel like if I were at home.

Finally I thank my family and friends for their support and encouragement from Argentina.

Chapter 1

Introduction

1.1 Motivation

Classical computers have been evolving for several years, and according to Moore's law, every 24 months computers reduce their size by a half, becoming faster and smaller. If this tendency were kept, in 10 years transistors will reach a level of few atoms, therefore quantum effects must be considered. Recent works says that now the rate has changed to 36 months [18]. One emerging techonology is quantum computers which take advantage of the quantum effects and use them for computation and to solve some problems may be faster than their classical counterparts. Theorists have been developing some possible applications for quantum computers, while experimentalists are trying to produce very basic quantum operations with few *qubits*. Some of the applications for quantum computers require distributed computation along *quantum networks* [20, 17, 2, 5, 13], or monolithic making use of a single quantum computer. These applications can be numerical (as classical computers manipulate digital data) or they can be physical (manipulation of *quantum states* that represent some physical property).

At the moment of writing this thesis, the only existing commercial application is QKD (Quantum Key Distribution) [20, 17] which is a very well known algorithm to distribute encryption keys in a secure manner. Built devices are limited by the distance between endpoints (up to 100km). For further distances it is necessary to *repeat the signal* somehow, in order to extend the range of the application. One proposal for this is a n*etwork of quantum repeaters*.

Richard P. Feynman suggested in 1982 that classical computers cannot simulate quantum systems in an efficient manner, and simulating large ones is very difficult to implement in a short time. Example of these quantum systems could be from different areas like physics, chemistry and biology (for big molecules). This would probably help scientists to improve the performance of their simulations and also extend them to bigger systems, currently impossible to simulate. Previous works have proposed numerical simulations [4] and direct physical simulations [6] to be run on quantum computers.

Quantum computers got a lot of interest from the scientific community after Shor's algorithm [35], as a proposal to factorize numbers with a much higher performance than classical computers. Factorizing big numbers is a hard task for classical computers and today's asymmetric encryption is based on this principle. Searching in a list of N elements with classical computers without any knowledge that can help reduce the searching time, will take and average of $\frac{N}{2}$ searches. Quantum search algorithms, based on Grover's algorithm [22], offers a quadratic speedup for searching heuristics.

Some applications will help improve performance of physical experiments and applications. In infrared and optical interferometer arrays, photons need to be brought together for the interference measurement. Longer-baseline telescopes using Quantum repeaters [21] will allow to transport the photons reducing losses and phase changes, improving the resolution while increasing the sensitivity. Another example is LIGO: The Laser Interferometer Gravitational-Wave Observatory [1], where sensitivity is also expected to be improved. A distributed quantum algorithm has also been proposed that will synchronize clocks to better-than-atomic-clock precision over a distance [25, 10].

Keeping many qubits together in the same computer without being affected by noise may be a challenge even in the future, therefore the use of distributed computation will allow tasks that require many qubits that could not be allocated in the same computer, to be executed. One Example of such solution is Recursive quantum

repeater networks [39].

1.2 Contributions

Our work is focused on quantum networks architecture and is placed between theorist and experimentalist researchers' work. We provide a protocol architecture for networks of quantum repeaters. A layered architecture that will allow the implementation of quantum network protocols and interconnectivity between different physical technologies. We defined each protocol's functions, finite state machines, interaction between layers and messages used between nodes. The architecture was designed so that future implementations of the current protocols or even the addition of new protocols could be supported. These protocols transport all the classical information that is needed to control and operate quantum networks. They proved to be scalable and provide the structure for future work. As it will be shown in next sections, quantum network protocols must be fast as *quantum states* deteriorate with time. So, nodes should make the same decisions like others, but without exchanging unnecessary packets, as the delay introduced by the arrival of these messages will affect the network performance. This is a very hard task that all the protocols must consider.

In this thesis we developed a simulator for quantum networks where we tested our set of protocols. We simulated two networks. In the first one, a dumbbell network, we studied how tuning uncontested links differently from contested ones resulted in a net gain in performance (total traffic was doubled). We also tested a more complex topology where many flows were competing for the networks resources in different parts of the network. We applied three multiplexing schemes, to manage the resources, which were compared with the traffic that each flow was able to obtain without any other competing flows. The behavior of the multiplexing schemes and our protocols proved to be fair to all the flows, having values of fairness between 0.97 and 0.99, being 1 is an equal distribution of resources.

We simulated a thirteen-node network with up to five flows sharing different parts of the network, measuring the total throughput and fairness for each case. Our results suggest that the Internet-like approach of statistical multiplexing use of a contested link gives the highest aggregate throughput. Time division multiplexing and buffer space multiplexing were slightly less effective, but all three schemes allow the sum of multiple flows to substantially exceed that of any one flow, improving over circuit switching by taking advantage of resources that are forced to remain idle in circuit switching. All three schemes proved to have excellent fairness. The high performance, fairness and simplicity of implementation support a recommendation of statistical multiplexing for shared quantum repeater networks.

In the future, such code can be used to write the software that will control quantum repeaters.

1.3 Structure of thesis

Chapter 2 provides a background on quantum information science, followed by chapter 3 with a background on quantum repeaters. Chapter 4 introduces quantum networks and provides the problem statement for this thesis. Chapter 5 is the protocol design and the main work. Chapter 6 describes the simulator written in Omnet++, where all the simulations for this work were done. Chapter 7 is the evaluation, where we run many simulations and chapter 8 are the conclusions of this work.

Chapter 2

Background on Quantum Information Science

2.1 What is a quantum state?

In quantum mechanics, quantum states completely define a quantum system. Erwin Schrödinger proposed an equation to describe quantum systems. Here we show the Hamiltonian version of such equation which uses state vectors:

$$H|\psi(t)\rangle = i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle$$

if H is independent of time:

$$|\psi(t)\rangle = e^{-\frac{iHt}{\hbar}}|\psi(0)\rangle$$

and

$$U = e^{-\frac{iHt}{\hbar}}$$

$|\ \rangle$ (called Ket) is the Dirac notation to represent vectors. H is the Hamiltonian of the system, and is an $N \times N$ matrix if the system has N basis states (note that this is different from the Hadamard gate, also written H, introduced below), $|\psi(t)\rangle$ is the quantum state at a time t, and $\hbar$ is the reduced Planck constant. The Hamiltonian operator represents the environmental influences that affect the state of the quantum system, such as local magnetic fields. The unitary operator U, usually evaluated for a specific amount of time, is an easier-to-use form, allowing us to write $|\psi'\rangle = U|\psi\rangle$ to represent many simple changes to the system. The solutions of this equation may have imaginary coefficients, which is a main difference between classical and quantum systems.

2.2 Quantum properties

2.2.1 Superposition

Quantum systems described by a quantum state $|\psi\rangle$ can be in a superposition of different quantum states:

$$|\psi\rangle = \sum_{i=1}^{n} \alpha_i|\psi_i\rangle$$

where $|\psi_i\rangle$ are the quantum states and α_i are the amplitudes of each state, being $|\alpha_i|^2$ the probability of measuring the $|\psi\rangle_i$ state.

As it will be explained later in this chapter, quantum states can be described as a superposition of some chosen basis. If each quantum in the system can take k possible states, then n of these quanta have k^n basis

states, and the system may be in a superposition of any or all of them. Proper manipulation of such superposition can lead to quantum computing. Superposition is kept until the particles are measured.

2.2.2 Interference

Quantum states can interfere being possible to obtain a constructive interference where the amplitudes sum, or destructive interference where the amplitudes cancel. The strenght of quantum computing comes from the interference of quantum states, which is done by interfering the phases and amplitudes of the different quantum states until the desired computing is obtained.

2.2.3 Entanglement

Entanglement happens when two different particles interact in such a way that, each particle is described by a single state that cannot be decomposed into states on the individual systems. This means, that their measurement outcomes can be random but correlated (in classical systems, correlation may also ocurr but in quantum mechanics is much stronger). This correlation is called entanglement. Entanglement is not restricted to a pair of qubits but to any number of them. However, in this thesis focusing on quantum networks, entanglement is produced only between a pair of qubits.

2.2.4 Measurement

As we mentioned before, quantum states can be in a superposition of states. This situation remains until we measure in some basis (any chosen set of vectors that fully describes the Hilbert space of the quantum system), destroying the superposition and projecting the quantum state into one of these basis' vectors. Therefore, when we do the measurement we will always measure one of the basis only and any other information kept before is lost.

2.3 How to describe quantum states?

2.3.1 Pure states and Mixed states

Pure states are states which cannot be described as a mixture of other quantum states. On the other hand, mixed states are a statistical mixture of pure states. While pure states are clean, mixed states are noisy and are in some pure state but we don't know which one.

2.3.2 State Vector

State vectors are used to describe any isolated physical system composed only of *pure states*. Any arbitrary state vector of a two-state system can be described as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

where

$$|\alpha|^2 + |\beta|^2 = 1$$

α and β are complex numbers, being $|\alpha|^2$ the probability of measuring $|0\rangle$, and $|\beta|^2$ the probability of measuring $|1\rangle$. $|0\rangle$ and $|1\rangle$ are the basis state vectors. They are usually treated as mathematical abstractions, but in the real world they correspond to the basis states of a two-state system, such as the up and down spin of an electron (or any other spin 1/2 phenomenon), or the horizontal and vertical polarization of a photon.

2.3.3 Density Matrix Representation

In case we don't want to limit our representation to pure states, and the system we want to describe includes mixed states, we need to use a matrix representation known as *Density Matrix Representation.* This notation is used for both pure states and mixed states. State vectors can also be represented by a density matrix which is an Hermitian, positive operator. This representation is easier to compute expectation values of physical properties of our system. The elements on the diagonal of this matrix must be real and non-negative. The trace of the density matrix must always be 1. By definition, the density matrix ρ is defined as:

$$\rho \equiv \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

where p_i are the probabilities of the system being in the quantum state $|\psi_i\rangle$
For pure states, the trace of the square of the density matrix equals 1:

$$Tr(\rho^2) = 1$$

For mixed states the trace is less than 1:

$$Tr(\rho^2) < 1$$

For example, we show how to represent the density matrix of the quantum state represented previously as a state vector:

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \begin{pmatrix} \alpha^* & \beta^* \end{pmatrix} = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}$$

2.4 Qubits

As classical computation and classical information science are based on bits as elementary units, quantum computation and quantum information are based on quantum bits or qubits. Classical bits are known to be on only one of these two possible states "0" and "1". Qubits, on the other hand, have two possible states which are vectors and can be represented as $|0\rangle$ and $|1\rangle$. But the main difference is that qubits can be in a superposition of both states until they are measured. Qubits can be represented by state vectors or by density matrixes.

The state of a qubit is a vector in a two-dimsensional complex vector space. According to the state vector notation explained before, the special states $|0\rangle$ and $|1\rangle$ are known as computational basis states, and form an orthonormal basis for this vector space. A system of n qubits has 2^n basis states, corresponding to each of the n-bit integers 00..0 to 11..1. Classical bits are easily measured and one can determine with certainty the value of them and produce as many copies as necessary. However, qubits' states cannot be measured. This means, we are not able to measure the values of α and β. Quantum mechanics restricts the measurement, and we can only tell that we will measure $|0\rangle$ with a probability of $|\alpha|^2$ and $|1\rangle$ with a probability of $|\beta|^2$.

A qubit can be in a continuum of states between $|0\rangle$ and $|1\rangle$ until it is observed. The non-cloning theorem shows that there is no unitary operator that can be applied to a qubit and produces a copy of it. One alternative that can be used for communications is based on the teleportation algorithm which will be explained later in Sec. 2.7. Once physically implemented, qubits can be classified as flying qubits or stationary qubits. Flying qubits refers to qubits that are no static, like photons. On the other hand, stationary qubits have no movement and their positions are fixed.

2.4.1 Physical representations

Every proposed carrier for a qubit must have two orthogonal states that can be used by convention as our $|0\rangle$ and $|1\rangle$ states. Here we list a few:

- **photon polarization:** The linear polarization of a photon can be vertical or horizontal; circular polarization can be left-circular (counter-clockwise) or right-circular (clockwise). We can use either pair of states as our $|0\rangle$ and $|1\rangle$ basis.

- **electron spin:** The spin of an electron can be up or down corresponding to aligned or anti-aligned with a reference magnetic field.

- **nuclear spin:** The nuclear spin can be up or down corresponding to aligned or anti-aligned with a reference magnetic field.
- **energy level:** Energy levels can be in an excited state or in the ground state.
- **presence / absence:** The presence or absence of a photon or electron in a cavity.
- **position:** Left/right quantum dot, Left/right photon path.

2.4.2 Physical technologies

The qubit carriers above can be implemented in a variety of different technologies; here, we briefly identify a handful. Note that some of the basic technologies can be used in conjuction with more than one physical carrier.

- **quantum dot:** Potential well, traps electron spin, energy level, presence, position.
- **optical lab bench:** Using beam splitters, mirrors, parametric down conversion (PDC) devices.
- **nanophotonics:** Waveguides and beam splitters are fabricated in semiconductors using VLSI-like techniques.
- **ion trap:** Energy level.
- **cavity QED:** Quantum dots, individual atoms.

2.5 Manipulating quantum states

In order to do quantum computing we need to be able to manipulate the quantum states. In analogy to classical computing, calculation can be done via quantum gates, which are represented using unitary operators, as described above in the discussion of the Schrodinger equation. A gate that acts on n qubits is represented by a $2^n \times 2^n$ unitary. The state of a single qubit can be represented as a point on a unit sphere (known as the Bloch sphere), with $|0\rangle$ being the Z axis, $(|0\rangle + |1\rangle)/\sqrt{2}$ being the X axis, and $(|0\rangle + i|1\rangle)/\sqrt{2}$ being the Y axis. Single-qubit gates can be thought of as rotations about an axis on this sphere. Conventionally, we restrict ourselves to rotations about X, Y, or Z. Although rotation through any arbitrary angle θ is possible, in this thesis we will only need to concern ourselves with a specific set of gates. Following, we show single-qubit rotations gates:

For example, the quantum NOT, also known as Pauli-X, has a behaviour like:

$$\alpha|0\rangle + \beta|1\rangle \Rightarrow \alpha|1\rangle + \beta|0\rangle$$

the matrix representation for this quantum gate is:

$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

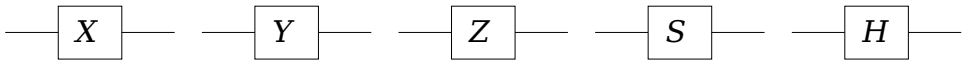
and the result of applying this gate to a general quantum state is written like:

$$X \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

and can also be written as:

$$X|\psi\rangle \Rightarrow |\psi'\rangle$$

its graphical representation is shown in Fig. 2.1(a)



(a) Pauli-X (b) Pauli-Y (c) Pauli-Z (d) S-Gate (e) Hadamard

Figure 2.1: 1-Qubit quantum gates

Other quantum gates are:
Pauli-Y, which is shown in Fig. 2.1(b), is described as follows:

$$Y \equiv \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Pauli-Z, which is shown in Fig. 2.1(c), is described as follows:

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Phase, which is shown in Fig. 2.1(d), is described as follows:

$$S \equiv \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$$

The Hadamard gate, which is shown in Fig. 2.1(e), is described as follows:

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

If we analyze the last one, it can be seen that if we have the states $|0\rangle$ or $|1\rangle$ and the Hadamard gate is applied to them, we obtain the following results:

$$H|0\rangle = H \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

$$H|1\rangle = H \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{|0\rangle - |1\rangle}{\sqrt{2}}$$

It is clear that for both cases we obtain a superposition of states. As it will be explained in Sec. 2.7, if we start with two qubits initialized in $|0\rangle$, and the first one is applied the Hadamard gate, we use this output to control a *C-NOT gate* which is applied to the second qubit. The result of these operations is the Bell pair $|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$.

Single-qubit gates are needed but they are not enough universal quantum computation. Now we show some examples of two-qubit gates.

Controlled-NOT (C-NOT): This gate applies the NOT gate (Pauli-X) to one qubit if the other one is in $|1\rangle$. Otherwise, the first qubit remains untouched. Fig. 2.2

The mathematical representation of such a gate is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

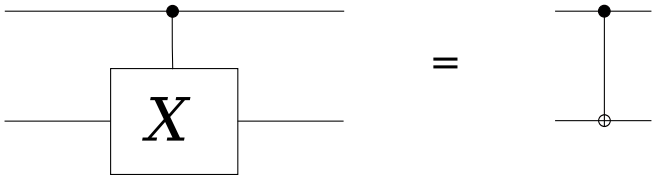


Figure 2.2: C-NOT Gate.

Finally, we show controlled-Z in Fig. 2.3

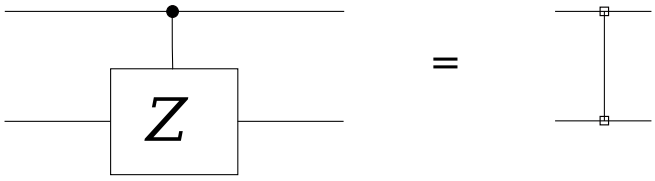


Figure 2.3: Controlled-Z Gate.

The mathematical representation is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

2.6 Entanglement

If a quantum system is described as:

$$|\psi\rangle_{AB} = \sum_{i,j} c_{ij} |i\rangle_A \otimes |j\rangle_B$$

where $|i\rangle_A$ is a basis that describes system A, $|j\rangle_B$ is a basis that describes system B and $\otimes$ is the tensor product. The system is said to be entangled if:

$$c_{ij} \neq c_i^A c_j^B$$

In such a case, we **cannot** describe each system A and B separately like this:

$$|\psi\rangle_A = \sum_i c_i^A |i\rangle_A$$

and

$$|\psi\rangle_B = \sum_j c_j^B |i\rangle_B$$

meaning that both systems are correlated, and they are not independent from each other.

2.6.1 Fidelity

Is a measure of distance between quantum states. It can be seen as how far a quantum state ρ_2 is from a target state ρ_1 :

$$F = \sqrt{\rho_2^{1/2} \rho_1 \rho_2^{1/2}}$$

In this thesis, the fidelity is calculated in an ensemble of mixed states. If we define as the fidelity F of the state $|\psi_1\rangle$ which is mixed with the state $|\psi_2\rangle$. The state can be defined as:

$$\rho = F|\psi_1\rangle\langle\psi_1| + (1-F)|\psi_2\rangle\langle\psi_2|$$

As it can be seen, the probability of measuring the state $|\psi_1\rangle$ is F. If F is close to 1, then we have a high-fidelity quantum state.

2.6.2 Bell Pairs

When two qubits are maximally entangled (high fidelity) they are called Bell pairs. Therefore, Bell pairs are two correlated qubits which quantum states cannot be independent from each other. They are fundamental resources for the teleportation of quantum states, which is needed to build quantum repeaters. The possible states of these pairs are called Bell states.

2.6.3 Bell States

For two entangled qubits, four vectors can fully describe the quantum system. In this thesis, we use a set of basis vectors, called Bell states, which is shown next:

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

$$|\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}}$$

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

$$|\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}$$

2.7 Teleportation

As has been explained before, every time a quantum state is measured, the state collapses into one of the measuring basis states. Quantum mechanics will not allow us to determine the exact quantum state before measurement, and there are no quantum operators that can duplicate any quantum state. However, if we need to move one qubit from one place to another, there are some quantum circuits that allow an operation called teleportation, in which the quantum state of a qubit is teleported from one qubit to another [3]. But, as this operation requires to measure the original qubit, it's state will be destroyed and we will only be able to obtain the same quantum state teleported into another qubit, usually far away. The main purpose of this, as it will be explained later, is that it is very difficult to transport a qubit without affecting its state. Therefore, instead of transporting qubits, we can teleport them with the circuit in Fig. 2.4.

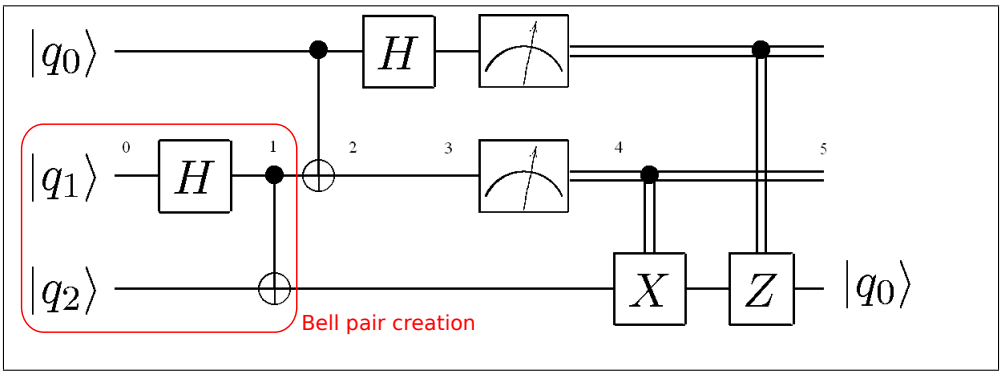


Figure 2.4: Teleportation Circuit - The Hadamard and CNOT that make the Bell pair can be replaced with any mechanism that creates a Bell pair over a distance, such as Qubus [36].

In this figure, q_0 and q_1 are two qubits in the transmitter. q_2 is another qubit but in the receptor. The quantum state that we want to teleport is $|q_0\rangle$. q_1 and q_2 become a Bell pair after the first Hadamard gate and CNOT gate. Here, we explain how this process works, where ψ is the general state considering the three qubits.

$$|\psi_0\rangle = |q_0\rangle|q_1\rangle|q_2\rangle$$

But for the initial conditions,

$$|q_0\rangle = \alpha|0\rangle + \beta|1\rangle$$
$$|q_1\rangle = |0\rangle$$
$$|q_2\rangle = |0\rangle$$

So

$$|\psi_0\rangle = [\alpha|0\rangle + \beta|1\rangle]|0\rangle|0\rangle$$

After the first Hadamard and CNOT gate we obtain:

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}[(\alpha|0\rangle + \beta|1\rangle)(|00\rangle + |11\rangle)] \Rightarrow$$

$$|\psi_1\rangle = \frac{1}{\sqrt{2}}[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|00\rangle + |11\rangle)]$$

After the second CNOT gate, we obtain:

$$|\psi_2\rangle = \frac{1}{\sqrt{2}}[\alpha|0\rangle(|00\rangle + |11\rangle) + \beta|1\rangle(|10\rangle + |01\rangle)]$$

Now $|q_0\rangle$ goes through the second Hadamard gate:

$$|\psi_3\rangle = \frac{1}{2}[\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle)]$$

This last state can be written in a different way, like this:

$$|\psi_3\rangle = \frac{1}{2}[|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)]$$

The coefficients of each of the four terms represent the quantum states $|q_0\rangle$ and $|q_1\rangle$. We now define $|q_3\rangle$ composed by the terms that are between parenthesis. Based on the possible values of $|q_0\rangle$ $|q_1\rangle$ (which are the ones we measure) , we can make a table like this:

$$00 \longmapsto [\alpha|0\rangle + \beta|1\rangle]$$
$$01 \longmapsto [\alpha|1\rangle + \beta|0\rangle]$$
$$10 \longmapsto [\alpha|0\rangle - \beta|1\rangle]$$
$$11 \longmapsto [\alpha|1\rangle - \beta|0\rangle]$$

These measured qubits produce two bits of classical information which are sent to the receiver. If these bits are 00, we would have the first case which belongs to the state $|q_0\rangle$ without applying any operation to $|q_3\rangle$. For 01 we can see that the states are inverted, so we need to apply a CNOT gate (X-Gate) to $|q_3\rangle$. For 10 we need to change the sign of $|1\rangle$, so a Z-Gate applied to $|q_3\rangle$ will do it. And finally, for 11, we need to apply both X-Gate and Z-Gate in order to restore $|q_0\rangle$ from $|q_3\rangle$. As we can see from this process, the quantum state is teleported to $|q_3\rangle$, but $|q_0\rangle$'s state is destroyed after measurement.

To explain this in a graphical manner, we show in the next sequence of pictures what are the differente operations that need to be done in order to teleport one qubit of information to a remote station.

In Fig. 2.5 we show two stations Alice and Bob. Alice holds one data qubit (colored in green) which is the piece of information that we want to teleport.

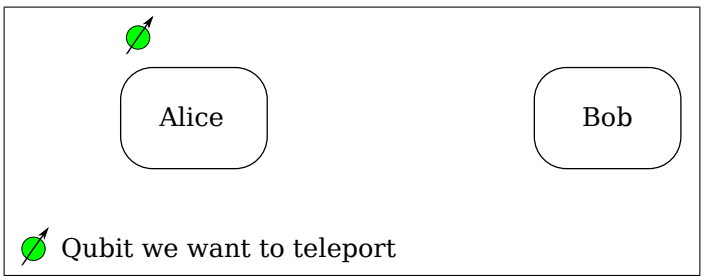


Figure 2.5: Teleportation Circuit - Qubit we wish to teleport

In Fig. 2.6 we have added one Bell pair (colored in red), which represent high-fidelity entangled qubits in Bob and Alice. This pair is the necessary resource to teleport quantum information.

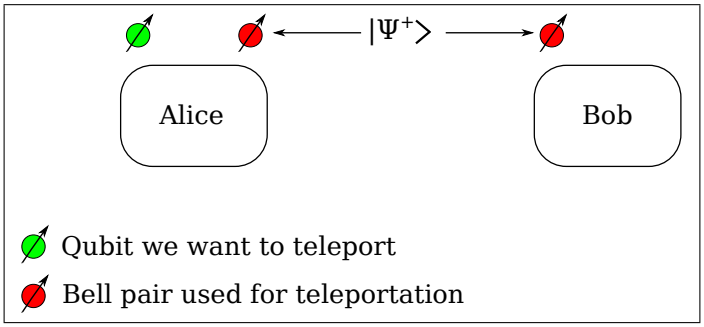


Figure 2.6: Teleportation Circuit - Bell pair available

As described before, we need to make some quantum operations in Alice, which are represented by Fig. 2.7.

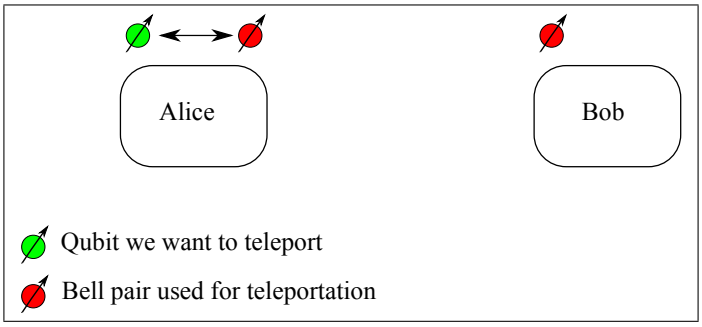


Figure 2.7: Teleportation Circuit - Interaction of Bell pair and data qubit

After this operation, the data qubit and Alice's entangled qubit are measured, and as a result, classical information is obtained (here represented by A & B bits). Once this measurement is done, the quantum states stored in Alice's qubits are destroyed. This is represented in Fig. 2.8.

The next step is to inform Bob about the results of these measurements, so we need to send a classical message with bits A & B. This is shown in Fig. 2.9

Next, Bob makes some operations on his qubit based on these classical bits Fig. 2.10 represents this.

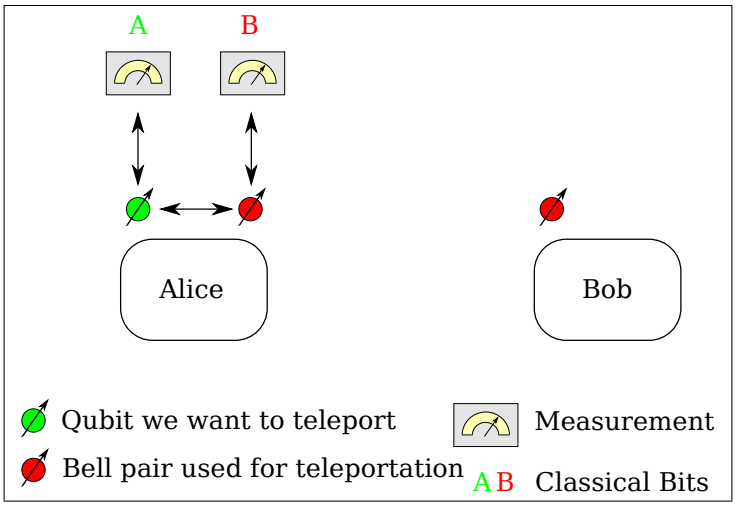


Figure 2.8: Teleportation Circuit - Measurement of qubits

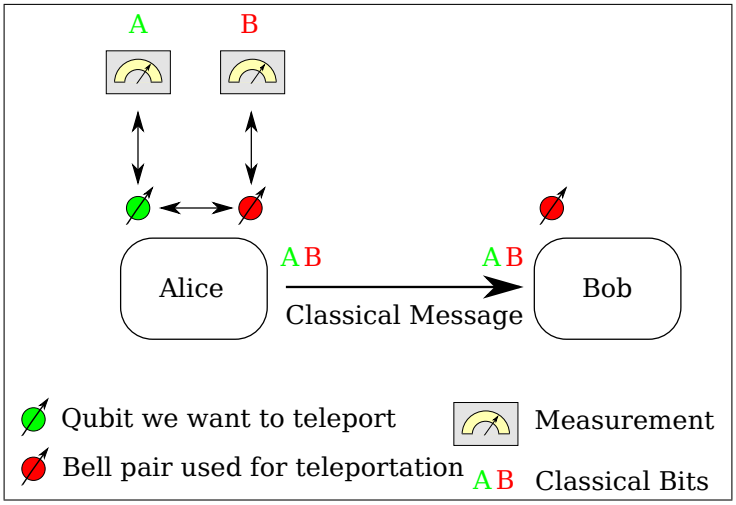


Figure 2.9: Teleportation Circuit - Send measurement results to Bob

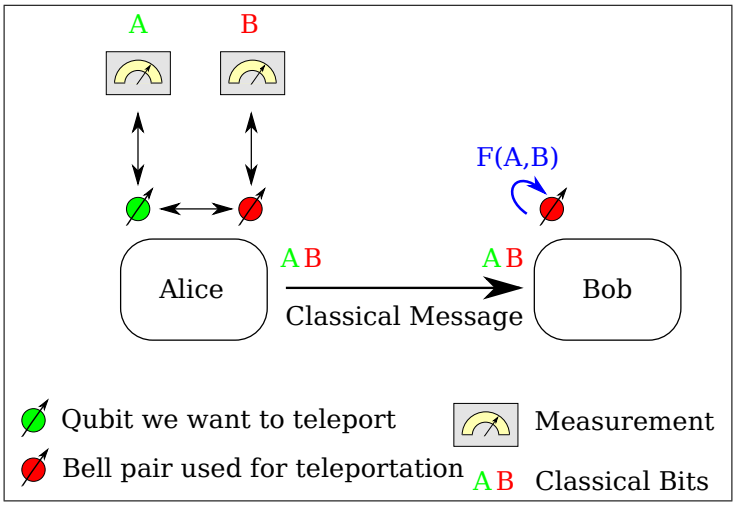


Figure 2.10: Teleportation Circuit - Apply operations to Bob's qubit

As a result, Bob qubit's quantum state changes into the state of Alice's original data qubit. This is shown in Fig. 2.11.

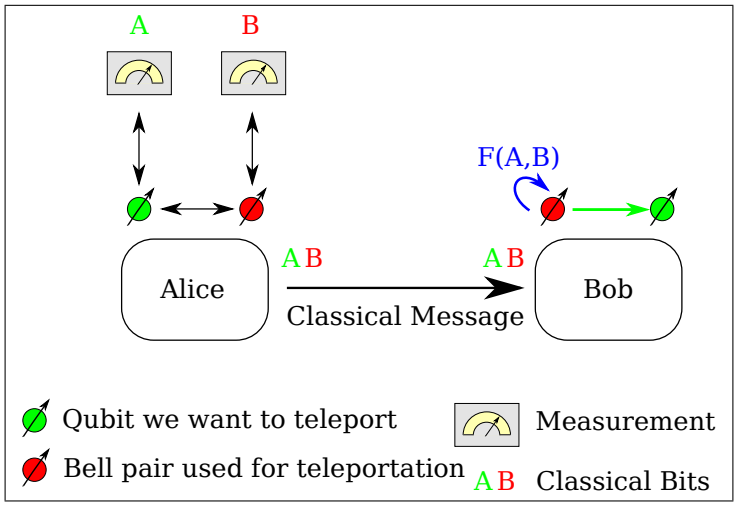


Figure 2.11: Teleportation Circuit - Bob's qubit turns into the data qubit

Chapter 3

Background on Quantum Repeaters

3.1 Introduction

Applications that use distributed quantum properties, such as QKD (Quantum Key Distribution) [20, 17], quantum Byzantine agreement [2] and other forms of distributed computation [13, 5], have the limitation that the *fidelity* of quantum states and the probability of success decrease with distance, making the use of these systems over long distances almost impossible. Therefore, researchers have proposed the design of quantum repeater networks [15] which would maintain *distributed quantum states* across greater distances.

Communication of quantum states depends on several quantum operations and properties: key among these is *entanglement*, in which the states of two or more quantum bits (qubits) are not independent. This operation is done by interacting qubits, producing this high correlation among their quantum states. For communications, one useful, basic form of entanglement is a *Bell pair*. Bell pairs can be created over a distance using optical pulses that are coupled to a qubit (represented as e.g. the spin of a single electron held in a quantum dot) at each end of a waveguide. Due to losses in the waveguide, this operation is probabilistic. Bell pairs can be used for teleporting a qubit from one location to another. The Bell pair is consumed in the process, so we must continually refresh the supply of available pairs. To cover distances of more than one hop, a form of teleportation called *entanglement swapping* is used to splice two short Bell pairs into one long one.

Previous work on quantum repeaters [11, 28, 7, 8, 14] has proposed different ways to produce entanglement via single photons or via very weak laser pulses. These produce high-fidelity Bell pairs, which makes purification almost unnecessary, but with a low probability of success. Other approaches improve the probability of success at the cost of reducing the initial fidelity [36].

When built, quantum repeaters will allow the distribution of entangled quantum states across large distances, playing a vital part in many proposed quantum systems. Enabling multiple users to connect through the same network will be key to their real-world deployment. Previous work on repeater technologies has focused only on simple entanglement production, without considering the issues of resource scarcity and competition that necessarily arise in a network setting.

Quantum repeaters are designed to produce high-quality entanglement between intermediate points tens or hundreds of kilometers apart, then extend that entanglement to end points spanning much longer distances. These entangled states can then become the fundamental resources for quantum protocols such as teleportation [3], QKD (Quantum Key Distribution) [20, 17], distributed quantum computing [13, 5, 2], and possibly improved optical interferometers for telescopes [21]. The main service that a quantum repeater provides is creation of high-fidelity physical entanglement (typically, though not necessarily, Bell pairs) between distant qubits. Depending on the mechanism used to generate Bell pairs, this may require entanglement purification to improve the fidelity, or some other kind of error correction. Once these high-fidelity Bell pairs are obtained, quantum states are forwarded using teleportation, producing long-distance Bell pairs which are finally used by applications such as the above. Several different designs have been proposed for the physical entanglement mechanism, such as the qubus mechanism [36] and single photon [7]. Recent experiments have demonstrated many of the building blocks necessary for repeater networks, including distribution of entanglement between separate quantum memories [9], transfer of photonic qubit states to matter qubits [29], teleportation between matter qubits [32] and purification of quantum states [41, 33]. Thus, the key hardware elements for large-scale quantum repeater networks are falling into place [26].

3.2 Physical entanglement

3.3 Purification

The purification algorithm allows the distillation of high-fidelity entangled qubits from an ensemble. Each purification step requires two Bell pairs (in this work both pairs are chosen of the same fidelity). Secondly, each station measures one Bell pair and sends the results of these measurements to the other one. If both stations measured the same, then the remaining Bell pair's fidelity would be boosted. The measured Bell pair is lost, clearly as a result of the measurement. This operation is non-deterministic, and the probability of success is higher for qubits with higher values of fidelity with respect to a Bell pair with lower fidelity.

Suppose we have two pairs of entangled qubits between station A and station B, defined by the density matrices ρ_1 and ρ_2, as follows:
First Pair:

$$\rho_1 = \begin{pmatrix} A_1 & 0 & 0 & 0 \\ 0 & B_1 & 0 & 0 \\ 0 & 0 & C_1 & 0 \\ 0 & 0 & 0 & D_1 \end{pmatrix}$$

The off-diagonal elements describe a superposition of states, which decay quickly to zero by using this purification procedure. Therefore, we ignore them in our calculations.

Second pair:

$$\rho_2 = \begin{pmatrix} A_2 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 \\ 0 & 0 & C_2 & 0 \\ 0 & 0 & 0 & D_2 \end{pmatrix}$$

As we are using the following basis:

$$\{|\Phi^+\rangle, |\Psi^+\rangle, |\Psi^-\rangle, |\Phi^-\rangle\}$$

each element in the diagonal of the matrices represent the probability of measuring one of these states. Then, after purification we obtain:

$$\rho_{PUR} = \begin{pmatrix} A_{PUR} & 0 & 0 & 0 \\ 0 & B_{PUR} & 0 & 0 \\ 0 & 0 & C_{PUR} & 0 \\ 0 & 0 & 0 & D_{PUR} \end{pmatrix}$$

where:

$$A_{PUR} = \frac{A_1A_2 + B_1B_2}{prob}$$

$$B_{PUR} = \frac{C_1D_2 + D_1C_2}{prob}$$

$$C_{PUR} = \frac{C_1C_2 + D_1D_2}{prob}$$

$$D_{PUR} = \frac{A_1B_2 + B_1A_2}{prob}$$

prob is the probability of success of the purification.

$$prob = (A_1 + B_1)(A_2 + B_2) + (C_1 + D_1)(C_2 + D_2)$$

As we repeat the purification algorithm, the value of A increases (with and ideal target value of 1) and the values B, C and D tend to zero. A value of almost 1 for A means that we would be able to measure the right

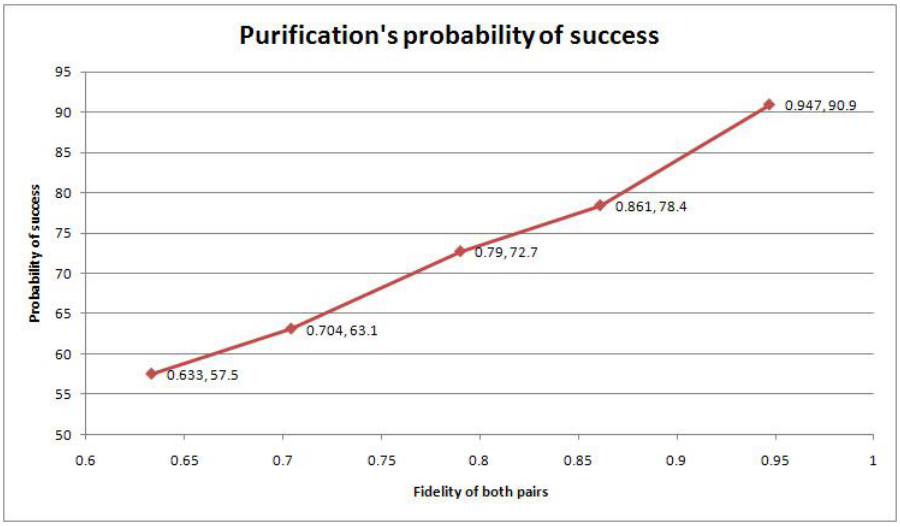


Figure 3.1: Purification.

Bell pair with high probability, while the others would unlikely be measured. For symmetric purification (both pairs of entangled qubits with the same fidelity) we obtain the graphic showed in Fig. 3.1. Each dot represents one purification step. The first number shows the initial fidelity of both Bell pairs, followed by the probability that the purification succeeds. Five purification steps are needed to reach a fidelity value of more than 0.98. As it can been seen, the probability of success increases with the initial fidelity of the Bell pairs.

3.4 Entanglement Swapping

3.4.1 Swapping

This function is based on the teleportation algorithm that allows us to teleport one qubit from one station to a further one. Thus, extending the entanglement range. This operation needs to be done many times until the final station is reached. In Fig. 3.2 the axis represent the fidelity of each Bell pair before swapping is done. The lines represents final fidelity levels after swapping. It can be seen that the final fidelity is always lower than the lowest fidelity of the Bell pairs to be swapped.

Suppose we have one Bell pair from station A to station B, and another Bell pair from station B to station C, defined by the density matrices ρ_1 and ρ_2, as follows:
First Pair:

$$\rho_1 = \begin{pmatrix} A_1 & 0 & 0 & 0 \\ 0 & B_1 & 0 & 0 \\ 0 & 0 & C_1 & 0 \\ 0 & 0 & 0 & D_1 \end{pmatrix}$$

Second pair:

$$\rho_2 = \begin{pmatrix} A_2 & 0 & 0 & 0 \\ 0 & B_2 & 0 & 0 \\ 0 & 0 & C_2 & 0 \\ 0 & 0 & 0 & D_2 \end{pmatrix}$$

By teleporting the qubit (entangled to station A) in station B to station C, we extend the range of the entanglement, obtaining the following density matrix:

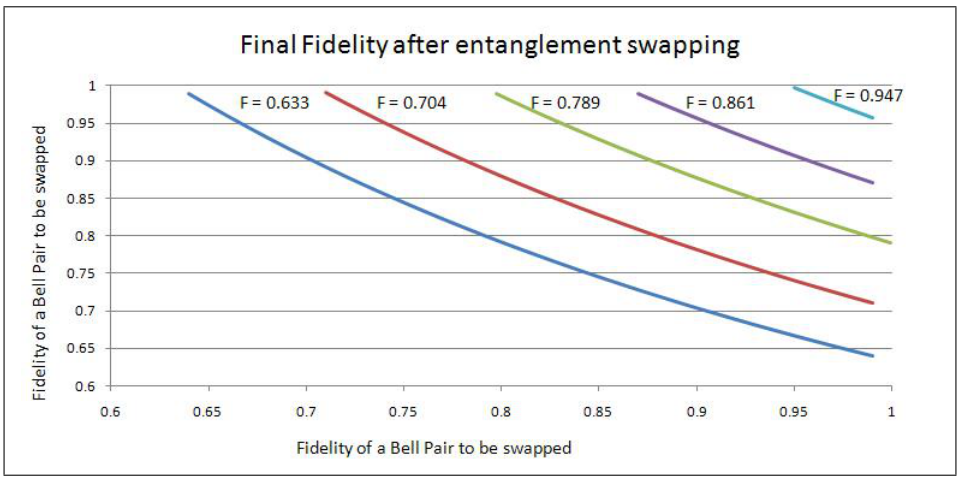


Figure 3.2: Swapping.

$$\rho_{SWP} = \begin{pmatrix} A_{SWP} & 0 & 0 & 0 \\ 0 & B_{SWP} & 0 & 0 \\ 0 & 0 & C_{SWP} & 0 \\ 0 & 0 & 0 & D_{SWP} \end{pmatrix}$$

where:

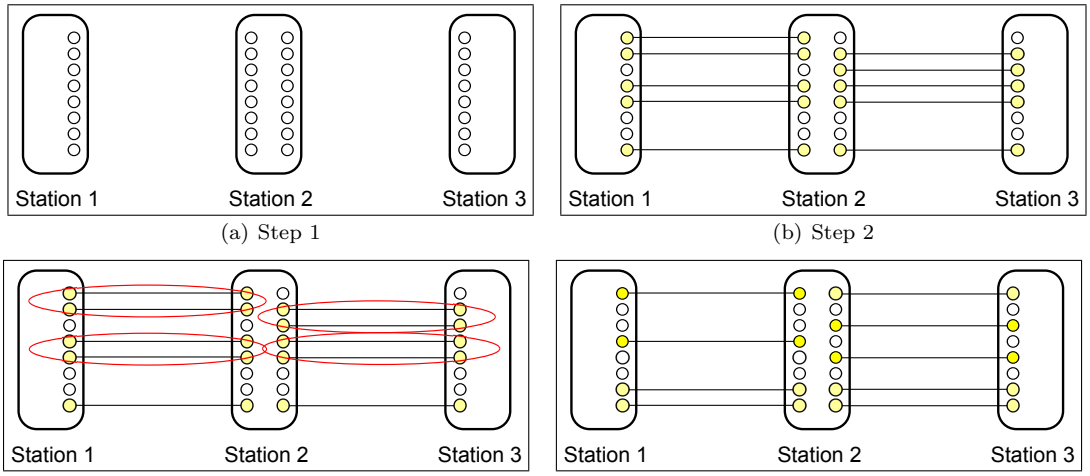
$$A_{SWP} = \frac{A_{12}}{Sum}$$
$$B_{SWP} = \frac{B_{12}}{Sum}$$
$$C_{SWP} = \frac{C_{12}}{Sum}$$
$$D_{SWP} = \frac{D_{12}}{Sum}$$
$$A_{12} = A_1A_2 + B_1B_2 + C_1C_2 + D_1D_2$$
$$B_{12} = A_1B_2 + B_1A_2 + C_1D_2 + D_1C_2$$
$$C_{12} = A_1C_2 + B_1D_2 + C_1A_2 + D_1B_2$$
$$D_{12} = A_1D_2 + B_1C_2 + C_1B_2 + D_1A_2$$
$$Sum = A_{12} + B_{12} + C_{12} + D_{12}$$

3.4.2 Graphical understanding of Purification and Swapping

In this section we explain in a series of pictures how the purification and swapping algorithms work in a three stations network. For this example, we assigned 8 qubits for station 1 and station 3 (which are the transmitter and the receiver) and 16 qubits for the repeater, where 8 qubits are assigned for reception and the others 8 for transmission. Initially, all the qubits are not entangled to others (what is represented by qubits in white color). This is shown in Fig. 3.3(a).

After initialization of qubits, entanglement is attempted between neighbor stations. As entanglement is a none deterministic operation, some of these attempts will fail and others will succeed (for the last ones, we represent them with a black line). In order to represent the fidelity of the obtained Bell pairs, we chose to represent them in yellow color, changing the intensity of it for higher-fidelity Bell pairs. The current state is represented in Fig. 3.3(b).

After obtaining some Bell pairs, the purification algorithm takes control of them and chooses some pairs of the same fidelity for purification. This selection is shown in Fig. 3.3(c).



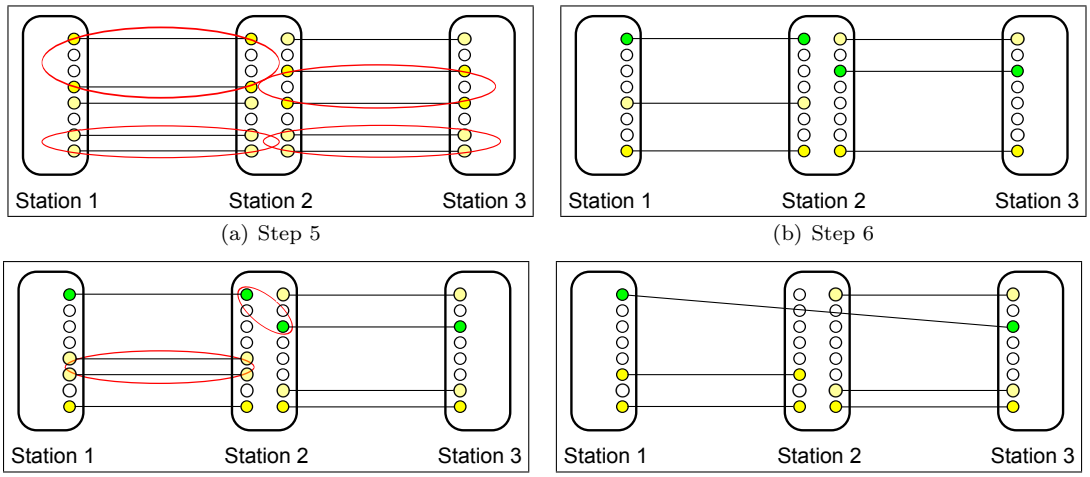
(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

Figure 3.3: Purification & Swapping - Sequence 1

Purification is also deterministic, so some of these operations may fail and both selected qubits need to be reset. In this representation, we supposed that all the operations succeeded, therefore only one qubit of each selected pair would be sacrificed. In order to represent the higher-fidelity Bell pairs we use a higher intensity yellow. In this representation we also added some other new Bell pairs to it. Fig. 3.3(d) shows this condition.

At this time, there are many Bell pairs with different fidelity levels. The purification algorithm must separate these Bell pairs and choose some in the same fidelity range. This is shown in Fig. 3.4(a).



(a) Step 5 (b) Step 6

(c) Step 7 (d) Step 8

Figure 3.4: Purification & Swapping - Sequence 2

In the next figure we show some Bell pairs that reach the level of fidelity that we decide it is enough for teleportation and swapping. These pairs are represented in green color. This can be seen in Fig. 3.4(b).

Once these Bell pairs are obtained, station 2 decides to swap as it has one high-fidelity Bell pair to each remote station. Chosen pairs are shown in Fig. 3.4(c).

Every time swapping is done, the fidelity of the new extended Bell-pair can be easily calculated as the product of the fidelity of each of the initial Bell-pairs. As the fidelity is always less than 1, the final fidelity will be less than the lowest value of fidelity of the initial Bell pairs. Therefore some further purification steps may be needed if the final fidelity decreases below some fixed threshold level. We don't show that situation here. Fig. 3.4(d) shows the state after swapping.

3.5 Quantum Memory

Quantum states in repeaters must be stored in stationary qubits usually referred as a quantum memory. Different types of quantum memory are being studied, each technology with different properties, but one common problem, fidelity decreases due to decoherence very fast. At the time of writing this thesis, the quantum memory time is not long enough to allow the usage of these qubits in quantum repeater networks, as for a 20Km link, the propagation time of the light in fiber optics is around 100 μseconds, by which timem the decoherence has destroyed the quantum state.

3.6 Basic Repeater Operation

Quantum repeaters are being built in order to create high-fidelity, long-distance entanglement. The work of a quantum repeater consists of three main functions: creation of some basic form of entanglement between repeaters directly connected via some physical channel (such repeaters can be referred to as "neighbors"); extension of that basic entanglement to create long-distance entanglement; and management of the infidelity introduced into the entanglement by channel imperfections, local gate errors, and finite memory lifetime. This work can be achieved using Bell pairs as an intermediate resource as well as the final goal.

The first role of a repeater is the creation of entanglement, almost exclusively in the form of maximally-entangled two-qubit states. The qubus mechanism, for example, uses a weak nonlinearity to create a small shift in a coherent state of light, detected using homodyne measurement against a reference state [36, 27]. This can be done, for example, using quantum dots held in cavity QED devices, with the spin of an electron in the dot as the qubit. The original qubus approach is proposed to be tuned to provide low-fidelity Bell pairs with high probability. A variant of the scheme using low photon numbers provides higher fidelity output with similar probability [31]. Nitrogen vacancy centers in diamond, again using electron spin as the qubit, can be used to create high-fidelity entanglement by forcing the emission of single photons and using interferometric methods to eliminate the which-path information [7]. Experiments have also been conducted using parametric down conversion to create entangled pairs of photons, but this approach is less useful for repeaters intended to couple immobile quantum memories [41].

The entanglement created by these physical mechanisms is inevitably imperfect, leading us to look for methods of raising the fidelity. Purification uses two or more Bell pairs to create one higher-fidelity Bell pair, sacrificing one or more of the Bell pairs to measure their parities and improve our confidence that the desired quantum state is selected [15, 16, 3]. Standard purification protocols require bidirectional communication of the classical measurement results to determine if the purification has been successful. Careful management of the order of purification operations has a large impact on performance, especially when the initial fidelity is low [37].

The purification scheme we used in this thesis is based on the scheme first given by Deutsch et al. [12, 15, 27]. Infidelity from the desired Bell state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is given by an admixture of other Bell states,

$$\rho = A|\Phi^+\rangle\langle\Phi^+| + B|\Psi^-\rangle\langle\Psi^-| + C|\Psi^+\rangle\langle\Psi^+| + D|\Phi^-\rangle\langle\Phi^-|$$

.

Pairs of such imperfect Bell pairs are purified together. Local rotations tailored to the specific state are performed before two controlled-NOT operations between the two pairs. The two target qubits are measured, and purification succeeds with probability $(A + D)$ when the two measurements coincide. After the operation the fidelity of the remaining pair is

$$\frac{A^2 + D^2}{N} > A$$

where N is the normalization factor for the remaining state after measurement.

Channel attenuation results in exponential decline in the probability of entanglement success, requiring multiple hops and some method of creating long-distance entanglement using short-distance entanglement. Entanglement swapping uses a form of teleportation to splice two shorter Bell pairs into one longer one. Dür and Briegel proposed using entanglement swapping in a nested fashion, with the length of the Bell pairs potentially doubling with each swapping step [15]. Swapping negatively affects the fidelity of the Bell pairs, requiring further purification over longer distances. If nodes A and B share a Bell pair with fidelity F and nodes B and

C share another Bell pair with fidelity F, performing swapping at B results in a single pair shared between A and C with fidelity F^2.

These three functions all use information transmitted through the network. The allowable sequence of messages, their contents and interpretation, the assumptions that each end is allowed to make about the other, and the definition of the actions to be performed by the node receiving the messages constitute a *network protocol*. The protocols can be organized into a *protocol stack* that collectively contributes to the end-to-end operation of the repeaters, this will be explained in detail in Ch. 5.

3.7 Multi-User Quantum Repeater Networks

A multi-user network with a complex topology requires repeaters to have additional intelligence beyond the three key protocol functions described above. One such function is selection of an appropriate *path* through the network to connect the end nodes. A second is some mechanism for *sharing* portions of the network in an effective manner. In this paper, we extend repeater studies to include complex topologies, and investigate their behavior when more than two of the nodes in such a network want to communicate at the same time.

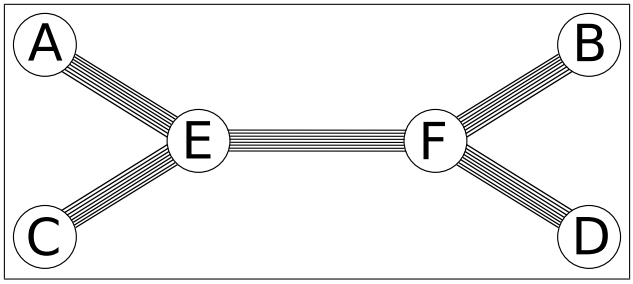


Figure 3.5: The dumbbell network, the simplest network with a shared link (EF) when A wishes to communicate with B and C wishes to communicate with D.

In Fig. 3.5, repeater A may wish to create Bell pairs with repeater B, at the same time that repeater C wishes to create Bell pairs with repeater D. Both communications (which we will refer to as communication *flows*) must use the link connecting repeaters E and F, resulting in *contention* for access to the link. Generically, we refer to quantum memories and channels as *resources* to be managed. This resource management problem is the focus of this paper.

In our example, how do we decide which flow, AB or CD, is allowed to use the EF link? How do we decide which flow is allowed to use the qubit memories at the E and F repeaters? The action for the flow that is given access is clear, but what should the flow that is *not* given access to the EF link do while waiting? In Sec. 5.8, we address the issue of multiple flows competing for resources, and we propose different multiplexing schemes based on what has been learned from classical networks.

3.8 Applications

In Sec. 1.1 we described some applications for quantum repeaters. For each of them, different functions will be done at the application level. In this thesis we focused only on the teleportation of quantum states, and using quantum repeaters as a network to transport them.

Chapter 4

Quantum Networks

In the same way as classical networks were designed to transport data between remote nodes, quantum networks have been proposed to do communication of quantum data. Though classical networks can use repeaters that amplify signals and copy data, quantum networks cannot rely on such operations as they are forbidden by quantum mechanics [40].

Quantum networks are based on the distribution of Bell pairs, which are high-fidelity entangled quantum bits (qubits). Each repeater needs to execute many quantum operations, which are done by small quantum computers. Therefore, a quantum network is considered to be a distributed quantum computing problem. In order to provide them for long distances, quantum repeaters were proposed, which allow to extend the distance of an entangled pair. The connection of many quantum repeaters in complex topologies makes a network of quantum repeaters.

4.1 What do networks do?

Dedicated end-to-end communication links are expensive, as the resources are assigned to one circuit whether they are used or not; for a fully-connected graph of N nodes, $N(N-1)/2$ links are required, quickly becoming prohibitively expensive as well as physically impractical as N grows. On the other hand, multi-user networks in which communicating parties may be connected over a series of *hops* lower the cost for communications, as the resources are shared among nodes and provide the flexibility to connect to more than one node using the same infrastructure. However, this networked approach results in a series of problems, mostly originating from issues in scaling the network to large numbers of nodes, the lack of centralized control, and errors that occur. In this chapter, we introduce some of these topics in the context of quantum repeater networks.

This thesis addresses of distributed control of the communications the network is designed to support, and management of the dynamic errors that inevitably occur in quantum systems.

4.2 Changing network topology

Key problems in computer systems include naming and resource management. These get harder as the systems grow and become distributed, because information gets out of date, autonomous systems don't want to share the information at all, and errors occur constantly. Networks should be carefully designed to support the addition of new nodes or new networks. The Internet is a perfect example of such a network which is scalable and can easily update changes in the topology. Quantum networks should also be developed with this consideration if we want them to scale as well as the Internet.

Like classical networks, links going down and up, make changes to the network topology. For quantum networks, we have two type of links, the quantum channel which is the connection to produce entanglement between neighboring nodes, and the classical network, which basically is a TCP/IP network that allows control messages to be exchanged between any node in the network. Failure of a quantum channel or of the classical network should be identified and reported to the routing protocols to try to find an additional path (if any) [34].

4.3 Dynamically changing network state

Even without taking into consideration changes to the underlying physical topology, attempts to use the network must deal continuously with the changing availability of resources as operations are tried, suceeded or failed. When high-fidelity Bell pairs are produced between neighboring nodes, a new resource becomes available for use by the communication flows. This resource sits around, consuming memory in the two nodes, until it is used. This happens many times per second, however, it is not the only dynamic change. Every time a Bell pair is swapped, the state of the network changes again, with the extension of the entanglement range. With this new resource available, different pairs of nodes are connected via entangled pairs, and new decisions about what actions are possible and preferred must be made. Also when fidelity decreases due to decoherence, some high-fidelity Bell pairs may need to be purified again, eliminating the resource temporarily from the topology. All these changes happen in different parts of the network constantly, making these updates available for the rest of the network a very difficult task. What is worse is that even if we decided to send these updates to the nodes, the information probably would be outdated by the time it arrived.

Every time purification is attempted, the results of the measurements that inform whether purification succeeds or fails must be reported to the remote node. This can take some time if the nodes are far in a network. But time is a constraint in quantum networks, as decoherence affects the fidelity of the Bell pairs, so quick actions must be taken and same decisions should be made by nodes with the less possible amount of messages exchanged.

End-to-end entanglement is created in this work by swapping Bell pairs. Unlike classical networks, where exchanged routing tables are used to make routing decisions, quantum networks cannot rely in them. Path selection in quantum networks would change constantly due to the reasons explained above and therefore it would become very difficult for the nodes to make decisions based on possibly-outdated available entangled resources tables. It is the responsibility of the swapping layer to choose the right Bell pairs to swap. In this thesis, for the small size simulated networks, we used an static approach, however, for future work, this layer needs to be studied in depth to find patterns and good algorithm to make routing decisions. After swapping is produced, information of the new destinations nodes and the final fidelity after swapping must be reported, and nodes should be able to adjust the density matrix considering the effects of decoherence as the time passed since the swapping operation was produced. Some additional considerations should be taken for swapping. Finding a middle point in the network and trying to produce Bell pairs to it, to finally swap them is an easy way to make swapping decisions, however, we need to find where the middle of the network is, and if we have dynamic topologies this may result in a hard task. The order in which the swapping operations are done may lead to deadlock situations where there are no possible swapping operations that may allow us to produce an end-to-end Bell pair, making all the resources be lost due to decoherence and gone unused.

Classical networks have costs for their links based on the number of hops, bandwidth or some fixed cost, so that routing protocols can make the best decision based on the network topology. Quantum networks also have some sort of cost which is based on the fidelity of the Bell pairs available for that link. Morover, as Bell pairs are swapped to reach a desired node, one node may have many Bell pairs, and some of them may be entangled to a node that is closest to the final destination and which be the best one to choose for swapping at that time. Bell pairs will show a dynamic behavior as fidelity decreases with time, purification is done to boost it, and swapping occurs in some nodes. Therefore, maintaining updated information of the Bell pairs for all the nodes is a very difficult task, making the calculation of a path cost not easy.

4.4 Fault tolerance

To provide fault-tolerant quantum networks, additional problems must be considered if we compare them to classical networks. Regarding quantum networks, we must distinguish two different communication links, the classical channel (TCP/IP networks to send classical control messages between nodes) and the quantum channel which is a point to point link and is the one that allows the creation of Bell pairs. Failure of any of them will result in a failure of the network. We rely on TCP for error correction and assurance of delivery of classical messages. However, if messages are retransmitted, further decisions and operations are also delayed, producing an impact in the fidelity of the resources, as they will remain idle waiting for the messages. Like classical networks, redundant paths may be consider to prevent the connection between nodes to fail if some of the links between them goes down. As we mentioned many times before, decoherence seriously affects the fidelity

of the Bell pairs, and how fast it decreases depends on the quantum memory used. In the case that nodes are using different quantum memories, the fidelity degradation ratio will be different for each node, and they cannot calculate the fidelity of the Bell pair without knowing the remote node's quantum memory. Therefore, information about when the Bell pair's fidelity will be decreased should be exchanged between nodes and only taken the worst of them in consideration. Finally, nodes may discard qubits if noise in the device increases or for whatever reason the node decides that a reinitialization is better. These actions should also be considered and reported to the remote nodes with whom the Bell pairs are shared.

4.5 Solving these problems

4.5.1 Protocols

To keep control of quantum operations and to report the results of the measurements, a communication protocol is necessary to properly deliver messages between nodes. These protocols must be designed layered to allow independence of the different layers and easy upgrade of each of them without impacting the rest of the protocols. In order to design robust protocols, we need to create finite state machines and decide a proper definition of the legal sequence of actions and timeouts that such protocols must follow.

4.5.2 Resource management

Moving to a network of quantum repeaters does, however, raise questions that are not posed by a simple linear topology. One of the most important is what to do when multiple users want to use the network at the same time, as the uses of qubit memories and the quantum channels become scarce *resources* for which *contention* must be managed. Such questions have close classical analogues in the way data is transmitted within networks and over the Internet, where several different classical schemes have been devised over the years to allow what is known as *multiplexing*: the use of a network by several different users, often competing for the same resources.

4.6 Metrics for Success in Networks

To study the performance of the simulated networks, we measured the *aggregate throughput* and calculated the *fairness*.

The aggregate throughput between two stations is defined as how many Bell pairs per second of a particular fidelity are teleported between them, what Dür and Briegel [15] refer to as the *yield*. In order to do that, high-fidelity end-to-end Bell pairs must be produced before the data qubit could be teleported.

The fairness we use in this thesis to study our networks is define using Jain's fairness measure for resource allocation [23],

$$\mathcal{J}(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

A fairness of $\mathcal{J} = 1.0$ indicates perfectly even distribution of resources among the users, while $\mathcal{J} = 1/n$ indicates that one user acquired all of the resources, shutting out all other users.

Chapter 5

Protocol Design

5.1 Quantum Network Protocol

In Ch. 4 we explained some of the problems that we must address for a proper operation of quantum networks. This chapter presents some solutions to solve them, we propose quantum protocols that will enable distributed and consistent decision making for the nodes. Some of the functions these protocols must handle are reporting results of quantum operations, results of any measurement, exchanging density matrices, the time when these operations were done, etc. In the next two chapters we evaluate these solutions via simulations.

Network architecture is more than simply the contents, formats and semantics of the messages themselves; it also includes many aspects of the behavior which may be visible only implicitly, rather than in the contents of the messages. One of the key areas, as discussed in Sec. 4.5.2, is management of sharing. Here we present several possible approaches, which will be evaluated in Ch. 7.

The process of designing quantum networks is similar to designing classical networks, as they require detailed protocol designs, including finite states machines to control physical resources and track logical state. A layered protocol stack has previously been proposed [37]; here we provide detailed designs for the individual layers. We give a brief description of each and the functions which are simulated. Fig. 5.1 shows the layers of this protocol.

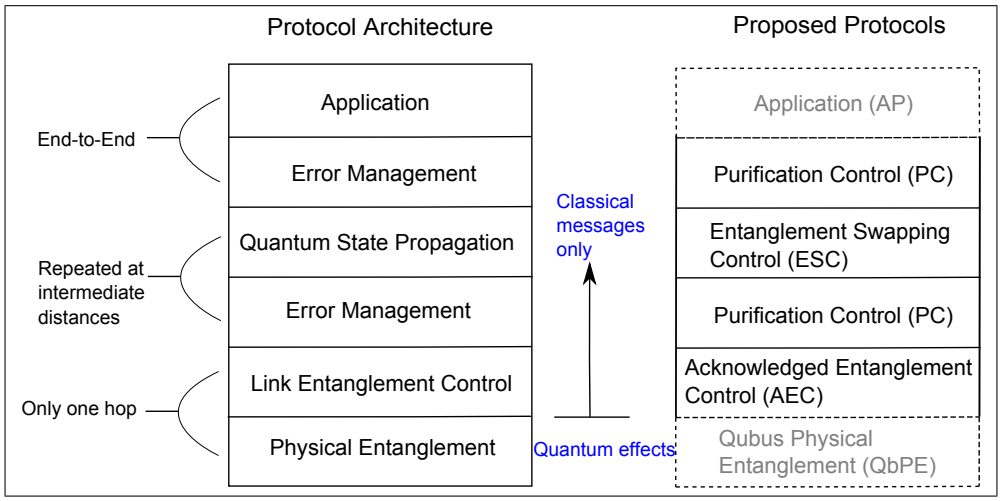


Figure 5.1: Protocol Stack Architecture (left) and Proposed Protocols (right)

All these layers interconnect in the way shown in Fig. 5.2. It can be seen that after the Entanglement Control (ESC) layer is done, the next higher protocol is again Purification Control (PC) but in this case the Bell pairs belong to further stations. And this keeps on repeating until the end-to-end Bell pair is purified. Finally, the application layer is reached and the data qubit can be teleported.

One of the key purposes of the classical protocols is to keep track of the fidelity of Bell pairs, which can only be estimated and not measured. Decisions are made based on these estimated values in order to allow Bell pairs

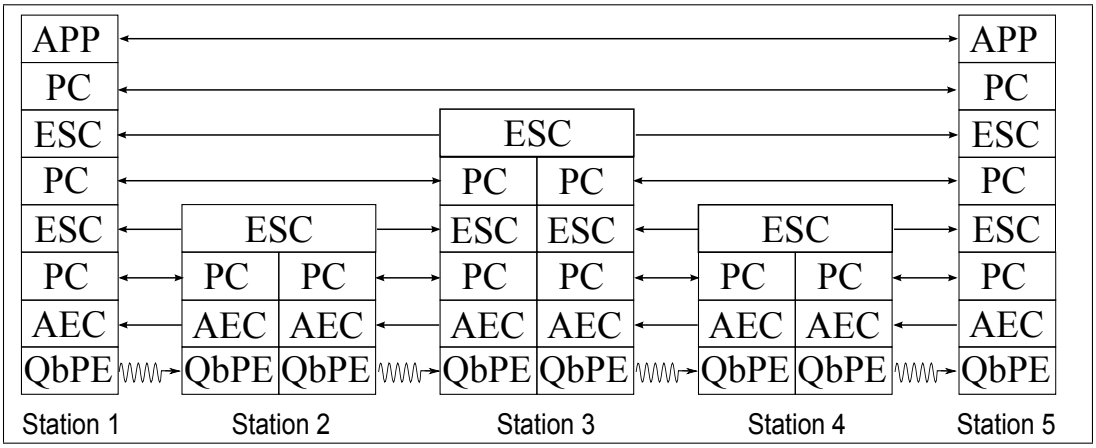


Figure 5.2: Protocol Stack with many stations. Each arrow represents a separate message or connection, rather than encapsulation of a single message.

to be swapped or sent to the Application layer for use. The control of a qubit (a single-qubit buffer) is passed from layer to layer until consumed by the Application layer or reinitialized to start over from the lowest layer.

5.1.1 Comparison with the OSI model

The Open Systems Interconnection model (OSI) was proposed by the International Organization for Standardization (ISO) to divide communication systems into independent layers which interact with others, and it is an excellent way to study and develop new systems. Considering these advantages, a protocol architecture was proposed by researchers [37] to be taken as a reference for future proposals for quantum networks. In this work we proposed a stack of protocols for a possible implementation for network of quantum repeaters.

In the OSI model, as packets go down through the layered architecture, they are added one header in each of the layers. When packets are received, these headers are removed as the packets go up through the layered architecture. Such concept is called encapsulation. Though in the protocol architecture used in quantum networks there is no concept as encapsulation, this model is still a good reference for our work. The reason why there is no encapsulation is that all these layers are for classical control messages, where each layer communicates with others on the same station and with the same one on a remote one. Clearly before sending classical information to other nodes we need to encapsulate them in some transport protocol, but there is no encapsulation inside our proposed architecture.

In order to deliver classical messages between nodes, we rely on TCP/IP to provide reliable and ordered delivery control packets. Therefore, quantum networks must work together with a classical network for control of the quantum operations and reports of results. Regarding IP addressing there are no special needs, the only requirement is to deliver classical messages via a reliable network. In addition to the classical addressing of nodes, it is necessary to address the resources (qubits) in the nodes. We proposed the following addressing architecture hierarchy:

- Node address: How to identify a node. This should be independent from the IP address.
- Interface address: Each node will have many interfaces where quantum channels are connected. This address identify them.
- Qubit address: Each node will hold many qubits in their quantum memories. This address specify each of them.

This allows us to address any single qubit placed in any node. However, Bell pairs refer to a pair of qubits, and the epoch is used to identify the Bell pair from future entanglements (epoch is increased every time a qubit is initialized).

While this approach worked well for this work, we may want to consider joining the interface address with the qubit address in the future. This will simplify the addressing and will hide the interface information from

the nodes. In addition, we considered using Bell pairs' labels to uniquely identify qubits, interfaces, and nodes, without actually knowing what the destination addresses are, just by referring to a label. This remains as a future work.

In the next section we explain how these messages are delivered.

5.2 Inter-node Message structure

As it will be explained in the following sections, there are two classes of messages that are sent to control the quantum operations. One class is sent between the same layers but in different nodes. The second class are the messages that are sent between the software entities that implement different layers but within the same node. The last class is explained in Sec. 6.5.1, as it is part of the simulator. We will use the big-endian order for the data fields in these messages.

In order to transport the protocols' messages of all the layers to other nodes we propose the header shown in Fig. 5.3 to be used for encapsulation, and this new packet is to be encapsulated in a reliable transport protocol such as TCP.

Bit offset	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31	
0	Source Address	
32	Destination Address	
64	Source Interface	Destination Interface
96	Protocol Type	Reserved
128 ...	Payload	

Figure 5.3: Encapsulation header

Where:

- Source Address: Address of the node sends the message. This is not an IP address, but the address to identify the node.
- Destination Address: Address of the node that receives the message. This is not an IP address, but the address to identify the node.
- Source Interface: Each node will have a gate (interface) where the quantum channel is connected to. This field represents this gate of the source node.
- Destination Interface: Each node will have a gate (interface) where the quantum channel is connected to. This field represents this gate of the destination node.
- Protocol Type: This field identifies what type of message is encapsulated.
- Payload: Here the encapsulated message is placed.

To address the protocol in use, we summarize the protocol type codes in Table 5.1.

5.3 Physical Layer

5.3.1 Introduction

The physical entanglement layer represents the physical interaction that creates Bell pairs between two different stations. There are many proposals for this layer and at the moment no clear winner. Our simulations model

Protocol Type	Layer	Description
100	QubusPE	Entanglement attempt
200	AEC	Entanglement report
300	PC	Purification measurement
400	ESC	Swapping report
500	APP	Teleportation measurement

Table 5.1: Proposed protocol type codes

the qubus mechanism [36] in which laser pulses of many photons generate low-fidelity Bell pairs with high probability, and the single photon mechanism [7] in which single photons generate high-fidelity Bell pairs with very low probability. This probability of success is due to the attenuation in the optical fiber, which has an exponential increase with distance. Thus, photons may be lost when traveling, or may not be detected at the receiver. For the qubus mechanism, for a distance of 20km over an optical fiber with a 0.17 dB/km loss, the system can be tuned to have a probability of success of the entanglement around 36%, with an initial fidelity of 0.633. On the other side, single photon mechanism produces Bell pairs with an initial fidelity around 0.97 but the probability of success is less than 1%.

Once the Bell pairs are produced, decoherence also decreases their fidelity as a function of time, as information leaks into the surrounding environment.

The physical capabilities of different physical layers vary. Some support only a single physical transceiver qubit, and so can support only a single outstanding entanglement attempt. Others support independent multiplexing of incoming light pulses to local qubits, which is done by a classical herald pulse (trigger) followed by one or more quantum pulses.

5.3.2 Messages

We define the following data structure shown in Fig. 5.4. It will be implemented as an array in the packet for the physical layer.

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Source Qubit																Epoch															

Figure 5.4: Data structure defined for the Physical Layer

Where:

- Source Qubit: Specifies which qubit was interacted by this quantum pulse.
- Epoch: Each qubit has an associated *epoch*, a counter of the number of times it has been initialized, to prevent old messages from being misinterpreted.

The messages exchanged between stations are proposed to be composed of the fields shown in Fig. 5.5.

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version								Type								Number of pulses															
32	Array of Physical data structure (32 bits)																															

Figure 5.5: Messages exchanged in Physical Layer

The fields in the messages are:

- Version: Version of the implementation of this protocol. To allow future updates and compatibility.

- Type: Type of message. This is to identify the different messages that each layer may use.
- Number of Pulses: The number of pulses sent in the entanglement attempt. Each pulse interacts with one qubit in the source node.
- Physical data structure: Array of elements defined in Fig. 5.4. The number of elements will be equal to the number of pulses.

Source Qubits and Epoch are repeated as many times as specified in the number of pulses field.

In Fig. 5.6 we can see the type of messages exchanged and the functions of each message for the physical layer.

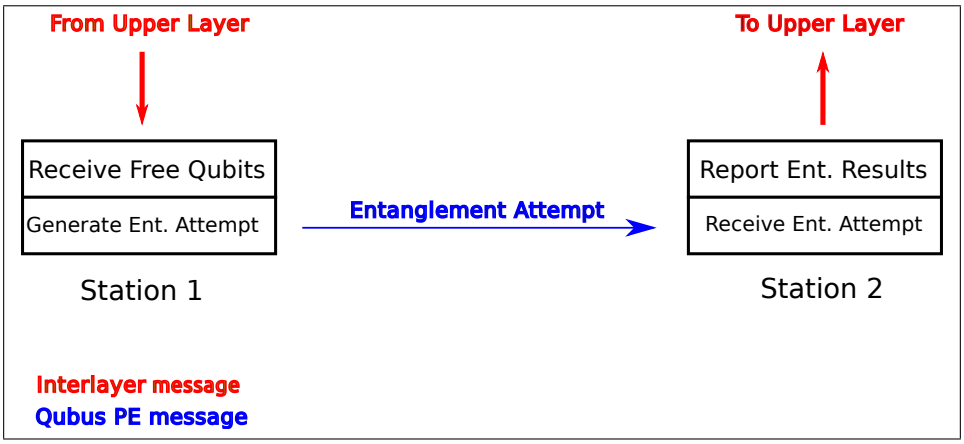


Figure 5.6: Message type exchanged in Physical Layer

5.4 Acknowledged Entanglement Control Layer

The second layer, AEC (ACKed Entanglement Control), is responsible for managing the single-hop physical entanglement process, selecting qubits to attempt entanglement at each end of the link, and utilizing classical messages to report the results. When a laser pulse is detected by the receiver, measurements are done, and a message will be sent back to the transmitter, informing it which qubits on the receiver were entangled to which qubits on the transmitter. The stationary qubits in a repeater are not destroyed when the qubit is measured or reinitialized, though the quantum information held in the qubit is. Epoch information is also exchanged and together with the qubit number are used to uniquely identify the Bell pair.

Once entanglement succeeds, this layer will transfer control of the Bell pair to a higher protocol layer.

5.4.1 Finite State Machine

Fig. 5.7 shows the finite state machine which describes the behavior of this layer for qubits in the transmitter, and Fig. 5.8 for qubits in the receiver.

Transitions represented in blue include an interaction with a remote station. OUT refers to messages sent, and IN to messages received. Black transitions represent local operations. The different states are:

- **Uninitialized**. This can be reached from a higher protocol layer or after starting up the repeater. Qubits are in an unknown state.
- **Unentangled**. During initialization of the qubits, they are prepared to have a known quantum state, ready to start entanglement, and the epoch is incremented. If the qubit's fidelity falls below a threshold, it becomes unusable and will be returned to the Uninitialized state.
- **Interim Entangled**. This state is reached after entanglement is attempted. This operation is done by sending a laser pulse to the remote station after interacting with the local qubits. The qubit will remain

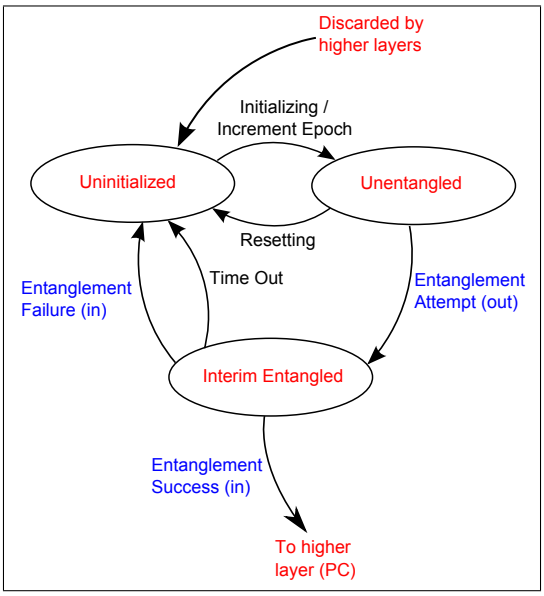


Figure 5.7: Finite state machine for Transmitter's ACKed Entanglement Control

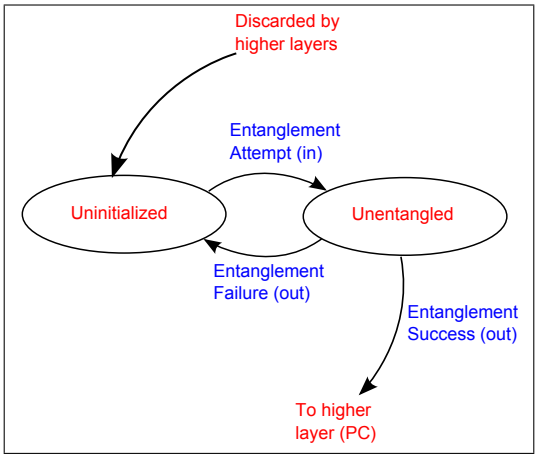


Figure 5.8: Finite state machine for Receiver's ACKed Entanglement Control

in this state until it receives an answer from the remote station or a local timer times out (to prevent decoherence from affecting the fidelity). If an Entanglement Failure message is received or no answer is received before the timer expires, the qubit will be moved to Uninitialized to start over again. If an Entanglement Success message is received, the qubit will be moved to the next higher protocol layer, in this case, Purification Control.

5.4.2 Messages

We define the following data structure shown in Fig. 5.9. It will be implemented as an array in the packet for the entanglement control layer.

Bit offset	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0	Source Qubit	Epoch
32	Zero Time	
64	Zero Time	

Figure 5.9: Data structure defined for the Entanglement Control Layer

Where:

- Source Qubit: Specifies which is the entangled qubit. If in zero, that means that entanglement failed for that pulse, otherwise the entangled qubit is specified.
- Epoch: This number represents how many times the qubits have been reset.
- Zero Time: Time of the interaction of the laser and the qubits. This is used to calculate how the fidelity decreases with time.

For this layer to communicate with other stations, we propose the packet format shown in Fig. 5.10.

Bit offset	0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0	Version	Type	Number of pulses
32	Array of Entanglement Control data structure (96 bits)		

Figure 5.10: Messages exchanged in AEC Layer

This packet looks similar to the one defined for the physical layer, however, there is a difference in the information held in both packets. Where:

- Version: Version of the implementation of this protocol. To allow future updates and compatibility.
- Type: Type of message. This is to identify the different messages that each layer may use.
- Number of Pulses: The number of pulses previously sent in the entanglement attempt.

Source Qubit and Epoch are repeated as many times as specified in the Number of Pulses field.

In Fig. 5.11 we can see the type of messages exchanged and the functions of each message for the AEC layer.

The physical layer and the entanglement control layer work together, as the first one attempts entanglement while the last one reports the results. The handshake of messages between neighbor stations can be shown by Fig. 5.12. Station 1 sends an entanglement attempt packet (together with the laser pulse) through the physical layer (red packet). When Station 2 receives both packet and pulse, it attempts to detect the pulses. Successful detections imply that the entanglement succeeded. These operations are represented by some delay which is much smaller than the propagation delay of the messages. After this operation is done, Station 2 reports the results to Station 1 by an AEC message (represented in red).

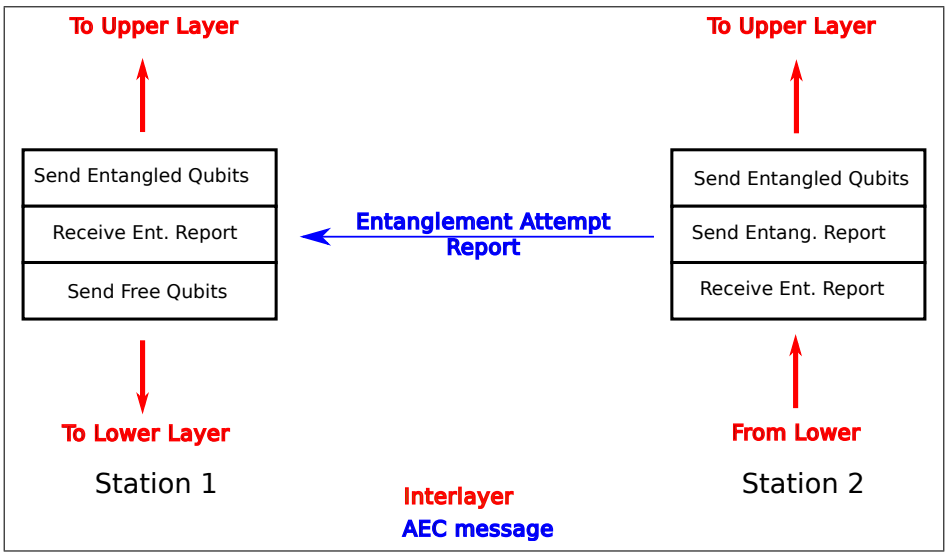


Figure 5.11: Message type exchanged in Acknowledged Entanglement Control Layer

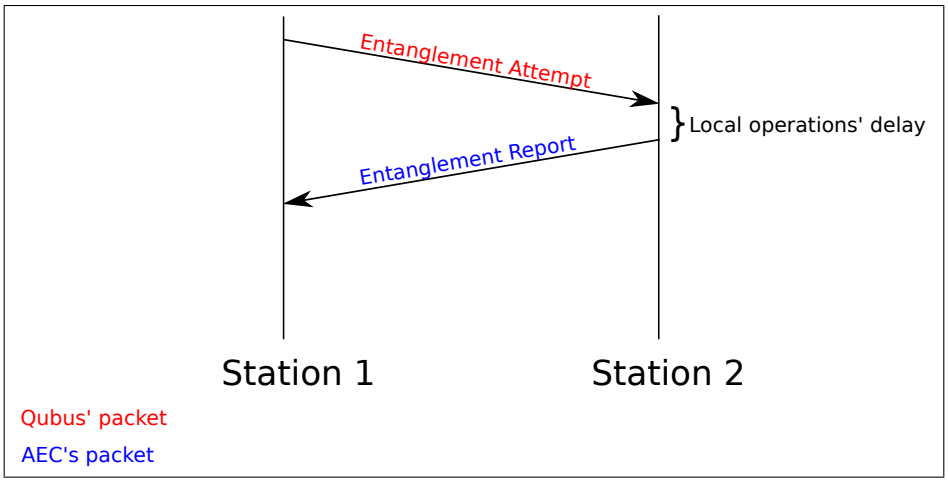


Figure 5.12: Handshake of messages between 2 neighbor stations during entanglement attempt

5.5 Purification Layer

In this work we use an error management method called purification, which has been explained in Sec. 3.3. In order to purify a Bell pair (boost its fidelity), an additional Bell pair is sacrificed in the process. The third layer of the protocol, PC (Purification Control), is responsible for choosing two Bell pairs, and electing one pair to have its fidelity boosted and the other to be sacrificed, assuring that both stations make the same decisions. Purification is done between two arbitrary stations, and no other stations need to be considered. Thus, PC does not need to make any complex routing decisions, but it does need to be able to address any station in the network. After the PC layer confirms a sufficient fidelity for the Bell pairs, control is given to the next higher protocol layer, which could be the Application layer, or the Entanglement Swapping Control (ESC), depending on whether or not this round of purification was done between end-to-end stations. If the physical layer produces high-fidelity Bell pairs, there is no need to execute any purification, so this layer could be configured as a null layer. Qubits to be purified are assigned a band depending on their fidelity values [37] and the qubits within the same band are selected for purification.

5.5.1 Finite State Machine

Fig. 5.13 shows the finite state machine which describes the behavior of this layer. The states are:

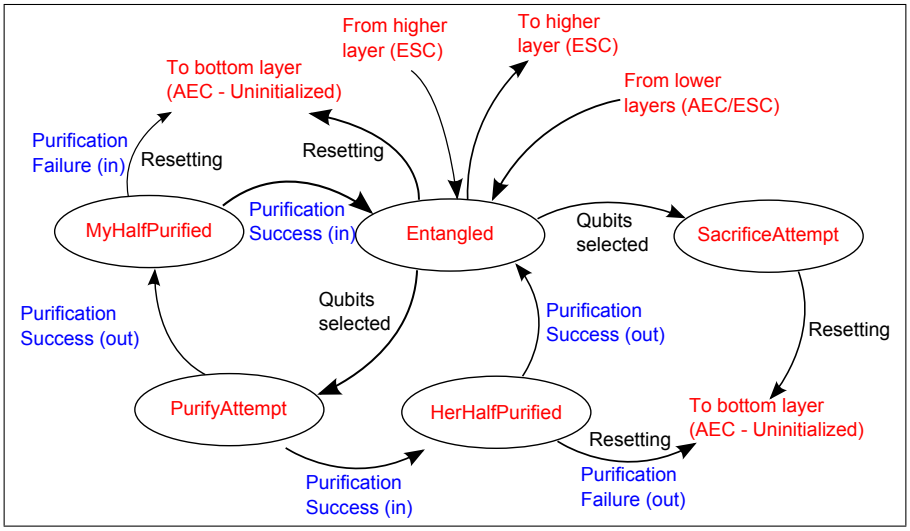


Figure 5.13: Finite state machine for Purification Control

- **Entangled**. This state indicates that the qubit is entangled to another qubit in a distant station, but with not enough fidelity to start teleportation. It can be entered from a lower layer like AEC (just after entanglement is produced) or ESC (as fidelity is always reduced by swapping), from a higher layer ESC (if decoherence affects the fidelity and the Bell pair needs to be purified again), or finally from the purification algorithm itself after successful purification. Once a high level of fidelity is reached, control of the qubit is transferred into the next higher layer (ESC).

  If the qubit remains in this state for a long time, the fidelity will drop due to decoherence. The qubit is moved to Uninitialized in order to attempt entanglement again.

- **PurifyAttempt**. Once we have two Bell pairs with similar fidelity, one Bell pair is assigned to this state. If purification succeeds, this pair will have its fidelity boosted. After attempting purification, the qubit is moved to the state MyHalfPurify or HerHalfPurify, depending on whether this station starts the purification before receiving any purification message from the remote station.

- **SacrificeAttempt**. The second Bell pair chosen for purification is assigned to be sacrificed to improve the fidelity of the first Bell pair. Regardless of the result of the purification, this qubit will always be moved to Uninitialized state after the purification process finishes.

- **MyHalfPurify**. If the station starts the purification process, after sending a message to the remote station, the qubit is moved to this state, until it receives an answer. On success it will be moved to Entangled, or to Uninitialized on failure.

- **HerHalfPurify**. This state indicates that the station received a message notifying it that the remote station has started the purification process. If the operation on the local station succeeds, the qubit will be moved to Entangled, after sending a message to the remote station. If it fails, it will be moved to Uninitialized, after sending a Purify Failure message to the remote station.

In Fig. 5.14 we can see the type of messages exchanged and the functions of each message for the PC layer.

5.5.2 Messages

The messages exchanged between stations are proposed to be composed of the fields shown in Fig. 5.15.

Where:

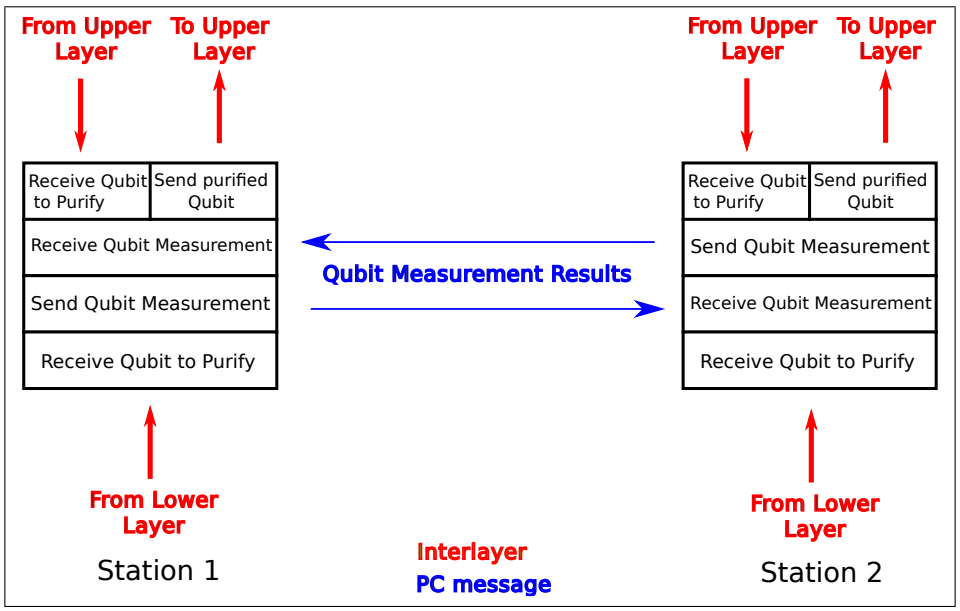


Figure 5.14: Message type exchanged in Purification Control Layer

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version								Type								Source Qubit to purify															
32	Destination Qubit to purify																Source Qubit to sacrifice															
64	Destination Qubit to sacrifice																M	E	Padding						Epoch							
96	Epoch								Zero Time																							
128	Zero Time																															
160	Zero Time								Reserved																							

Figure 5.15: Messages exchanged in Purification Layer

- Version: Version of the implementation of this protocol. To allow future updates and compatibility.
- Type: Type of message. This is to identify the different messages that each layer may use.
- Source Qubit to Purify: Qubit number which was selected for purification in the source node.
- Destination Qubit to Purify: Qubit number which was selected for purification in the destination node.
- Source Qubit to Sacrifice: Qubit number which was selected for sacrifice in the source node.
- Destination Qubit to Sacrifice: Qubit number which was selected for sacrifice in the destination node.
- M bit: This bit specifies the result of the measurement done for purification.
- E bit: This bit is in 1 if the local gate operations failed, and therefore, purification failed too. 0 for none errors.
- Epoch: This number represents how many times the qubits have been reset.
- Zero Time: Time when the measurements for purification are done.

In Fig. 5.16 we see the handshake of messages between stations during the entanglement purification process.

Purification algorithm performs measurements on both stations and report the results to the remote site. Once these measurements are received, the stations decide that the purification succeeds if both stations measure

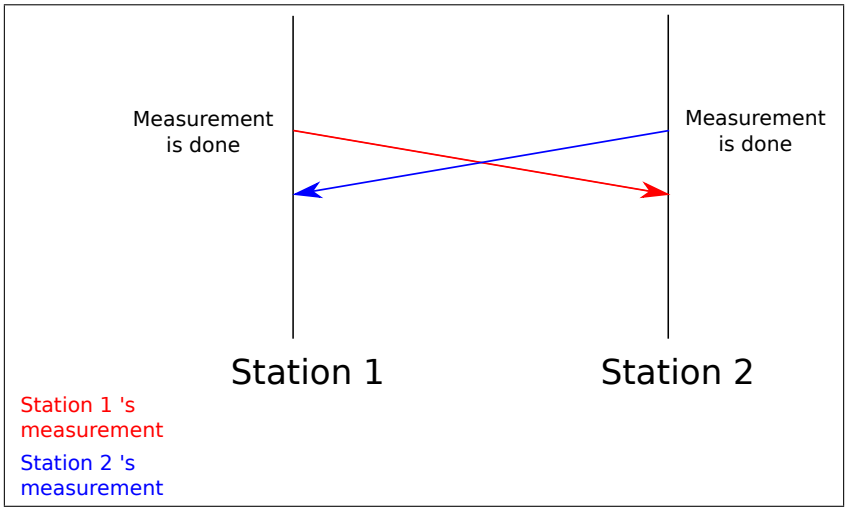


Figure 5.16: Handshake of messages between 2 neighbor stations during purification

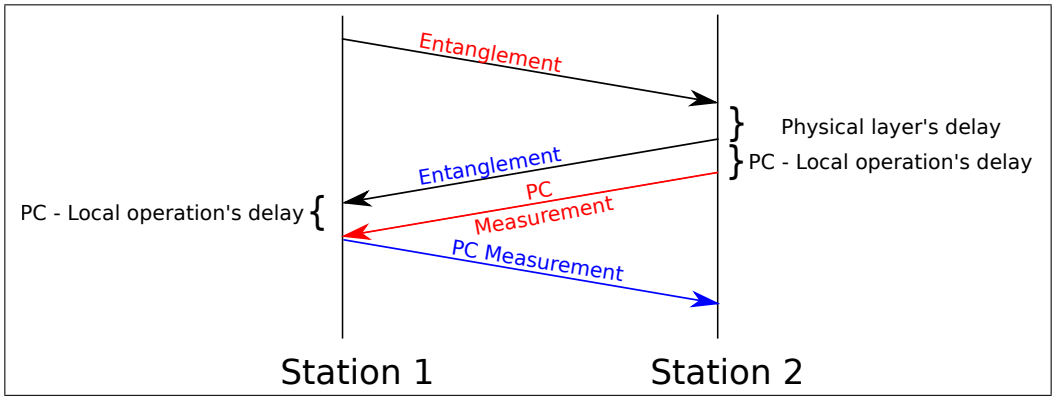


Figure 5.17: Handshake of messages between two neighboring stations during purification, after entanglement

the same value. In this figure, it looks like these measurements are done at the same time, but in fact only in some particular cases this happens. We will show the asymmetric case which is produced when purification is attempted between neighbor stations, as in Fig. 5.17.

5.5.3 Purification Policy

The fidelity obtained after a Bell pair is purified depends on the initial fidelity of both Bell pairs chosen. Previous work has proved that selecting Bell pairs of a fidelity within a range of values (banded purification scheme) boosts the final fidelity to higher values, therefore reducing the necessary purification steps [38]. In our work we chose this scheme as our purification policy.

5.6 Entanglement Swapping Control Layer

For networks which have more than two stations, further steps are required. As mentioned in the introduction, in this work we focus on *entanglement swapping*. A node in the middle of the network waits until it has two high-fidelity Bell pairs, one to each node it wants to couple. Then, this middle node performs the Bell state measurement that splices the two short Bell pairs into a longer one. As a consequence of this operation, the fidelity of the extended new Bell pair drops, and further purification may be necessary. ESC and PC are repeated until we have an end-to-end Bell pair of sufficient fidelity for our application.

5.6.1 Finite State Machine

There are two types of finite state machines for this layer, one for the nodes that hold entanglement to two other nodes and make the decision of swapping, and another one for the nodes that are informed that a swapping operation was performed. The finite state machine for the stations that are in the middle and make the decision to swap is shown in Fig. 5.18. As we can see in the figure, control of qubits come from a lower layer (purification in this work). If the qubit remains in this layer idle, the fidelity will decrease and when the lower threshold is reached (timeout shown in the figure), the control of the qubit is given again to the purification layer in order to boost the fidelity above the threshold once more. When another Bell pair arrives and this layer determines that it is a match for swapping, teleportation is done and the results of the measurement of the qubits are sent together with the reports of the swapping operation informing the remote nodes where the qubits are now entangled to. The final fidelity after swapping varies depending on the initial fidelity of both Bell pairs as shown in Fig. 3.2. Finally the qubits in this node are reset to be used for other operations.

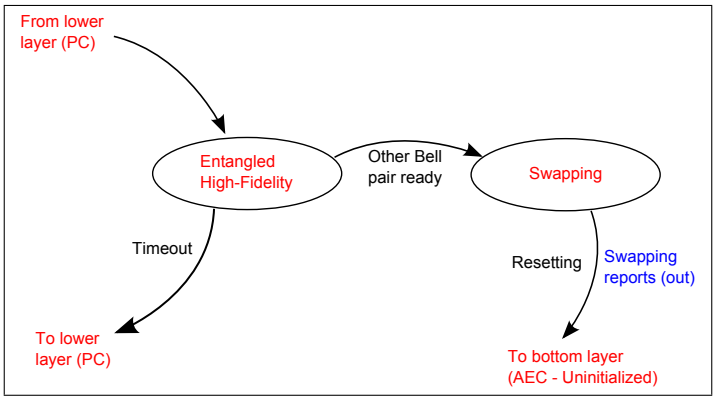


Figure 5.18: Finite State Machine for Entanglement Swapping Control for a middle node

On the other hand, the stations which receive the swapping reports follow the finite state machine as in Fig. 5.19. As we can see in this figure, control of qubits come from a lower layer and may go back there after a timeout meaning that the fidelity has decreased and purification is needed. If a swapping report is received, the new fidelity value is checked to see if more purification is needed. If the fidelity is below a threshold value, control of the qubit is sent to the lower layer for purification. If fidelity is still above the threshold, this layer checks to see if the new entangled qubit belongs to the node where the final communication is to be done. If that is the case, control of the qubit is sent to the application layer. If the desired node has not yet been reached, the node will remain in this state until further reports arrive.

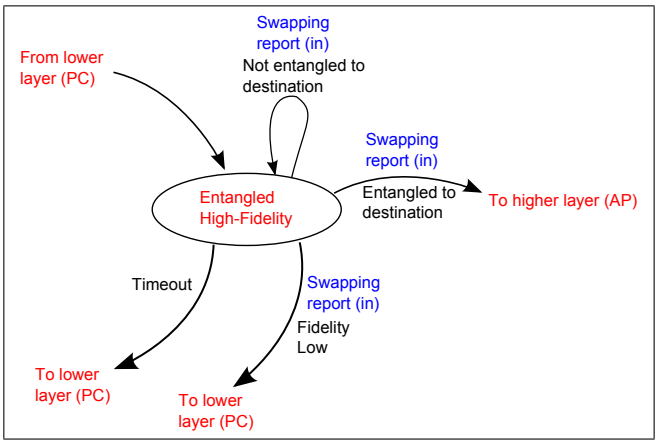


Figure 5.19: Finite state machine for Entanglement Swapping Control for an end node

In Fig. 5.20 we can see the type of messages exchanged and the functions of each message for the ESC layer.

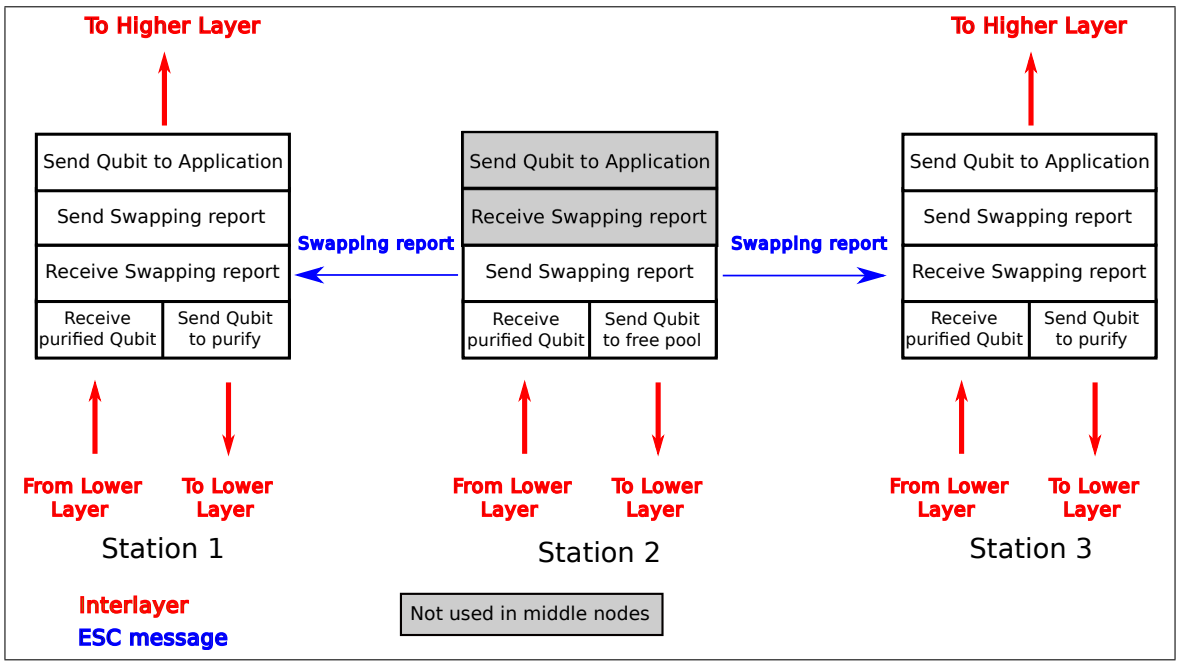


Figure 5.20: Message type exchanged in Entanglement Swapping Control Layer

5.6.2 Messages

The messages exchanged between stations are proposed to be composed of the fields shown in Fig. 5.21.

Where:

- Version: Version of the implementation of this protocol. To allow future updates and compatibility.
- Type: Type of message. This is to identify the different messages that each layer may use.
- Source Qubit: Qubit which has been swapped.
- Destination Qubit: Qubit which's entanglement distance has been extended.
- New Destination Address: Address of the node where the entanglement has been extended.
- New Destination Interface: Interface where the entanglement has been extended.
- New Destination Qubit: Qubit number of the node where the entanglement has been extended.
- Density Matrix x Element: Value of the element of the diagonal of the new density matrix after swapping. 8 bytes-long to represent double precision floating point numbers.
- Zero Time: Time of the interaction of the laser and the qubits. This is used to calculate how the fidelity decreases with time. 8 bytes-long to represent double precision floating point numbers.
- Epoch: This number represents how many times the qubits have been reset.
- M bit: 1 if the node that receives this message is where the teleported qubit is sent.
- M1 bit: Result of the measurement of qubit to be teleported.
- M2 bit: Result of the measurement of qubit to be used for teleportation.

As we explained in Sec. 2.7, for the teleportation algorithm to work, 2 classical bits of information need to be sent to the station where the qubit is teleported. For the swapping function, teleportation is applied in one of the links, in order to extend the entanglement length of the first link. Therefore, in swapping, only the station

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version								Type								Source Qubit															
32	Destination Qubit																New Destination Address															
64	New Destination Interface																New Destination Qubit															
96	Density Matrix 1st element																															
128	Density Matrix 1st element																															
160	Density Matrix 2nd element																															
192	Density Matrix 2nd element																															
224	Density Matrix 3rd element																															
256	Density Matrix 3rd element																															
288	Density Matrix 4th element																															
320	Density Matrix 4th element																															
352	Zero Time																															
384	Zero Time																															
416	Epoch																M	M1	M2	Padding												

Figure 5.21: Messages exchanged in Entanglement Swapping Control Layer

that receives the teleported qubit needs the results of these measurements (bits M1 and M2 in the packet) and we specify which such a station with the M bit set in 1.

There is no handshake of messages for this layer, just the report that a swapping operation has been done. This can be seen in the Fig. 5.22.

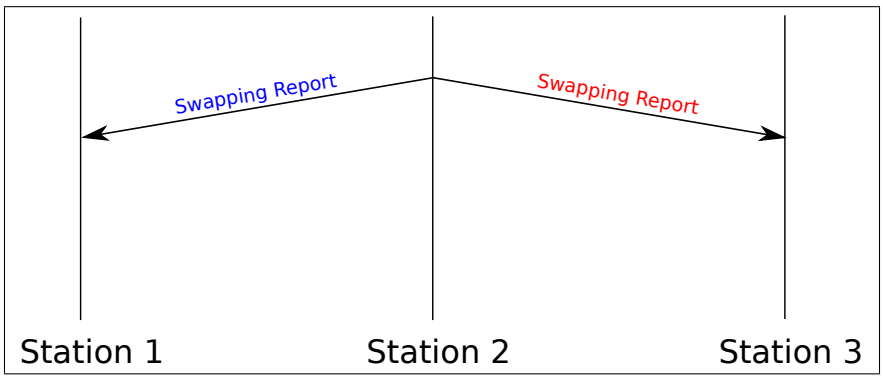


Figure 5.22: Reports sent in Entanglement Swapping Control Layer. Message in red has the bit M set to 1, meaning that the receiver will have to perform quantum operations to conclude the teleportation of the qubit. Message in blue has the bit M set to 0, no operations are required for the receiver.

5.7 Application Layer

Some applications were introduced in Sec. 1.1. Each of them will have their own application functions and messages, being teleportation a function included in many of them. However, for the scope of this thesis we show an example of what this layer should be for the case of teleportation of quantum states only. Though not a real application, it allows us to study the behavior of the application layer on top of the others. Therefore, the proposed application packet is shown in Fig. 5.23.

Where:

- Version: Version of the implementation of this protocol. To allow future updates and compatibility.

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version								Type								Source Qubit															
32	Destination Qubit																M1	M2	Padding						Epoch							
64	Epoch								Reserved																							
96	Zero Time																															
128	Zero Time																															

Figure 5.23: Messages exchanged in the Application Layer

- Type: Type of message. This is to identify the different messages that each layer may use.
- Source Qubit: Qubit in the trasmitter that is used for teleportation
- Destination Qubit: Qubit in the receiver that is used to receive the teleportation.
- M1, M2 bits: Measurements of the qubits done in the transmitter.
- Epoch: This number represents how many times the qubits have been reset.
- Zero Time: Time when the measurements for teleportation are done.

5.8 Quantum Repeater Multiplexing

In the following subsections we propose different types of quantum multiplexing to manage resources where competing flows exist. Each proposal differs in how the resources are handled (shared by time, by space, or without any resource management at all, or dedicated use with no sharing). These multiplexing schemes are derived from those in common use in classical networks and can be used fairly directly in quantum repeater networks.

In classical networks, stations are usually competing for shared resources, the service is offered on a best effort basis, and the path selection is a difficult task handled by many different routing algorithms. Quantum networks will not be an exception. Based on analogy with the classical model, to solve the shared resources issues, we propose the following quantum multiplexing schemes and evaluate several schemes which we are simulating. The principal operational difference between classical and quantum is that entanglement swapping is a truly distributed stochastic computation, with state held at each station along the path.

Fig. 3.5 shows a simple circuit which is used here to explain multiplexing in quantum networks. Each circle represents a repeater node, connected (e.g., by fiber) to one or more other nodes. In this and subsequent figures (except Fig. 7.4), we draw multiple lines to indicate the maximum number of multiple Bell pairs that can exist simultaneously between neighboring nodes, limited by the availability of qubit memories at the repeaters. In this section, we assume that each pair of neighboring nodes can have a maximum of 8 Bell pairs at the same time.

5.8.1 Quantum Circuit Switching

Classical version

Used in standard telephony service, reserves a complete circuit for one connection and no other traffic is allowed to use those resources at the same time. When the connection is finished, the circuit is released, allowing other stations to transmit. This way of handling traffic provides the best service to the end-points using the circuit, however the usage of the network resources may not be optimized as in the case that the station enabled to transmit has no traffic to send, or the traffic is very low compared to the link capacity. Moreover, if other stations need to transmit, they need to wait until the circuit is enabled for them, adding a stand-by time to the total time needed for transmitting.

Quantum version

In quantum circuit switching, a path between the nodes that wish to communicate is selected, and all of the qubit memory and quantum channel resources are reserved for the exclusive use of that flow. As this approach is equivalent to the original analog telephone system, in which wires were connected to form a physical electrical circuit end to end, it is known as *circuit switching*. When the connection is finished, the circuit is released, allowing other stations to request access to the resources. This way of handling traffic provides the best service to the end-points using the circuit, as no interefering traffic is present. However, the usage of the resources of the entire network may not be optimized. Although the resources along the entire circuit are dedicated to a particular flow, the flow is not required to actually use the network; it can pick and choose when to use it. The resources of a particular path through the network, then, can be substantially underutilized, depending on the behavior of the communication flow.

Moreover, if other stations need to transmit but access to their chosen path through the network is blocked because another flow is using the resources, then they need to wait until they can allocate the necessary set of links to form a circuit, adding a stand-by time to the total time needed for transmitting. Considering the circuit shown in Fig. 3.5, station A competes with station C for the shared resources between nodes E and F. If node A requires a circuit to be enabled to connect to node B, all of the resources along the path are reserved and configured to form a circuit, as shown by the red lines in Fig. 5.24. During this time, stations C and D cannot allocate an end-to-end circuit, as entanglement swapping done at stations E and F is restricted only to the enabled flow. Once this circuit is released by station A, the resources become free, and station C can request the connection as shown in Fig. 5.25.

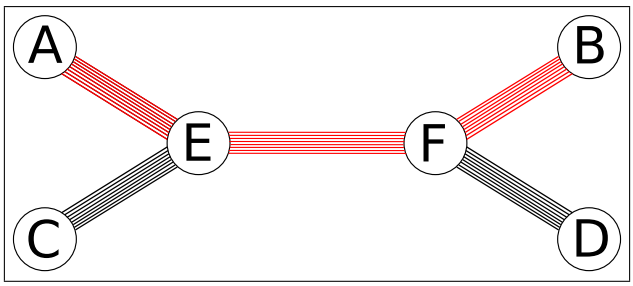


Figure 5.24: Circuit Switching. A to B enabled.

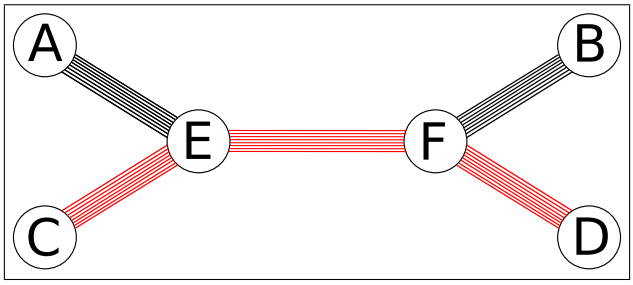


Figure 5.25: Circuit Switching. C to D enabled.

This approach may work well for short bursts of traffic, switching circuits from AB to CD once each burst is transmitted. For large amounts of traffic this may not work well, as a station may be forced to wait for an indeterminate amount of time before the circuit is released.

5.8.2 Time Division Multiplexing (TDM)

Classical version

Designed as a way to handle multiple phone calls in a same line, while transportating voice as data. Each phone call is assigned a time slot, and the line must provide enough bandwidth to accomodate all the calls. A further

implementation was to transport any type of data, and this type of lines could assign many time slots for some end-points if more speed in the communications was required.

Quantum version

To reduce the potentially very long waiting time associated with circuit switching, it can share the resources of a link. In *time division multiplexing*, assignment of resources is done in a round-robin fashion for each station that requests a connection. Time is divided into a set of fixed-length *time slots*, and one (or possibly more than one) time slot is assigned to each connection that requests use of the link. During the time slot, the shared resources can only be used for the selected flow. After a time slot ends, the resources are assigned to the next flow. If more throughput is needed for one flow, more than one time slot can be allocated to that flow, allowing a greater usage of the resources during a round. Every station will wait its turn to transmit, and if a station is not ready to use the channel on its turn, the resources go unused. This behavior is represented in Fig. 5.26, alternating between two connections using two time slots. During time slot 1, node C cannot communicate with D. During time slot 2, node A cannot communicate with node B. The sequence is repeated whether or not traffic is present in the network.

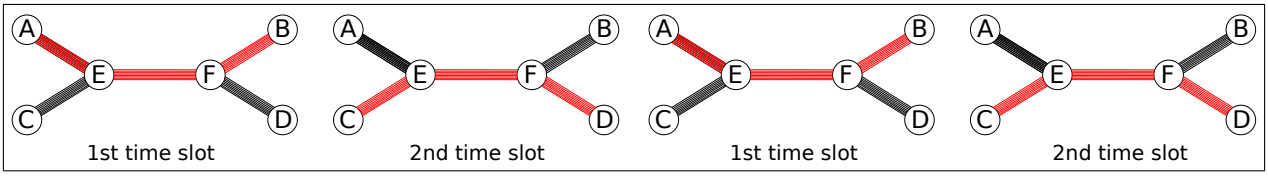


Figure 5.26: Time division multiplexing of two communication flows. The red lines indicate links enabled for the active flow.

5.8.3 Buffer Space Multiplexing

As an alternative means of sharing, we can consider *buffer space multiplexing*, dividing the available qubit memory space and assigning part of it to each flow. Fig. 5.27 represents this multiplexing scheme for the case of two flows. Half of the qubits from the link between stations E and F are assigned to the flow from station A to B and the other half are assigned to the flow CD. As a result, two different and separated circuits are built, so traffic from AB will not interfere with traffic from CD.

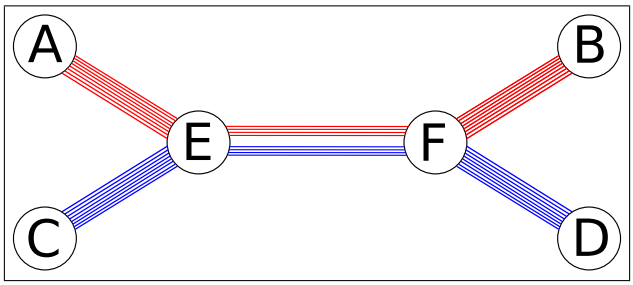


Figure 5.27: Buffer Space Multiplexing.

Though the circuits are now independent, this benefit has a cost; half the resources naturally will be slower. The statistical availability of Bell pairs may result in less than half the throughput. Under some circumstances, using the preferred banded purification scheme becomes impossible [37].

5.8.4 Statistical Multiplexing

Classical version

IP networks are the best example of packet switching, as a best effort service. No resource reservation is done. If a packet arrives to a router, it is forwarded based on the available bandwidth at that time. In case no resources

are available, the packet may be queued or even dropped.

Quantum version

In *statistical multiplexing*, no resource reservation is done, and the service is provided *best effort* fashion. In our example, when an EF Bell pair completes its purification, node E checks for the presence of ready-to-swap AE and CE Bell pairs. If only one is ready, swapping is performed, $AE + EF \rightarrow AF$ or $CE + EF \rightarrow CF$. If both AE and CE are awaiting swapping, one of the pairs is chosen at random, without regard to the waiting time or state of other traffic. For dynamic and complex topologies it is very difficult to implement the other multiplexing schemes, and the service is provided based on the current availability of resources. A well-known type of network using statistical multiplexing is the Internet.

5.8.5 Aggressive Use of Resources

In the discussions in the preceding subsections, we have assumed that all resources either are in dedicated operation for a particular flow, or are idle. In practice, however, even when only a partial path is available, a flow can make use of those resources while awaiting its turn for currently-unavailable resources.

In Fig. 5.24, for example, use of the EF link is temporarily allocated to the AB flow. Rather than sitting idle, the CE and FD links can prepare Bell pairs over the single hops, so that the minimum of time is wasted when the EF link becomes available and is allocated to the CD flow. In Sec. 7.1 we have investigated setting the target fidelity very high for the portions of the path that would otherwise be idle. We have found that links for which there is competition can be utilized more effectively in this fashion, allowing the throughput (measured in Bell pairs per second) of two flows sharing one link to be substantially higher than a single flow.

Chapter 6

Quantum Network Simulator

6.1 Introduction

In quantum networks, due to the huge amount of messages, measurements and the probabilities of success of quantum operations, make a calculation completely analitic of the protocols behaviour is a very hard task. In addition, the construction of quantum repeaters is not yet possible due to many physical challenges, so build a network and measure the protocols performance is not possible either. The only way to study our protocols today is by using a quantum network simulator. Not only can we study the right implementation of our quantum protocols, but we can see inside the system in ways not possible in the real world, and many interesting results can also be obtained from simulations, such as the effects of routing optimization, reduction of the number of packets, and even what are the minimum values of time that quantum memories should support in order to allow the construction of quantum repeaters. Another advantage of the simulations is that we can test many scenarios (bigger, faster or just different) rapidly. For our work we chose Omnet++ as a network simulator upon which to build.

6.2 Simulating Distributed Quantum States

Running simulations of quantum systems in classical computers comes with a problem apart from the computational power needed. As we explained in Sec. 2, qubits that are entangled produce instantly correlated measurements. Representing superluminal signals in a network simulator can be done with a zero-delay channel, through which we could send this information immediately after one qubit is measured, or by accessing some shared memory space that all the nodes can read and write without the simulator adding any delay. This approach works well for simulations done on a single computer, but it becomes a difficult task to solve when we run the simulations in a distributed environment. How can we send the results of a measurement instantly to another computer as delays on the network will always be present? Simulating this quantum property in such environment remains as a future work. In our simulations, the measurements done in the purification layer and the superluminar signals were not actually simulated, allowing this implementation to be done in a distributed computing infrastructure. We explain in detail such implementation in Sec. 6.5.4.

6.3 Why Omnet++?

Omnet++ is not created to work only with TCP/IP networks, but allows users to create their own layers, define finite state machines and messages. For these reasons, for its flexibility, we chose it as our network simulator. For this work, a lot of calculations related to quantum operations are done in C++ and we use Omnet's libraries to run the simulations.

6.3.1 Omnet++ general description

Omnet++ allows us to define the configuration parameters of a node and the network topology in a .ned extension input file (NED language topology description), where all the hardware and the connections between

stations, distances, bandwidth, etc are defined. These .ned files are used to compile the simulator. A configuration file (.ini) makes changes to the simulations easy without having to compile everytime, this file being read at running-time. We can add any parameters in the .ini file, but in our work we can choose from different topologies (written in various .ned files), select the multiplexing schemes, change the resources (qubits in the transmitters and in the receivers), etc. Finally, the .msg files are used to define the messages used by the different layers. In the next sections we show an example of these types of files. The code is mainly written in .cc and .h files as any regular C++ code.

Files used in our project

The files we used for our simulation are detailed in Table 6.3, Table 6.2, Table 6.1.

File Name	Description
omnetpp.ini	Configuration file to specify the simulation's parameters
Stations.ned	Definition of each node's variables, properties, like the number of qubits, interfaces
SPIE.ned	Network definition for SPIE's conference simulation
Dumbbell.ned	Network definition for the protocol simulation
package.ned	Information about the license. GPL license for this work

Table 6.1: Run-time Omnet configuration files in the simulator

File Name	Description
Stations.cc, Stations.h	Main code, defines Stations class
Gates.cc, Gates.h	Defines Gates class
Qubits.cc, Qubits.h	Defines Qubits class
Flows.cc, Flows.h	Defines Flows class
Layer1.cc, Layer1.h	Defines Layer 1 class
QubusPE.cc, QubusPE.h	Defines Qubus Physical Entanglement class
Layer2.cc, Layer2.h	Defines Layer 2 class
AEC.cc, AEC.h	Defines Acknowledged Entanglement Control class
Layer3.cc, Layer3.h	Defines Layer 3 class
PC.cc, PC.h	Defines Purification Control class
Layer4.cc, Layer4.h	Defines Layer 4 class
ESC.cc, ESC.h	Defines Entanglement Swapping Control class
Layer5.cc, Layer5.h	Defines Layer 5 class
APP.cc, APP.h	Defines Application class

Table 6.2: Classes defined in the simulator

6.3.2 Omnet Operation

Omnet++ is a discrete event system simulator, which means that events happen at discrete instances in time. Every event in Omnet++ is defined are the arrival of a message (from other node or a self message sent as a timer). In order to begin the simulations, some stations need to start sending messages at time zero. The next event in time would be when one of these messages is first received by any station, which in this case is fixed by the distance between nodes. The simulation continues to run as long as there are any messages left in the network. When the last message is received, and no further messages are generated, then the simulation ends. Omnet's simulations require the creation of three main functions in our own class that inherits all the functions from Omnet. These functions are: `initialize`, `handleMessage` and `finish`. Following is the explanation of them:

1. `initialize()` This is the first function of our code that is executed for each of the nodes defined in the NED file. It includes the initialization of all the variables used in the simulation. As this type of simulation

File Name	Definition of the messages
Protocol.msg	Main class of messages, the other classes inherit from this one
Interlayer.msg	Used between protocols layers in the same node
Layer1.msg	Used by layer 1. Implementations inherit from this one
QubusPE.msg	Used by the Qubus mechanism, an instance of Layer 1
Layer2.msg	Used by layer 2. Implementations inherit from this one
AEC.msg	Used by the Acknowledged Entanglement Control, an instance of Layer 2
Layer3.msg	Used by layer 3. Implementations inherit from this one
PC.msg	Used by the Purification Control, an instance of Layer 3
Layer4.msg	Used by layer 4. Implementations inherit from this one
ESC.msg	Used by Entanglement Swapping Control, an instance of Layer 4
Layer5.msg	Used by layer 5. Implementations inherit from this one
APP.msg	Used by the application, an instance of Layer 5
TrafficPath.msg	Definition of the messages used to identify flows

Table 6.3: Omnet message files in the simulator, used during compilation of simulator.

continues as long as there are any messages left in the system, we need to generate some messages to start the simulation. The first messages sent are from the initializing method to the Link Entanglement Control layer in use. This layer will be in charge of communicating the free qubits to the physical layer. Then successfully entangled qubits are passed from this layer to the upper one. In addition, self-messages are sent and used as timers to define when to send other laser pulses, and for multiplexing timing.

2. `handleMessage()` Everytime a message is received by a station, this function is called and the message processed. This is the main code of the simulation, and where we need to continue sending messages in order to keep the simulation alive.

3. `finish()` This function is used to print reports showing the state of the simulations. It is usually called when we ask the simulation to terminate.

We have added a *state* property that identifies not only which layer currently controls the qubit, as well as the state of the finite state machine of the layer. In this context, "state" refers to the current role of the qubit in the system, rather than the density matrix or the state vector. Table 6.4, table 6.5 and table 6.6 show all the possible states of the qubits.

State Number	State Definition
0	Uninitialized
1	Unentangled - just initialized
2	Interim Entangled - Entanglement Attempt

Table 6.4: Qubit states for Qubus and Entanglement Control Layer

State Number	State Definition
3	Entangled with low fidelity
40	Purify Attempt - qubit to be purified
41	Sacrifice Attempt - qubit to be sacrificed
42	My Half Purified - sent measurements
43	Her Half Purified - receive measurements

Table 6.5: Qubit states for Purification Control Layer

State Number	State Definition
5	Qubit has a high-fidelity level and it is ready for swapping

Table 6.6: Qubit states for Entanglement Swapping Control Layer

6.4 Organization of Simulator Files

6.4.1 Topology Design (NED files)

In this type of file is where we can define the topologies, how many nodes and how they are connected, distance between them, and whether they are transmitters or receivers. Once this file is created, further configuration of the parameters need to be done in the `Omnet.ini` file. In Fig. 6.1 we show an example of the file `Dummbell.ned` where we define the stations, and the connections between them:

Fig. 6.2 shows the graphical representation of this network.

6.4.2 Messages Design (MSG files)

As Omnet allows the creation of network layers, the MSG files are used to define the fields used in the messages exchanged between the layers. In these files, only the definition of variables is needed. Omnet will do the hard work and make .cc and .h files from the very simple MSG, which can be easily used in the main code. There are two types of messages. One which is sent from one station to the other, and the delay for those messages to arrive is based on the distance between nodes. The other type of message is called a self-message or scheduled-message. Nodes send these messages to themselves to schedule different functions, such as sending laser pulses and expiration of multiplexing timers.

`Protocol_Message.msg` is a main type of message from which the rest of the layers and implementation of layers will be inheriting its properties. Each general layer specified in the protocol architecture inherits from this one. Therefore, `Layer1.msg`, `Layer2.msg`, `Layer3.msg`, `Layer4.msg`, `Layer5.msg` are all subclasses. Finally, each implementation of each layer, inherits from them. In our work, QubusPE is a subclass of Layer1, AEC a subclass of Layer2, PC a subclass of Layer3 and ESC a subclass of Layer4.

In Fig. 6.3 we show an example of the message type used in `QubusPE.msg`.

6.4.3 Configuration file (INI file)

This configuration file named `Omnet.ini` is where we define what parameters to load into the simulations. For example, we can address different NED files with different topologies and configure different parameters for each of them. The parameters that are included in this file are those which allow us to produce different results in different scenarios. These models are grouped under defined names, which can be chosen at the beginning of the simulations. In Fig. 6.4 is an example of part of this file used for Circuit-Switching for the dumbbell network, where the total number of stations, qubits, multiplexing type, the network ID (to identify which network we simulate) and the end-to-end wished fidelity target are specified.

6.5 Layers Implementation

6.5.1 Interlayer Messages

As mentioned before, the layers need to exchange messages reporting the result of quantum operations, requesting new operations to be done or just to transfer the control of the qubit to other layer. In order to identify the destination layer of these messages and the requested function, we propose the following type of message shown in Fig. 6.5.

Where:

- Destination Layer: Identifies which layer should receive this message.
- Protocol Type: This field identifies what type of message is encapsulated.

```
network Dumbbell
{
    submodules:

        StationA: Stations {
            parameters:
                TOTAL_GATES = 1;
                GATE0_TX = true;
                GATE0_SOURCE = true;
                @display("i=device/pc4_l;p=50,50;is=l");
        }
        StationB: Stations {
            parameters:
                TOTAL_GATES = 1;
                GATE0_TARGET = true;
                @display("i=device/pc4_l;p=300,50;is=l");
        }
        StationC: Stations {
            parameters:
                TOTAL_GATES = 1;
                GATE0_TX = true;
                GATE0_SOURCE = true;
                @display("i=device/pc4_l;p=50,250;is=l");
        }
        StationD: Stations {
            parameters:
                TOTAL_GATES = 1;
                GATE0_TARGET = true;
                @display("i=device/pc4_l;p=300,250;is=l");
        }
        StationE: Stations {
            parameters:
                TOTAL_GATES = 3;
                GATE0_TX = true;
                GATE0_TRUNK = true;
                @display("i=block/routing;p=100,150;is=l");
        }
        StationF: Stations {
            parameters:
                TOTAL_GATES = 3;
                GATE1_TX = true;
                GATE2_TX = true;
                GATE0_TRUNK = true;
                @display("i=block/routing;p=250,150;is=l");
        }
    connections allowunconnected:
        StationA.gate[0] <--> Channel <--> StationE.gate[1];
        StationE.gate[0] <--> Channel <--> StationF.gate[0];
        StationE.gate[2] <--> Channel <--> StationC.gate[0];
        StationF.gate[1] <--> Channel <--> StationB.gate[0];
        StationF.gate[2] <--> Channel <--> StationD.gate[0];
}
```

Figure 6.1: Excerpt of Dumbbell.ned

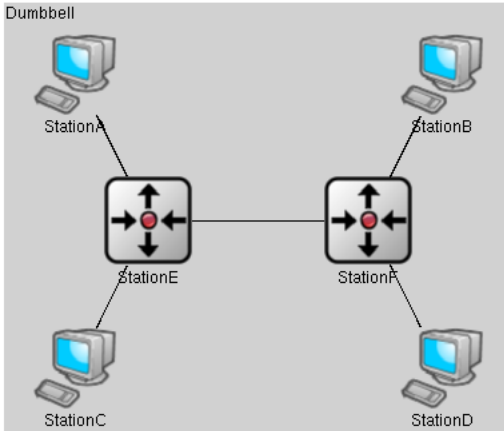


Figure 6.2: Dumbbell network in Omnet

```
cplusplus {{
#include "Layer1_m.h"
}}
message Layer1Message;
//
// TODO generated message class
//
message QubusPE_Message extends Layer1Message {
    Kind = 100;    // Defines the protocol type. Constant in all the packets
    int Pulse[51]; // Pulse[0] contains the number of pulses in the train
    int Epoch[51];
}
```

Figure 6.3: QubusPE.msg

- Payload: Here the encapsulated message is placed.

In Table 6.7 we show the protocol types and their functions.

Protocol Type	Protocol	Description
1000	QubusPE	Receives information of qubits to entangle
2000	AEC	Receives entanglement report from physical layer
3000	PC	Receives qubits to purify
4000	ESC	Receives high-fidelity qubits to swap
5000	APP	Receives qubits to be used by the application

Table 6.7: Interlayer protocol codes

6.5.2 Physical Layer

This layer will wait for the reception of a message coming from the AEC layer. This message contains the information of which qubits are free. Based on that information, the physical layer will generate a packet to send to the neighbor node. We are not simulating the train of quantum pulses that would be sent from the transmitter to the receiver, we only simulate the classical message that would go along with this pulse, containing the number of pulses sent and the addresses of the qubits in the transmitter that were interacted

```
[Config Dumbbell-Circuit_Switching]
network = Dumbbell
record-eventlog = false
**.TOTAL_STATIONS = 6
**.MULTIPLEXING = 1        # 1: Circuit Switching. 2: TDM. 3: Statistical MUX. 4: Spatial MUX
**.TOTAL_TX_QUBITS = 50    # Total number of qubits per transmiter gate
**.TOTAL_RX_QUBITS = 16    # Total number of qubits per receiver gate
**.GAMMA = 10              # For dephasing. (Not in use)
**.PRINT_MATRIX = 0        # 1: Print the values of the density matrixes  0: Do not Print
**.NETWORK_ID = 1       # 1: Static (AINTEC paper's simulation), 2: Unassigned, 3: Unassigned
**.END_TO_END_FIDELITY = 0.98  #  Minimun wished end to end fidelity
```

Figure 6.4: Excerpt of Omnet.ini

Bit offset	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Destination Layer								Protocol Type																Payload							
32 ...	Payload ...																															

Figure 6.5: Message type exchanged betwen layers

by each of the pulses. Other information such as the epoch (number of times each qubit was reset), source and destination nodes and gates are also included. When the neighbor receives this message it will generate a random number for each pulse that is compared with the probability of entanglement success (which is based on the technology in use, fiber optic attenuation and distance). This layer will save the results of success and failure in an array which is then sent to the upper layer via an internal message. This report tells whether each pulse has failed or succeeded to entangle, and with which qubit in the receiver they succeeded. Also a parameter called *ZeroTime* attaches the time when the entanglement operation was done. This value is used in order to calculate the degradation of the fidelity as time passes.

The responsibilities of this layer can be represented by the Fig. 6.6.

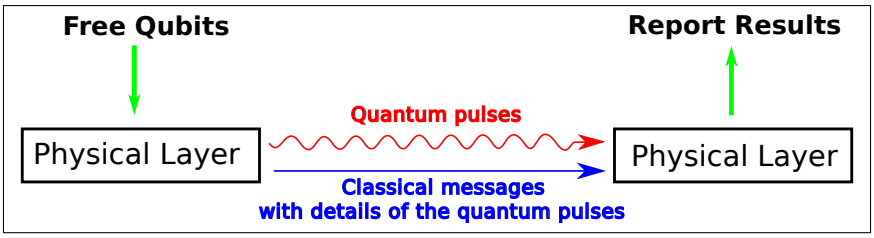


Figure 6.6: Functions of the Physical Layer

6.5.3 Acknowledged Entanglement Control Layer

This layer receives the entanglement report message from the physical layer and then updates the entanglement information for the local qubits. This information includes to which node, gate and qubit each local qubit is entangled, what is the current density matrix (fidelity comes from it) and the time when the entanglement was done. After doing this, the state of the qubit is changed to entangled (code 3) and a message is sent to the purification layer for each qubit that was entangled. In addition, a message report is sent to the transmitter node with all this information. When the transmitter receives this report, it updates the variables also and sends a message to the purification layer reporting which qubits are ready for purification. Both transmitter and receiver will send the failed qubits to a pool of free qubits, that would be ready for next entanglement

attempt operation.

6.5.4 Purification Layer

When qubits are received by this layer, they are classified in different bands based on their fidelity [37]. Therefore, qubits within the same band and chosen for purification. As explained in the purification algorithm, when two qubits are in the same band, we need to choose which qubit to be purified and which one to be sacrificed. In order to make consistent decisions, the nodes that attempt purification must decide the same Bell pair to purify and the same one to sacrifice. We propose that the transmitter will choose the qubit with the smallest qubit number to be purified and the biggest one to be sacrificed. The receiver will have the information of the remote qubits entangled, therefore can choose the same order based on the trasmitter's qubits. In addition, in the real world purification algorithm the nodes would measure the qubits and send the results which could be 0, 1, or fail (in case that the local gates failed). On the reception of the measurements, if both measured the same ((0 & 0) or (1 & 1)) then the purification would succeed. Any other combination, like (0 & 1), (1 & 0), or any Fail condition would make the purification fail. In the simulator we didn't implement this in the same way, as doing this would mean the additional complication of superluminal signals produced by the quantum measurements. Therefore, we decided to send a measurement value of 1 with a probability of $\sqrt{Prob.Succ}$, and if both nodes send a value of 1 (this event would happen with a probability equal to the probability of success of the purification), then we say that the purification succeeded.

6.5.5 Entanglement Swapping Control Layer

This is probably the most difficult layer to implement. The reason is that it needs to take care of what qubits to swap, to which destination, at what time, choose a fidelity threshold for each link. And all these decisions rise in difficulty when the networks become more complex or dynamic. Qubits with some pre-defined level of fidelity arrive here from the purification layer. After the swapping decision is made (in this work we defined the swapping decisions statically, together with the fidelity thresholds for each link), the qubits are swapped, local qubits are reset and a swapping report is sent to both remote nodes which now hold the new Bell pair. In this report, the new density matrix is also included and the time of the swapping operation. When the remote stations receive the report, they update the local variables (where the remote entangled qubit information is also held) and then it is decided whether more purification is needed (in this case, the control of the qubit is sent to the purification layer via a message). If the fidelity is still good enough, then this layer checks whether we need to do more swapping or if the Bell pair extends end-to-end and needs to be sent to the application layer via a message. If it hasn't, then the qubit is kept by this layer until other swapping operation can be done.

6.5.6 Application Layer

As there could be many possible application layers for quantum networks, each one doing many different operations on the end-to-end qubits, here we focused on the only purpose of teleporting qubits from one node to another. In order to do this, we simulate the required time for measurement of the quantum local gates, and we send the results of such measurement (two classical bits) via a message to the end node. When this message arrives, another delay is added to simulate the local operations in the node, and then we assume that the qubit has been teleported. At this time, we increment the counter used to calculate throughput in Qubits/sec.

6.6 Main Code Description

Message exchange and timing are functions done by Omnet++. The simulation starts at time zero, and control is given to each node in the network. The order in which this control is given is based on the Node Id that each node has. These Ids are created by Omnet and chosen sequentially from the order in which the nodes were defined in the `Stations.ned` file, and are used to address the nodes when sending messages. After the initialization routine, messages are sent from the transmitter stations in order to start and keep the simulation alive. These messages represent the entanglement attempts done physically by laser pulses, and are sent once at a fixed amount of time (fixed to half a round trip time to the neighbor, $100\mu sec$ in our simulations). Each entanglement attempt is represented by one single message containing information about many pulses sent together. In addition, self messages are also sent as a timer to force another shot of laser pulses to be generated.

Omnet++ keeps track of these timers and will mark the reception of any message as an event. The order in which messages are processed for a particular time is again based on the Node Id of the nodes, giving priority to the smaller ids. Every time a message arrives at a node, the function `Stations::handleMessage` is called.

The pseudo-code for the function handleMessage is shown in Fig. 6.7.

Currently the simulator provides a one-line report everytime one qubit is teleported, and information such as the receiver Id, the total number of qubits teleported at that time, the end-to-end fidelity obtained is given. When the simulations are done, a final report shows how many qubits have been teleported for each station.

6.7 Key differences between proposed protocols and actual simulations

The simulations we run tried to completely followed the proposed protocols, however, some of the quantum properties could not be simulated and we needed to adapt our code without affecting the results of the simulations. Layers 1 and 2 (Qubus Physical Entanglement and Acknowledged Entanglement Control) where simulated as they were proposed. The only difference is with real-world quantum repeater networks, as we didn't simulate the physical detection system, but we just obtained a random number and based on that we decided whether the entanglement succeeded or failed. For layer 3 (Purification Control) the proposed protocol exchanges the measurement of entangled qubits, which can basically be "0", "1", or gate failure. Measuring entangled particles require the use of superluminar signals difficult to simulate in a distributed computing environment. Though our simulations were run on a single computer, we wanted to make it compatible for running greater simulations in such environments. How did we solve this? The measurement value is not important for the purification algorithm, what really matters is that both stations measure the same value. So, we calculated the probability of success of purification and send the value "0" from each station under the purification process with a probability of square root the value of success. So, the probability of both stations sending the value "0" is the probability of success of purification, and if both stations send and receive a value of "0", they will agree that the purification was successful. The proposed algorithm will send the results of the measurement and then compare what each station measure and what they receive, being a match of results considered a purification success. Finally, layer 4 (Entanglement Swapping Control) in the proposed protocol does teleportation, and there is a quantum measurement and the transmission of two classical bits as a result of such measurement. The station receiving these bits applies some quantum gates to its qubit based on the information received. As in purification algorithm, this includes measurement of entangled qubits. However, teleportation always succeeds (if we consider perfect quantum gates), so we didn't include the simulation of this measurement and the use of the quantum gates. Though assumptions of measurement were done in our simulations, they didn't affect the performance of the proposed protocols, or the results of the simulations.

```
If (msg != SelfMessage)
   MessageforUs = CheckAddress(msg)
If (MessageforUs)
   Switch(Protocol)
      case 10:
         ReceiveShooterTimer(msg)
      case 20:
         ReceiveTDMTimer(msg)
      case 30:
         ReceiveTrafficPath(msg)
      case 100:
         QubusLayer.ReceiveMessage(msg)
      case 200:
         AECLayer.ReceiveMessage(msg)
      case 300:
         PCLayer.ReceiveMessage(msg)
      case 400:
         ESCLayer.ReceiveMessage(msg)
      case 500:
         APPLayer.ReceiveMessage(msg)
      case 99:
         InterLayerMessage *Msg = check_and_cast<InterLayerMessage *>(msg)
 switch (Msg->getDestinationLayer())
    case 1:
       QubusLayer.ReceiveMessage(Msg)
    case 2:
       AECLayer.ReceiveMessage(Msg)
    case 3:
       PCLayer.ReceiveMessage(Msg)
    case 4:
       ESCLayer.ReceiveMessage(Msg)
    case 5:
       APPLayer.ReceiveMessage(Msg)
else if (Protocol == 30)
   ReceiveTrafficPath(msg)
PrintEntanglementReport
```

Figure 6.7: Pseudo-code for the function handleMessage

Chapter 7

Evaluation

7.1 Multiple hops

We simulated a qubus mechanism, with 20km hops with a number of qubits in each transmitter is 50, and 16 in the receivers. In all of our simulations, we use a target end-to-end fidelity of 0.98. We have run simulations for two cases: only one flow, and two flows competing for shared resources in the network shown in Fig. 7.1. Both flows are over three-hop paths (AEFB and CEFD), with the middle hop (EF) being a shared link and hence the throughput bottleneck. Bell pairs created on the EF link are assigned randomly to be used for the AB or CD flows. Used naively, the first and third hops on each path will remain idle half of the time.

We hypothesized that careful tuning of the purification thresholds might better balance the system. If we raise the required fidelity on the under-utilized links, can we reduce the fidelity penalty incurred by entanglement swapping and improve aggregate performance? To test this hypothesis, we ran simulations with the purification threshold of each link set to several different values, while keeping the end-to-end target $F = 0.98$. We used two values for the first and third hops, $F_1 = 0.98$ and $F_1 = 0.99$, with the second case requiring an additional round of purification on those single hops. We then varied the fidelity threshold of the middle hop, $F_2 = 0.86, 0.94, 0.98$, altering the number of purification rounds required over that single hop. The aggregate throughput of the flows for each of the twelve simulations is shown in Fig. 7.2. With $F_1 = 0.99$, the performance of two flows doubled that of a single flow, as seen in the green and purple bars on the right of the figure. Fig. 7.3 plots the final, delivered fidelity of the same cases. In the $F_2 = 0.98$ cases, the tradeoff for higher throughput is that the final fidelity just barely clears our established end-to-end target of $F = 0.98$.

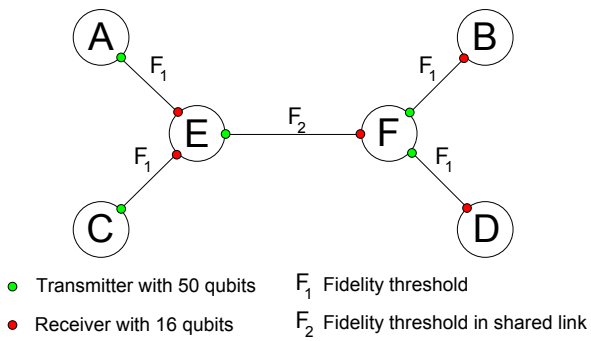


Figure 7.1: Simulated dumbbell network

7.2 Shared resources

To quantitatively compare these multiplexing approaches, we simulate the network shown in Fig. 7.4. We want to study the behavior of a traffic flow from station A to station K in the presence of competition from additional

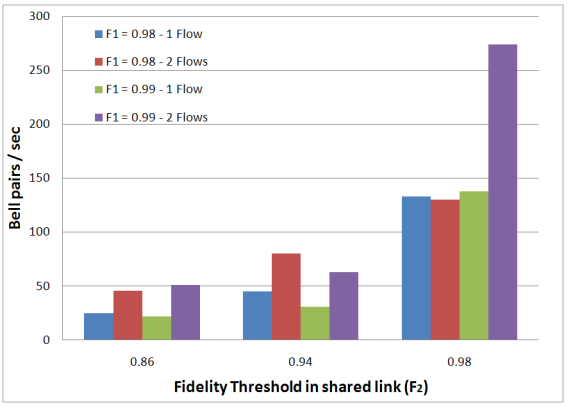


Figure 7.2: Throughput in Bell pairs/sec, one flow versus two flows

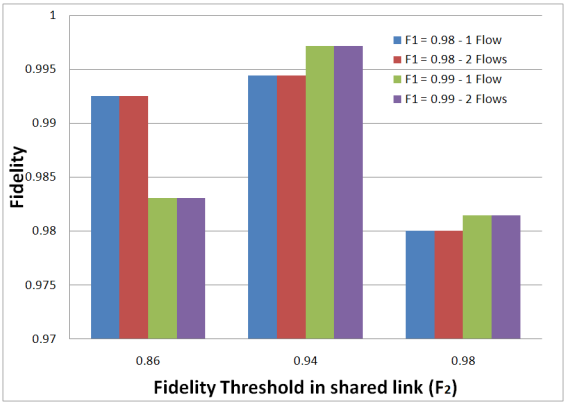


Figure 7.3: End-to-end fidelity of teleported qubits

flows in the network, using different multiplexing schemes for the shared network links. The simulation scenarios are listed in Table 7.2.

We focus on the creation of Bell pairs, so we use the magnitude of the $|\Phi^+\rangle\langle\Phi^+|$ component of the Bell-basis density matrix as our fidelity. In prior work [37], we have shown that a fidelity of 0.98 is a good target end-to-end fidelity; using higher values results in performance that varies dramatically depending on local gate errors and memory errors, making it difficult to interpret results in the context of the question at hand. In this paper, we retain $F = 0.98$ as our operational goal, running purification over both individual links and longer distances to achieve and maintain this fidelity.

We are using the purification scheme of Deutsch [12], with reordering of the states using local operations to ensure that $|\Phi^+\rangle\langle\Phi^+|$ remains the largest and $|\Phi^-\rangle\langle\Phi^-|$ is the smallest component. This operation is non-deterministic, and the probability of success is higher for qubits with higher fidelity with respect to a Bell pair than those with lower fidelity. Many operations may be required in order to obtain a high-fidelity Bell pair ready to be used as a resource in the network.

Our simulation is organized into a protocol stack directly modeled on separation of the functions described in Sec. 3.6. We simulated a qubus mechanism as the physical protocol, Deutsch purification as the protocol for error management, and entanglement swapping as the protocol responsible for making the entanglement span from end to end. Each link's length is fixed at 20km over a fiber optic of 0.17 dB/km loss. The number of qubits in each transmitter is 50, and 32 in the receivers. We measure performance in throughput of teleported qubits per second, simulating the measurement to complete the teleportation of the qubits in the transmitter and the time needed for the classical information to arrive at the destination.

Five flows are used in the five scenarios listed in Table 7.2. First we studied each flow separately using circuit switching, and measured the throughput that the network can provide when no other traffic is present. This multiplexing scheme provides the best performance for the active circuit, and is used here as a reference

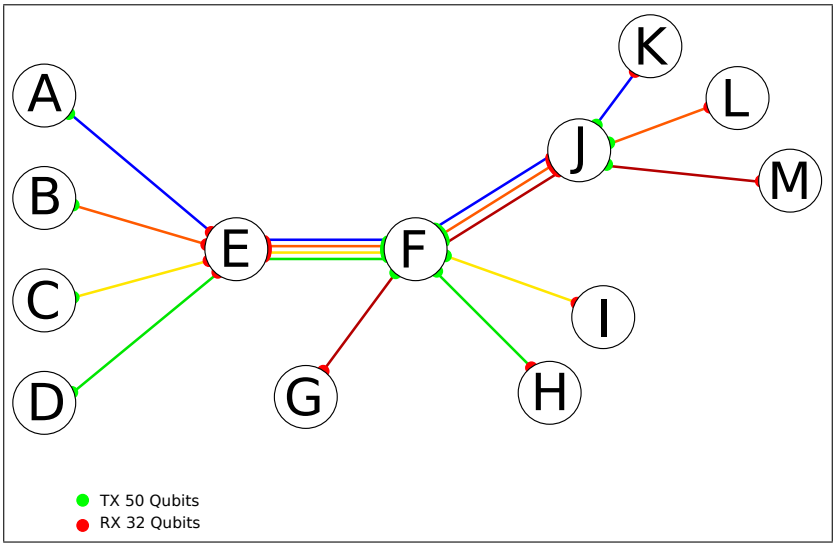


Figure 7.4: Simulated network. Each color of line represents one communication flow. The four-hop AK flow is our primary flow, and the others are enabled and disabled in various cases to test the impact of multiplexing schemes on that primary flow.

Hardware	Properties	Comments
Fiber length	20 Km	Each hop
Fiber attenuation	0.17 dB/km	
Transmitter qubits (TX)	50	
Receiver qubits (RX)	32	
End-to-end fidelity target	0.98	Fixed threshold
Contested links purification threshold	0.98	
Uncontested links purification threshold	0.99	
Base pair fidelity	0.633	
Entanglement success probability	0.36	

Table 7.1: Hardware-Configuration

to compare the other schemes when we try to accommodate additional flows. Then we simulate the other multiplexing schemes.

In all of our simulations, we use a target end-to-end fidelity of 0.98, but using the same level throughout the entire network results in portions of the network sitting idle while waiting for a flow for its turn to use a shared resource, as discussed in Sec. 5.8.5. In Sec. 7.1, we have shown that setting a higher fidelity threshold for those under-utilized resources can result in a net gain in performance. In the simulations presented here, we follow the same strategy, and set the purification threshold for the uncontested links (AE, BE, CE, DE, JK, JL, etc.) to 0.99, while the contested links EF and FJ are set to 0.98.

7.2.1 Circuit Switching

Table 7.3 shows the measured throughput for each flow when no other traffic is present. In this particular network, with this traffic pattern, only a single connection can operate at a time, so the aggregate throughput of the network is limited to that of a single connection. We use these values and the total as a reference to compare the other mulxtiplexing schemes.

Scenario	Flows	Comments
1	AK alone	Baseline case
2	AK+CI	Competition on one link
3	AK+BL	Competition on two links
4	AK+CI+DH	Two competing flows on one link
5	AK+BL+CI+DH+GM	Several competing flows in different parts of the network

Table 7.2: Traffic flows

Flow	Number of hops	Throughput
AK	4	64 Qubit/sec
BL	4	65 Qubit/sec
CI	3	133 Qubit/sec
DH	3	135 Qubit/sec
GM	3	124 Qubit/sec
SUM	-	521 Qubit/sec

Table 7.3: Maximum traffic per flow using circuit switching.

7.2.2 Statistical Multiplexing

Fig. 7.5 shows the performance of our five scenarios, compared to the throughput of an ideal, impossible-to-achieve case of totally uncontested access to resources for all flows (the "SUM" line in Table 7.3). The first case is the baseline, where we only activate the flow AK. This is the maximum possible for this circuit. In the second case we enabled the flows AK and CI only, with competition for shared resources only on one link (between E & F). In the third case we enabled flows AK and BL, with competition for resources on two links (between E & F and F & J). In the fourth case we enabled flows AK, CI and DH, where two flows compete on one link with AK (between E & F). Finally, we studied all the flows together, where there are several competing flows in different parts of the network and measured the throughput for each one.

7.2.3 TDM

For this scheme, flows are enabled based on time. Each station is assigned a time slot, so for each simulation we have the same number of time slots as the number of flows. For the cases AK+CI and AK+BL, two time slots are assigned; in AK+CI+DH, three timeslots, and for the last case with all the flows active, we used five time slots. The measurements are shown in Fig. 7.6.

7.2.4 Buffer Space Multiplexing

In this simulation, we divide the shared resources into the number of flows in transit. For the first case AK+CI, there are two competing flows in the link E-F, so we assign half the resources for each flow (16 qubits in station E and 25 qubits in station F). For AK+BL we have two flows, but in this case there are two shared links, E-F and F-J, so stations E and J were assigned 16 qubits, station F was assigned 25 qubits for each flow in each link. In the third case, AK+CI+DH, three flows compete for the resources in the link E-F, the resource assignment is: station E, 11 qubits for each flow AK and CI, and 10 qubits for flow DH; station F, 17 qubits for flows AK and CI, and 16 qubits for flow DH. Finally, for all the flows active, between stations E & F there are four flows from stations A, B, C & D, so the total number of resources for each flow is 8 qubits in station E. In station F, 17 qubits were assigned for flows AK and BL, and 16 qubits for flows CI and DH. Between stations F and J there were only three flows, giving 17 qubits for flows AK and BL, and 16 qubits for the flow GM in station F. In station J, 11 qubits were assigned to flows AK and BL, and 10 qubits to GM. Fig. 7.7 shows the throughput for buffer space multiplexing compared with the ideal, uncontested case of five flows with dedicated resources.

Flows	AK	AK+CI	AK+BL	AK+CI+DH	AK+BL+CI+DH+GM
AK	64 Qubit/sec	61 Qubit/sec	56 Qubit/sec	40 Qubit/sec	29 Qubit/sec
BL	0 Qubit/sec	0 Qubit/sec	59 Qubit/sec	0 Qubit/sec	33 Qubit/sec
CI	0 Qubit/sec	127 Qubit/sec	0 Qubit/sec	92 Qubit/sec	68 Qubit/sec
DH	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	89 Qubit/sec	75 Qubit/sec
GM	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	52 Qubit/sec
SUM	64 Qubit/sec	188 Qubit/sec	115 Qubit/sec	221 Qubit/sec	257 Qubit/sec

Table 7.4: Throughput using statistical multiplexing

Flows	AK	AK+CI	AK+BL	AK+CI+DH	AK+BL+CI+DH+GM
AK	64 Qubit/sec	59 Qubit/sec	55 Qubit/sec	49 Qubit/sec	25 Qubit/sec
BL	0 Qubit/sec	0 Qubit/sec	56 Qubit/sec	0 Qubit/sec	23 Qubit/sec
CI	0 Qubit/sec	125 Qubit/sec	0 Qubit/sec	95 Qubit/sec	65 Qubit/sec
DH	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	83 Qubit/sec	53 Qubit/sec
GM	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	35 Qubit/sec
SUM	64 Qubit/sec	184 Qubit/sec	111 Qubit/sec	227 Qubit/sec	201 Qubit/sec

Table 7.5: Throughput using Time Division Multiplexing

7.2.5 Discussion

We have studied different multiplexing schemes in order to recommend a mechanism for sharing resources in a multi-user network, and ultimately to be able to predict the performance of a given network under certain traffic patterns. To recommend a scheme, we want to know how the performance changes as traffic changes, beginning with the aggregate performance of the network. We also want to be assured that short-distance flows are not able to "shut out" long-distance flows and prevent them from making forward progress.

Above, we alluded to the fact that two flows using one shared link can sometimes exceed the performance of a single flow. This counter-intuitive behavior arises because the unshared resources can continue to improve beyond their minimum required fidelity threshold, making more efficient use of the shared resources when they do gain access. This can be seen clearly in the AK+CI and AK+BL cases, where two flows *each* achieve 86% to 95% of their ideal performance. The behavior of a single flow, then, under at least some circumstances, can be said to be only minimally affected by the presence of a second flow using one or two of the same links. The addition of a third flow (AK+CI+DH) raises the total throughput again, but begins to have significant impact on the performance of each flow relative to the ideal case. Knowledge of this behavior can be used to guide to design of a network topology, if the expected traffic pattern is understood.

The total throughput of all five flows is highest for the statistical multiplexing case, achieving 257 teleported qubits per second, compared to 228 qubits per second for buffer space multiplexing and 201 qubits per second for time division multiplexing. Statistical multiplexing substantially outperforms the other two schemes (by 13% and 28%, respectively), though the specific numbers are traffic- and network-specific. For the cases with fewer flows, the performance advantage was smaller, only a few percent. We expect to further confirm this advantage by simulating additional networks and traffic patterns (especially larger, more complex networks) in future work.

The above analyses assess the steady-state throughput of our flows. Let us briefly compare these schemes for a variant with a fixed amount of work by assumimg that all five flows in Table 7.2 issue their initial requests to use the network at the same time and run until 100 qubits have been teleported. For the circuit switched case, first we would run the AK flow to completion, then the BL flow. Next, the CI and GM flows can run at the same time, because they use independent parts of the network. Fourth and last would come the DH flow. This would take approximately $100/64 + 100/65 + 100/133 + 100/135 = 4.6$ seconds. In contrast, in statistical multiplexing all five flows begin work at the same time. Using a slightly more complex calculation to take into account that the three-hop CI, DH and GM flows would complete more quickly than the four-hop AK and BL flows, we estimate that all 500 qubits could be teleported in 2.7 seconds, or 1.7× as fast. TDM and buffer space multiplexing require more detailed simulation to produce reasonable estimates of their performance under

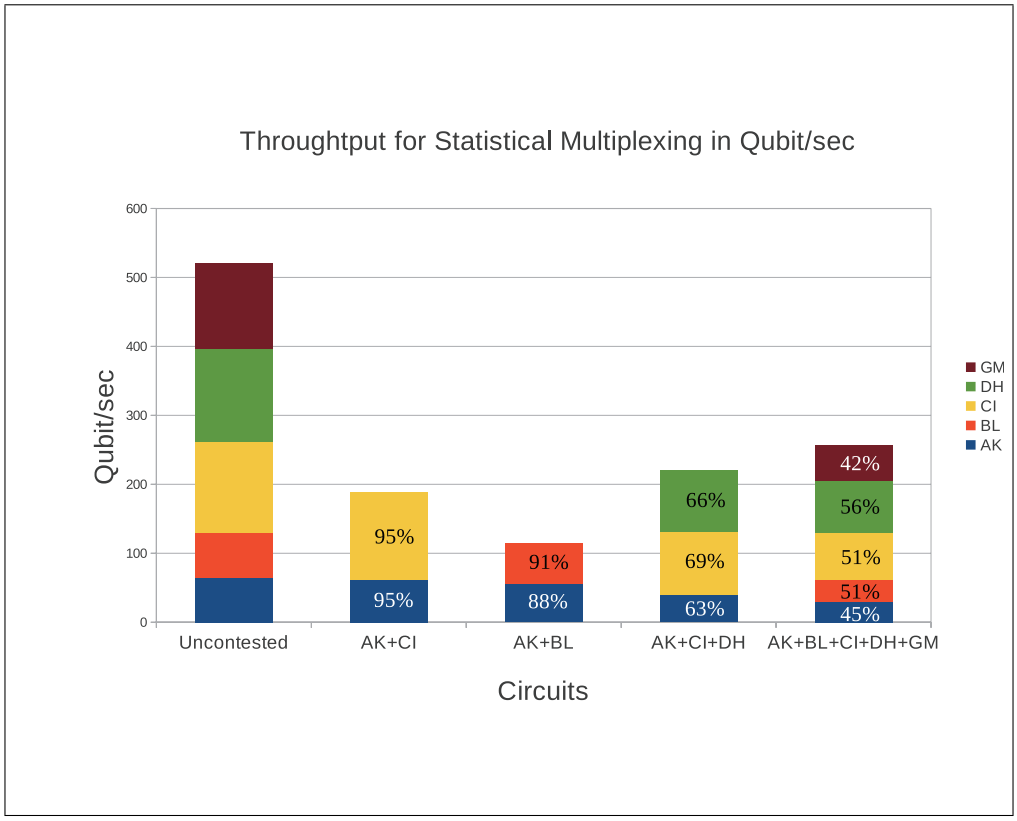


Figure 7.5: Throughput of statistical multiplexing compared to five uncontested flows (left).

changing workloads. We defer simulation and analysis of such dynamic activity, including flows that start and stop at different times in different parts of the network, for future work.

Although statistical multiplexing has the highest throughput, we might suspect that with no control on resource use, it is potentially susceptible to being unfair to some flows. In particular, we are concerned that long-distance flows, which naturally react more slowly to changing conditions than shorter flows, may be penalized more than short ones as the dynamic network state changes.

We evaluated the fairness of the multiplexing schemes using Jain's fairness measure for resource allocation [23],

$$\mathcal{J}(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

for the five-flow cases for each multiplexing method. A fairness of $\mathcal{J} = 1.0$ indicates perfectly even distribution of resources among the users, while $\mathcal{J} = 1/n$ indicates that one user acquired all of the resources, shutting out all other users. Because the maximum capability of each flow differs even when given uncontested access to all links, we applied the measure to the set of throughputs normalized to each flow's circuit-switched throughput, the percentages shown in Figures 7.5, 7.6 and 7.7. Statistical multiplexing, with a range of 42% to 56% of maximum, has a nearly perfect fairness of $\mathcal{J} = 0.99$. Buffer space multiplexing likewise comes in with $\mathcal{J} = 0.99$. TDM, despite the relatively large spread from 28% to 49%, also has an excellent fairness of $\mathcal{J} = 0.97$. In particular, the four-hop flows fall in the middle of of the group in terms of performance degradation, giving us no reason to infer that long-distance flows are penalized more heavily, though further confirmation with longer flows is desirable. From these values we conclude that all three multiplexing schemes share contested resources fairly.

Finally, we observe that statistical multiplexing is simpler to implement that any scheme requiring explicit resource management, whether circuit switching, TDM or buffer space multiplexing. Both the software implementation and the network protocols have fewer requirements using statistical multiplexing, reducing implementation and deployment cost.

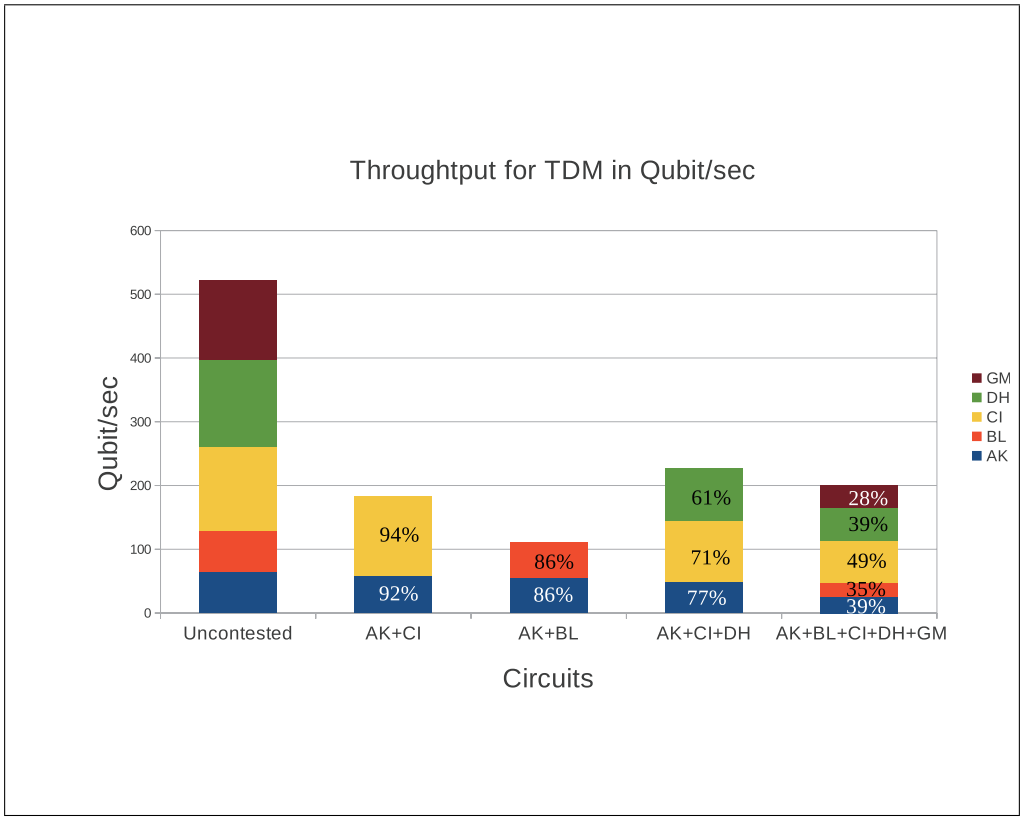


Figure 7.6: Throughput of TDM compared to uncontested flows.

We have presented our simulations of four different approaches to managing the sharing of resources in purify-and-swap quantum repeater networks. Future work includes addition of one or more memory decoherence mechanisms, as well as simulations of additional scenarios and additional statistical tests of behavior. However, the data in this paper gives a clear indication that statistical multiplexing akin to that used in the Internet will give us good performance and robust, fair operation.

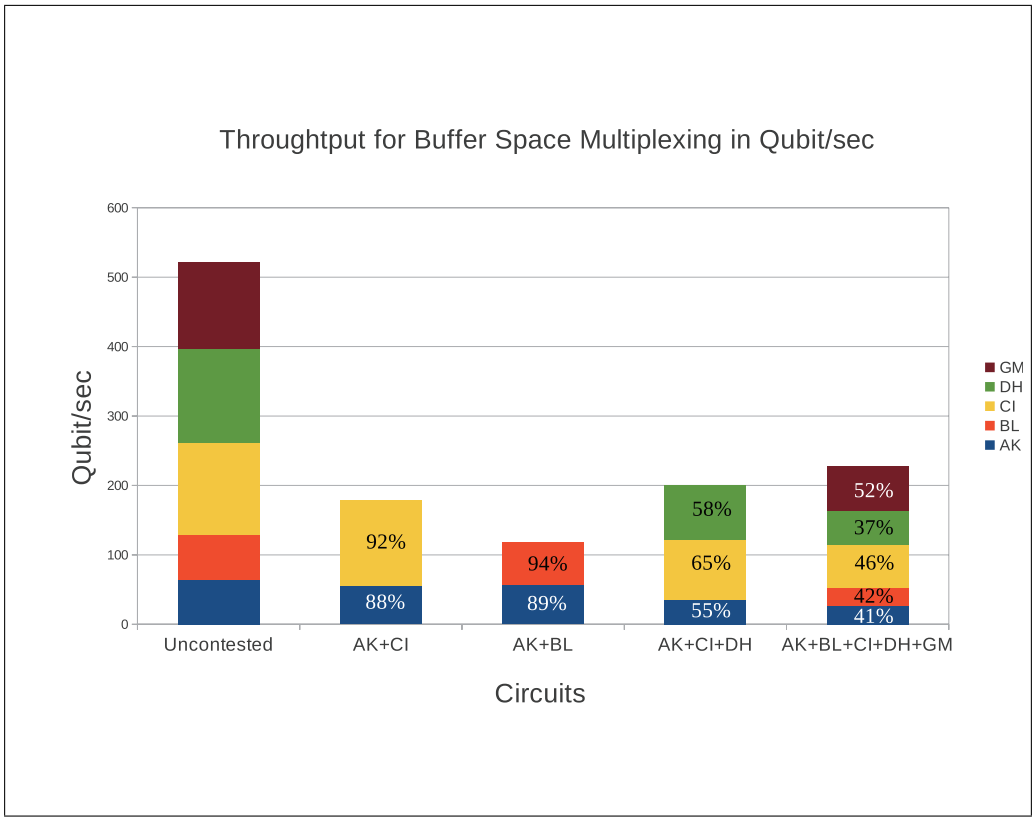


Figure 7.7: Throughput of buffer space multiplexing compared to uncontested flows.

Flows	AK	AK+CI	AK+BL	AK+CI+DH	AK+BL+CI+DH+GM
AK	64 Qubit/sec	56 Qubit/sec	57 Qubit/sec	35 Qubit/sec	26 Qubit/sec
BL	0 Qubit/sec	0 Qubit/sec	61 Qubit/sec	0 Qubit/sec	27 Qubit/sec
CI	0 Qubit/sec	123 Qubit/sec	0 Qubit/sec	87 Qubit/sec	61 Qubit/sec
DH	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	78 Qubit/sec	50 Qubit/sec
GM	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	0 Qubit/sec	64 Qubit/sec
SUM	64 Qubit/sec	179 Qubit/sec	118 Qubit/sec	200 Qubit/sec	228 Qubit/sec

Table 7.6: Throughput using buffer space multiplexing

Chapter 8

Conclusions

The proposed protocols for networks of quantum repeaters can be used on real-world quantum repeaters once they are built. Not only can they be used for repeaters, but other implementations like system area networks are also feasible. We have studied the behavior of these protocols in networks with shared resources and we found out that for the topologies tested, the best multiplexing scheme was statistical multiplexing. More complex and different topologies should be tested in the future in order to make a firm statement regarding that statistical multiplexing is the best approach for all the networks. Not only did it prove to have a better performance, but also its simplicity of implementation is a good argument against the other schemes. We also found out that proper tuning of uncontested links improves the network performance while spending time of unused links doing purification and reducing the number of end-to-end purification steps, which are slower due to the addition of more hops and the longer propagation delay for the classical messages.

All this work was done without including memory degradation due to decoherence, which is an important characteristic that will give more reality to the simulations of a physical system. Clearly after adding decoherence to our simulations, the performance will be reduced as fidelity will decrease, and more steps of purification will be needed to keep a high level of fidelity for swapping. Quantum memory degradation remains as a future work, and will be implemented in the simulator in order to extend our work.

For this thesis we kept track of the addresses of nodes, interfaces and qubits where the Bell pairs are held. This approach worked well and may also be applied to bigger networks. However, in order to virtualize the networks, such information should be hidden from the nodes and referring to each Bell pair with a label only, and the only information that the nodes should concern about is which node is this label referring to. This remains as a future work and implementation in the simulator.

A problem with simulations of quantum systems is that when measuring one particle belonging to an entangled group, immediate influence is produced to the rest of the particles, without any signal propagation, therefore running simulations in a distributed computing environment should consider how to manage this information. How to handle superliminar signals? For the purification algorithm and teleportation, we fixed the values of the measurements, without affecting the obtained results. However, for applications that need real measurements, we cannot fix the values, and simulations of superluminar signals must be done.

Designing the protocols such that the stations were able to make the same consistent decisions without exchanging unnecessary messages proved to be a hard task. As in our work, the Entanglement Swapping Control layer was assigned fixed *routes*, we didn't have to face many of the difficult tasks that this layer has for more complex topologies. Intelligent networks must be able to find what is the *middle* of the network and attempt to produce Bell pairs from this node to the end-points, even though dynamic changes may occur and further calculations must be done in order to keep the *middle* node updated. Other difficult but very important issue is to find the best order of operations like purification and swapping. How much purification should we do before swapping? Different topologies will have different optimal order for such operations, and this task should also be addressed by the Entanglement Swapping Control layer and kept updated if the network changes.

In this work, we provided a simulator that allows the use of independent layers for quantum repeaters and we also proposed a set of protocols that can be applied to the future quantum networks. It is our intention to make the code open-source, and we would like other laboratories to use our code to run simulations of their own protocols by just replacing the layers of our code with their own. This code may also be a reference for the code used in the future real-world quantum repeaters.

Bibliography

[1] Alex Abramovici, William E. Althouse, Ronald W.P. Drever, Michael E. Zucker, and et.al. Ligo: The laser interferometer gravitational-wave observatory. *Science*, 256:325–333, 1992.

[2] M. Ben-Or and A. Hassidim. Fast quantum Byzantine agreement. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 481–485. ACM, 2005.

[3] Charles H. Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Phys. Rev. Lett.*, 70(13):1895–1899, Mar 1993.

[4] Katherine L. Brown, William J. Munro, and Vivien M. Kendon. Using quantum computers for quantum simulation. *entropy*, 12:2268–2307, 2010.

[5] Harry Buhrman and Hein Röhrig. *Mathematical Foundations of Computer Science 2003*, chapter Distributed Quantum Computing, pages 1–20. Springer-Verlag, 2003.

[6] Iulia Buluta and Franco Nori. Quantum simulators. *Science*, 326:108–111, 2009.

[7] L. Childress, J. M. Taylor, A. S. Sørensen, and M. D. Lukin. Fault-tolerant quantum repeaters with minimal physical resources and implementations based on single-photon emitters. *Phys. Rev. A*, 72(5):052330, Nov 2005.

[8] C. W. Chou, H. de Riedmatten, D. Felinto, S. V. Polyakov, S. J. van Enk, and H. J. Kimble. Measurement-induced entanglement for excitation stored in remote atomic ensembles. *Nature*, 438:828–832, Dec 2005.

[9] Chin-Wen Chou, Julien Laurat, Hui Deng, Kyung Soo Choi, Hugues de Riedmatten, Daniel Felinto, and H. Jeff Kimble. Functional quantum nodes for entanglement distribution over scalable quantum networks. *Science*, 316(5829):1316–1320, 2007.

[10] Isaac L. Chuang. Quantum algorithm for distributed clock synchronization. *Phys. Rev. Lett.*, 85(9):2006–2009, Aug 2000.

[11] J. I. Cirac, P. Zoller, H. J. Kimble, and H. Mabuchi. Quantum state transfer and entanglement distribution among distant nodes in a quantum network. *Phys. Rev. Lett.*, 78(16):3221–3224, Apr 1997.

[12] David Deutsch, Artur Ekert, Richard Jozsa, Chiara Macchiavello, Sandu Popescu, and Anna Sanpera. Quantum privacy amplification and the security of quantum cryptography over noisy channels. *Phys. Rev. Lett.*, 77(13):2818–2821, Sep 1996.

[13] Ellie D'Hondt. *Distributed quantum computation: A measurement-based approach.* PhD thesis, Vrije Universiteit Brussel, July 2005.

[14] L.-M. Duan and H. J. Kimble. Scalable photonic quantum computation through cavity-assisted interactions. *Phys. Rev. Lett.*, 92(12):127902, Mar 2004.

[15] W. Dür, H.-J. Briegel, J. I. Cirac, and P. Zoller. Quantum repeaters based on entanglement purification. *Phys. Rev. A*, 59(1):169–181, Jan 1999.

[16] W. Dür and H.J. Briegel. Entanglement purification and quantum error correction. *Rep. Prog. Phys.*, 70:1381–1424, 2007.

[17] Chip Elliott, David Pearson, and Gregory Troxel. Quantum cryptography in practice. In *Proc. SIGCOMM 2003*. ACM, ACM, August 2003.

[18] ESIA, JEITIA, KSIA, TSIA, and SIA. International technology roadmap for semiconductors. Technical report, ESIA and JEITIA and KSIA and TSIA and SIA, 2009. http://public.itrs.net/.

[19] Austin G. Fowler, David S. Wang, Charles D. Hill, Thaddeus D. Ladd, Rodney Van Meter, and Lloyd C. L. Hollenberg. Surface code quantum communication. *Phys. Rev. Lett.*, 104(18):180503, May 2010.

[20] Nicolas Gisin, Grégoire Ribordy, Wolfgang Tittel, and Hugo Zbinden. Quantum cryptography. *Rev. Mod. Phys.*, 74(1):145–195, Mar 2002.

[21] D. Gottesman, T. Jennewein, and S. Croke. Longer-baseline telescopes using quantum repeaters. *Arxiv preprint arXiv:1107.2939*, 2011.

[22] Lov K. Grover. A fast quantum mechanical algorithm for database search. *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[23] Raj Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.

[24] Liang Jiang, J. M. Taylor, Kae Nemoto, W. J. Munro, Rodney Van Meter, and M. D. Lukin. Quantum repeater with encoding. *Phys. Rev. A*, 79(3):032325, Mar 2009.

[25] Richard Jozsa, Daniel S. Abrams, Jonathan P. Dowling, and Colin P. Williams. Quantum clock synchronization based on shared prior entanglement. *Phys. Rev. Lett.*, 85(9):2010–2013, Aug 2000.

[26] H.J. Kimble. The quantum internet. *Nature*, 453:1023–1030, 2008.

[27] T. D. Ladd, P. van Loock, K. Nemoto, W. J. Munro, and Y. Yamamoto. Hybrid quantum repeater based on dispersive CQED interaction between matter qubits and bright coherent light. *New Journal of Physics*, 8:184, 2006.

[28] Duan L M, Lukin M D, Cirac J.I., and Zoller P. Long-distance quantum communication with atomic ensembles and linear optics. *Nature*, 414:413–418, Nov 2001.

[29] D. N. Matsukevich and A. Kuzmich. Quantum state transfer between matter and light. *Science*, 306(5696):663–666, 2004.

[30] W. J. Munro, K. A. Harrison, A. M. Stephens, S. J. Devitt, and Kae Nemoto. From quantum multiplexing to high-performance quantum networking. *Nature Photonics*, 2010.

[31] W. J. Munro, R. Van Meter, Sebastien G. R. Louis, and Kae Nemoto. High-bandwidth hybrid quantum repeater. *Phys. Rev. Lett.*, 101(4):040502, Jul 2008.

[32] S. Olmschenk, D. N. Matsukevich, P. Maunz, D. Hayes, L.-M. Duan, and C. Monroe. Quantum Teleportation Between Distant Matter Qubits. *Science*, 323(5913):486–489, 2009.

[33] R. Reichle, D. Leibfried, E. Knill, J. Britton, RB Blakestad, JD Jost, C. Langer, R. Ozeri, S. Seidelin, and DJ Wineland. Experimental purification of two-atom entanglement. *Nature*, 443(7113):838–41, 2006.

[34] Takahiko Satoh and Rodney Van Meter. Path selection in heterogeneous quantum networks. *10th Asian Conference on Quantum Information Science (AQIS)*, 2010.

[35] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *SFCS '94: Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[36] P. van Loock, T. D. Ladd, K. Sanaka, F. Yamaguchi, Kae Nemoto, W. J. Munro, and Y. Yamamoto. Hybrid quantum repeater using bright coherent light. *Phys. Rev. Lett.*, 96(24):240501, Jun 2006.

[37] Rodney Van Meter, Thaddeus D. Ladd, W. J. Munro, and Kae Nemoto. System design for a long-line quantum repeater. *IEEE/ACM Transactions on Networking (TON)*, 17(3):1002–1013, jun 2009.

[38] Rodney Van Meter, Thaddeus D. Ladd, W. J. Munro, and Kae Nemoto. System design for a long-line quantum repeater. *IEEE/ACM Trans. Netw.*, 17(3):1002–1013, 2009.

[39] Rodney Van Meter, Joe Touch, and Clare Horsman. Recursive quantum repeater networks. *Progress in Informatics*, (8):65–79, 2011.

[40] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802, October 1982.

[41] Z. Zhao, T. Yang, Y.A. Chen, A.N. Zhang, and J.W. Pan. Experimental realization of entanglement concentration and a quantum repeater. *Phys. Rev. Lett*, 90(20):207901, 2003.