Master's Thesis (Academic Year 2023)

Link Management in a Quantum Network

Keio University

Graduate School of Media and Governance

Makoto Nakai

Abstract of Master's Thesis - Academic Year 2023

Link Management in a Quantum Network

Quantum networking is a new networking paradigm that allows specialized network nodes to share quantum states in order to achieve new applications. RuleSet-based communication protocol is known to be one of the practical communication protocols to establish a scalable quantum network. Ideally, the network should be able to handle the multiple connections and the subsequence management of the link-level Bell pairs in order to improve the overall performance and the aggregate use of the available resource on the network, even when the number of the active connections changes. However, the current state of the protocol only allows the establishment of a single connection and lacks the ability to terminate it and release the allocated resources on the link level. This thesis proposes the link management protocol for quantum network that allows the allocation and release of Bell pairs on the link level, which is the key to change of the number of active connections. It discusses the relationship between the connection management and the resource management in the link level. The behavior of the proposed protocol is validated by a set of numerical simulations on an existing quantum networking simulator.
Keywords :
1. Quantum Networking, 2. RuleSet-Based Communication Protocol, 3. Networking Protocol,

Keio University Graduate School of Media and Governance
Makoto Nakai

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The recent development of quantum technologies such as quantum computing, quantum networking and quantum sensing are expected to provide new capabilities. For example, quantum processors can theoretically simulate quantum systems whose size are intractable for their classical counterpart [5]. A quantum network realizes the secure generation of an encryption key [6] [7]. Quantum sensing allows the detection of sensitive physical properties such as magnetic field.[8].

Also, various applications can be realized by connecting these technologies via a quantum network, such as distributed quantum computing [9], blind quantum computing [10], precise clock synchronization [11] and improvement of the resolution of telescopes [12]

However, there are two major problems for transmitting quantum data to a distance location. One is the "non-cloning theorem" [13], which is the fact that a quantum state cannot be copied. Unlike a classical network, it is almost impossible to amplify a quantum state or send it forward because the quantum state will be heavily corrupted by the high probability of loss and high error rate. The other problem is that it is difficult to establish a Bell pair between nodes separated by a long distance, again due to a photon will be spoiled by the physical noise and photon loss.

These two problems can be solved by using nodes called quantum repeaters [14]. Quantum repeaters perform entanglement swapping [15] and purification [16], each of which extends two neighboring Bell pairs to a single longer Bell pair, and improves the fidelity of the Bell pair, respectively. These operations generate an end-to-end Bell pair that can be used by quantum teleportation [17], which is the protocol to send an arbitrary quantum state to a distant location.

Entanglement swapping and purification involve frequent message exchange with neighboring nodes in order to coordinate actions, such as entanglement swapping and purification, with neighboring nodes and those communication slow down the generation of an end-to-end Bell pair. However, a communication protocol [18] called a RuleSet-based communication protocol solves this problem by distribute an object called a RuleSet, which a sequence of operations to be executed at each node. This feature reduces the amount of unnecessary communication and improves the scalability of the entire network.

1.2 Research Contribution

Multiple connections should be established simultaneously in order to enhance the overall performance and robustness of the entire network. However, the previous work only proposes the method to allocate required physical Bell pairs and establish a single end-to-end Bell pair, in other words, single connection by consuming those physical resources. This thesis proposes a protocol to realize three important tasks, which are the negotiation of the set of connections are going to be established, when to switch from those in the previous round, and coordinated resource management between two nodes connected by each link. It also discusses the updated procedure of establishing a new connection and tearing down one of the existing connections while several connections are being established by applying the proposed protocol. The approach presented in this thesis is validated by the simulation of RuleSet-based quantum networks under several circumstances.

1.3 Thesis Structure

The structure of this thesis is as follows.

Chapter 2 provides the background knowledge to understand the key concepts readers would encounter throughout this thesis.

Chapter 3 presents the problem that this thesis addresses.

Chapter 4 offers the overview of the link management protocol and the messages required for its negotiation process.

Chapter 5 provides how link management protocol proposed in this thesis will be triggered after the process of connection setup and teardown. It also includes the pseudocode of methods that the node software need to execute and messages outside of the link management protocol.

Chapter 6 explains the detail of RuleSet-based quantum networking.

Chapter 7 presents several scenarios used to validate this protocol.

Chapter 8 offers the conclusion of this thesis and discusses future works.

Chapter 2

Background

2.1 Linear Algebra

This section provides the minimum background knowledge of advanced concepts in the field of linear algebra.

2.1.1 Unitary Operator

If an operator is unitary if it satisfies the following conditions.

$$U^{-1} = U^{\dagger} \tag{2.1}$$

$$UU^{\dagger} = U^{\dagger}U = I \tag{2.2}$$

2.2 Quantum Physics

This section provide the fundamental knowledge of quantum physics, which will make readers feel familiar with the concept and notations that they will encounter throughout this thesis.

2.2.1 Four Postulates

This subsection introduces the basic principles, in other words, the postulates of quantum mechanics [19].

Postulate 1

The state of any isolated physical system is a part of complex vector space with inner product known as the *state space* of the system. The system is completely described in the form of *state vector*, which is a unit vector in the system's state space.

Postulate 2

The evolution of a *closed* quantum system is described by a *unitary transformation.* That is, the state $|\psi\rangle$ of the system at time t_1 is transformed into the state $|\psi'\rangle$ of the system t_2 by a unitary operator U which depends only on the times t_1 and t_2.

$$|\psi(t_2)\rangle = U|\psi(t_1)\rangle \tag{2.3}$$

Postulate 3

Quantum measurements are described by a collection $\{M_m\}$ of *measurement operators.* These are operators acting on the space of the system being measured. The index m refers to the outcome that may be measured in the experiment. If the state of the quantum system is $|\psi\rangle$ prior to the measurement, then the measurement probability of the result m is

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{2.4}$$

and the state of the system after the measurement is

$$\frac{M_m|\psi\rangle}{\langle\psi|M_m^\dagger M_m|\psi\rangle} \tag{2.5}$$

The measurement operators satisfy the *completeness equation.*

$$\sum_m M_m^\dagger M_m = I \tag{2.6}$$

The completeness equation expresses the fact that probability sum to one.

$$1 = \sum_m p(m) = \sum_m \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{2.7}$$

Postulate 4

The state space of a composite physical system is the tensor product of the state space of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi_i\rangle$, then the joint state of the total system is $|\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle$.

2.2.2 Pure State

A pure state is the representation of quantum state of the whole system without the assumption of external noise.

Definition

A conventional computer uses a bit to represent a basic unit of information, which has the possible values 0 and 1. A basic unit of quantum information, on the other hand, is

called a quantum bit (or **qubit** in short) wit possible values $|0\rangle$ and $|1\rangle$, each of which can be described in the form of a vector.

For example

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2.8}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.9}$$

The state of a single qubit $|\psi\rangle$ can be described as follows.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \, (\alpha, \beta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 = 1). \tag{2.10}$$

After the operation called measurement, the quantum state would be collapsed into either 0 or 1. The measurement probability of 0 is $|\alpha|^2$ and that of 1 is $|\beta|^2$. In other words, a single qubit can take both states probabilistically at the same time.

For instance, a qubit can be

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{1}$$

whose measurement probability of 0 and 1 is 50% and 50% respectively.

Bloch Sphere

Because $|\alpha|^2 + |\beta|^2 = 1$, the notation of a single qubit state can be represented like this.

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2} + e^{i\phi}\sin\frac{\theta}{2})(\gamma, \phi, \theta \in \mathbb{R}) \tag{2.11}$$

.

Because $e^{i\gamma}$ is just a global phase, it can be ignored because it does not change the measurement probability of each state and therefore it does not cause any observable effect.

$$|\psi\rangle = \cos\frac{\theta}{2} + e^{i\phi}\sin\frac{\theta}{2}(\phi, \theta \in \mathbb{C}) \tag{2.12}$$

Because the equation above has two parameters, any pure single qubit state can be considered as a point on the surface of a sphere and its geometric representation is called the **Bloch sphere**.

Multi-Qubit State

The quantum state for multi-qubits is a **tensor product** of a state vector of each qubit. The general notation of a two qubit state is

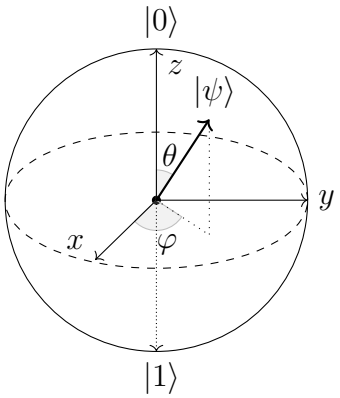

Figure 2.1: Bloch Sphere

$$\begin{aligned} |\psi\rangle &= (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle) \\ &= \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle \\ &(\alpha, \beta, \gamma, \delta \in \mathbb{C}, |\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1) \end{aligned} \tag{2.13}$$

For example, the state $|00\rangle$ is equal to

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.14}$$

.

However, some quantum states such as

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{2.15}$$

cannot be decomposed into independent quantum state of each qubit. These special quantum states are called **entangled** states.

Measurement

Quantum measurement can be described by using a group of measurement operators $\{M_m\}$ (m is the measurement result that is expected to get). It is worth noting the fact that a measurement operator is not unitary. If the quantum state before measurement is $|\psi\rangle$, the measurement probability of value m is

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle \tag{2.16}$$

The quantum state after the measurement is

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}} \tag{2.17}$$

The measurement operators satisfy the completeness equation

$$\sum_m M_m^\dagger M_m = I \tag{2.18}$$

Also, the sum of the measurement probability of each possible measurement outcome is equal to one.

$$\sum_m p(m) = \langle\psi| \sum_m M_m^\dagger M_m |\psi\rangle = 1 \tag{2.19}$$

2.2.3 Mixed State

A mixed state is another representation of quantum state in more general cases, such as the presense of physical error. A mixed state is described in the form of a matrix which is called a density matrix.

Definition

The density matrix of this system ρ is described by

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|. \tag{2.20}$$

Evolution

The quantum system after applying a unitary operator U is

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i p_i U|\psi_i\rangle\langle\psi_i|U^\dagger \tag{2.21}$$

It is worth noting that a unitary operation on a single qubit is equivalent to the rotation of a vector on the surface of the Bloch sphere.

Measurement

Suppose one performs measurement on a quantum state $|\psi_i\rangle$ using a measurement operator M_m.

Then, the measurement probability of m is

$$p(m|i) = \langle\psi_i|M_m^\dagger M_m|\psi_i\rangle = tr(M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|) \tag{2.22}$$

The measurement probability of m from the collection of states $\{|\psi_i\rangle\}$ is

$$\begin{aligned} p(m) &= \sum_i p_i p(m|i) \\ &= \sum_i p_i \langle\psi_i|M_m^\dagger M_m|\psi_i\rangle \\ &= \sum_i p_i \,\mathrm{tr}(M_m^\dagger M_m|\psi_i\rangle\langle\psi_i|) \\ &= \mathrm{tr}(M_m^\dagger M_m \rho). \end{aligned} \tag{2.23}$$

The quantum state after the measuring $|\psi_i\rangle$ is

$$|\psi_i^m\rangle = \frac{M_m|\psi_i^m\rangle}{\sqrt{\langle\psi_i^m|M_m^\dagger M_m|\psi_i^m\rangle}}. \tag{2.24}$$

The corresponding density matrix is

$$\rho_m = \sum_i p(i|m)|\psi_i^m\rangle\langle\psi_i^m| = \sum_i p(i|m)\frac{M_m|\psi_i\rangle\langle\psi_i|M_m^\dagger}{\sqrt{\langle\psi_i^m|M_m^\dagger M_m|\psi_i^m\rangle}} \tag{2.25}$$

$$\begin{aligned} p(i|m) &= \frac{p(m,i)}{p(m)} = \frac{p(m|i)p_i}{p(m)} \\ &= \frac{\mathrm{tr}(M_m^\dagger M_m\rho)p_i}{\mathrm{tr}(M_m^\dagger M_m\rho)} \\ &= p_i. \end{aligned} \tag{2.26}$$

Therefore, the state can also be described by the equation

$$\begin{aligned} \rho_m &= \sum_i p_i \frac{M_m|\psi_i\rangle\langle\psi_i|M_m^\dagger}{\mathrm{tr}(M_m^\dagger M_m\rho)} \\ &= \frac{M_m\rho M_m^\dagger}{\mathrm{tr}(M_m^\dagger M_m\rho)}. \end{aligned} \tag{2.27}$$

2.2.4 Fidelity

Fidelity is one of a distance measure between two quantum state. the fidelity of quantum state ρ and σ is

$$F(\rho,\sigma) = \mathrm{tr}\sqrt{\rho^{\frac{1}{2}}\sigma\rho^{\frac{1}{2}}} \tag{2.28}$$

The fidelity between these two states would be

$$\begin{aligned} F(\rho,\sigma) &= \mathrm{tr}\sqrt{\sum_i r_i s_i|i\rangle\langle i|} \\ &= \mathrm{tr}(\sum_i \sqrt{r_i s_i}|i\rangle\langle i|) \\ &= \sum_i \sqrt{r_i s_i} \end{aligned} \tag{2.29}$$

The fidelity between a pure state $|\psi\rangle$ and a mixed state ρ is

$$\begin{aligned} F(\psi,\rho) &= \mathrm{tr}\sqrt{\langle\psi|\rho|\psi\rangle|\psi\rangle\langle\psi|} \\ &= \sqrt{\langle\psi|\rho|\psi\rangle} \end{aligned} \tag{2.30}$$

2.3 Quantum Operations

Operators of quantum gates are unitary, while the one of the measurement operation is non-unitary. The definition of a unitary operator is described in 2.1.1

2.3.1 I Gate

The I gate is equal to the 2×2 identity matrix, which is

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.31}$$

.

For example,

$$I|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{2.32}$$

$$I|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{2.33}$$

.

2.3.2 X Gate

X gate

The X gate flips the logical value of a qubit.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.34}$$

.

For example,

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \tag{2.35}$$

$$X|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{2.36}$$

.

2.3.3 Y Gate

The Y gate flips the logical value of a qubit and adds an imaginary phase.

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{2.37}$$

.

For example,

$$Y|0\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ i \end{bmatrix} = i|1\rangle \tag{2.38}$$

$$Y|1\rangle = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -i \\ 0 \end{bmatrix} = -i|0\rangle \tag{2.39}$$

.

2.3.4 Z Gate

The Z gate flips the phase of $|1\rangle$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.40}$$

.

For example,

$$Z|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \tag{2.41}$$

$$Z|1\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -|1\rangle \tag{2.42}$$

.

2.3.5 H Gate

The Hadamard or, the H gate creates superposition.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.43}$$

.

For example,

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.44}$$

$$H|1\rangle = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.45}$$

.

2.3.6 Rotation Gate

Rotation gates are quantum gates that represent rotation through an arbitrary angle θ with respect to one of the x, y, z-axes of the Bloch sphere.

$$R_x(\theta) = e^{-iX\theta/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}X = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.46}$$

$$R_y(\theta) = e^{-iY\theta/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Y = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{2.47}$$

$$R_z(\theta) = e^{-iZ\theta/2} = \cos\frac{\theta}{2}I - i\sin\frac{\theta}{2}Z = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{2.48}$$

2.3.7 General One Qubit Gate

An arbitrary single qubit operation U can be decomposed into three rotation gates with an additional phase α.

$$U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta) = \begin{bmatrix} e^{i(\alpha-\beta/2-\gamma/2)\cos\frac{\delta}{2}} & e^{i(\alpha-\beta/2+\gamma/2)\sin\frac{\delta}{2}} \\ e^{i(\alpha+\beta/2-\gamma/2)\sin\frac{\delta}{2}} & e^{i(\alpha+\beta/2+\gamma/2)\cos\frac{\delta}{2}} \end{bmatrix} \tag{2.49}$$

2.3.8 Controlled-NOT Gate

A CNOT gate involves two qubits, one is called **the controll qubit** and the other is called **target qubit**. If the control qubit is 1, the bit value of the target qubit is flipped.

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.50}$$

.

For example,

$$CNOT_{0,1}|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle \tag{2.51}$$

$$CNOT_{0,1}|11\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle \tag{2.52}$$

.

2.3.9 Controlled-Z Gate

A CZ gate involves two qubits, one is called **the control qubit** and the other is called **target qubit**. If the control qubit is 1, the Z gate is applied to the target qubit.

$$CZ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.53}$$

.

For example,

$$CZ_{0,1}|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle \tag{2.54}$$

$$CZ_{0,1}|11\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -1 \end{bmatrix} = -|11\rangle \tag{2.55}$$

.

2.4 Quantum Circuit

Fig 2.4 is an example of a quantum circuit.

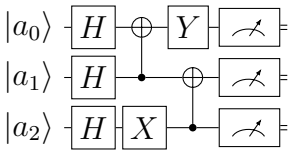

Figure 2.2: A example of a quantum circuit

Each horizontal line represents each qubit and the square boxes that contain letters mean single quantum gates. The sign which involves a vertical line means a CNOT gate, and the box on the most right side indicates measurement.

2.5 Quantum Entanglement

Quantum entanglement is a special type of quantum state that cannot be described in the form of tensor product of the state of each particle.

2.5.1 Bell Pair

The entangled states between two qubits are called Bell pairs or EPR pairs [20], and each of four states has a special notation.

$$|\Phi^+\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{2.56}$$

$$|\Phi^-\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}} \tag{2.57}$$

$$|\Psi^+\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}} \tag{2.58}$$

$$|\Psi^-\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}} \tag{2.59}$$

.

2.5.2 Multipartite Entanglement

There are cases that more than two qubits are entangled. One such state is called the Greenberger-Horne-Zeilinger state or GHZ state. The GHZ state is the only one of the infinite number of possible entangled states.

2.60 is the braket notation of the GHZ state that involves three qubits.

$$|GHZ\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}} \tag{2.60}$$

.

In the general case, the braket notation of the GHZ state of N qubits is the following.

$$|GHZ\rangle = \frac{|0\rangle^{\otimes N} + |1\rangle^{\otimes N}}{\sqrt{2}} \tag{2.61}$$

.

2.5.3 Graph State

A graph state is a particular multipartite entangled state. Given a particular graph $G(V, E)$, V indicates the collection of vertices and E indicates the collection of edges. The associated graph state $|G\rangle$ would be represented as

$$|G\rangle = \prod_{(a,b)\in E} CZ_{a,b}|+\rangle \tag{2.62}$$

.

First, all the qubits are initialized as $|+\rangle$ states and then Controlled-Z gates are applied to each pair of qubits that is represented by an edge. A graph state is also known as the initial state for measurement-based quantum computation (MBQC) [21].

2.5.4 Bell State Measurement

Bell state measurement is a special type of quantum measurement that determines which Bell pair the given two qubit entangled state is [22]. If the state is a superposition of Bell states, the system is projected into one of them probabilistically.

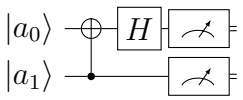

Figure 2.3: Quantum circuit for Bell state measurement

Measurement results	Bell state
00	$\|\Phi^+\rangle$
01	$\|\Phi^-\rangle$
10	$\|\Psi^+\rangle$
11	$\|\Psi^-\rangle$

Table 2.1: A table of correspondence between measurement result and Bell pair

2.5.5 Quantum Teleportation

Unlike classical communication, quantum states cannot be just copied and transmit to other nodes due to the no-cloning theorem, which forbids duplication of any quantum state.

However, a method called quantum teleportation was proposed [17], which overcomes the restriction and allows sender to transmit single qubit state to a distant location.

This method requires both the single qubit state and a new Bell pair. After applying a CNOT gate and an H gate between the single qubit and the first qubit of the Bell pair, the sender has to measure both qubits and send those measurement results over the classical network. Then, the receiver gets those measurement results and apply some quantum gates if the measurement results of corresponding qubits on the sender's side are 1, in order to correct on the quantum state on the receiver's side.

This protocol has been experimentally demonstrated over up to 1,400 km [23].

Fig 2.4 is the figure of the quantum circuit that performs quantum teleportation for a single qubit information [24].

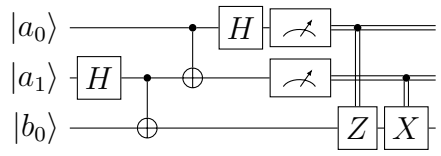

Figure 2.4: Quantum circuit for quantum teleportation

2.5.6 Entanglement Swapping

Entanglement swapping is the method to extend quantum entanglement by performing joint measurement on several quantum entanglement [15].

For example, assume Alice has a single qubit, Bob has two qubits, and Charlie has one qubit. Then, there are Bell pairs between Alice's qubit and Bob's first qubit, and Bob's second qubit and Charlie's qubit, respectively. If Bob performs Bell state measurement on both of his qubits, Alice's qubit and Charlie's qubit are eventually entangled, even though they have not interacted with each other. This can be also seen as the teleportation of a Bell pair by sending one of its particles.

This method has been experimentally demonstrated by using polarization photon qubits in free space [25], in optical fiber [26], and even using continuous variables [27].

2.5.6 is the figure of quantum circuit to perform entanglement swapping [28].

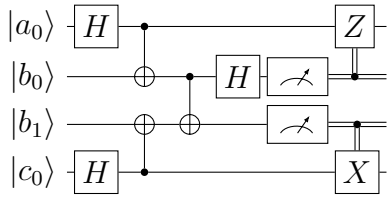

Figure 2.5: Quantum circuit for entanglement swapping

2.5.7 Entanglement Purification

Entanglement purification [16] is a scheme to generate a set of quantum entanglements with higher fidelities from a larger set of imperfect quantum entanglements, local quantum operations, and classical communications. This procedure is also called entanglement distillation, or quantum concatenation. This section presents an example of entanglement purification that generates a single Bell pair with higher fidelity from two of those with less fidelity.

This protocol has been experimentally demonstrated using photons [29], atoms [30] and electron-spin qubits [31].

Assume Alice and Bob are supposed to share $|\Phi^+\rangle$, which is one of the Bell pairs. However, the state would be converted to the following mixed state due to the noisy nature of a quantum channel.

$$\rho_{AB} = P_{\Phi^+}|\Phi^+\rangle\langle\Phi^+| + P_{\Phi^-}|\Phi^-\rangle\langle\Phi^-| + P_{\Psi^+}|\Psi^+\rangle\langle\Psi^+| + P_{\Psi^-}|\Psi^-\rangle\langle\Psi^-| \tag{2.63}$$

$$\sum_{s\in\{\Phi^+,\Phi^-,\Psi^+,\Psi^-\}} P_s = 1 \tag{2.64}$$

Any mixed state might be converted to a Werner state by applying Pauli operations and $\frac{\pi}{2}$ operations, so Alice and Bob can obtain the following state.

$$\rho'_{AB} = F|\Phi^+\rangle\langle\Phi^+| + \frac{1-F}{3}(|\Phi^-\rangle\langle\Phi^-| + |\Psi^+\rangle\langle\Psi^+| + |\Psi^-\rangle\langle\Psi^-|) \tag{2.65}$$

Fig 2.6 is the quantum circuit to prepare a Werner state.

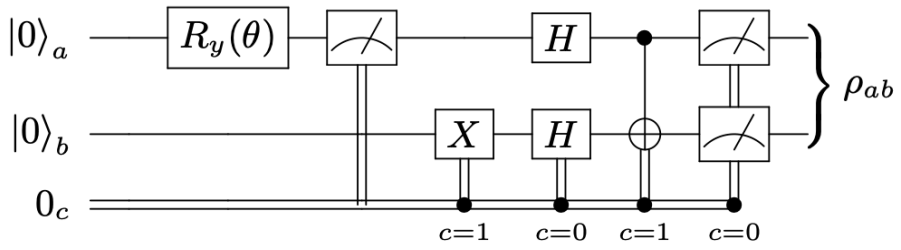

Figure 2.6: The figure of the circuit for entanglement purification [1]

Two noisy Bell pairs are required for entanglement purification. One of the Bell pair $\rho'_{a_1b_1}$ is called the source Bell pair, which may be purified, and the other one $\rho'_{a_2b_2}$ is the called target Bell pair, which is going to be measured. Then, Alice and Bob perform CNOT operations between a_1 and a_2, and b_1 and b_2, respectively. After that, they measure a_2 and b_2 respectively, which is the qubit on the target Bell pair on their side and exchange the measurement results. If their measurement results match, the purification is successful, while they have to discard the source Bell pair and try again if those results do not match.

The quantum state after measuring the target Bell pair, is

$$\rho'_{ab} = \frac{1}{N}\left[F^2 + \frac{1}{9}\left(1-F\right)^2\right]|\Phi^+\rangle\langle\Phi^+| + \frac{2F(1-F)}{3N}|\Phi^-\rangle\langle\Phi^-| + \frac{2(1-F)^2}{9N}(|\Psi^+\rangle\langle\Psi^+| + |\Psi^-\rangle\langle\Psi^-|)$$
$$(N = F^2 + \frac{2F(1-F)}{3} + \frac{2(1-F)^2}{9}) \quad (2.66)$$

The purification becomes successful if F $> \frac{1}{2}$. The readers can refer to more detailed calculation in the Appendix**??**

2.6 Quantum Networking

This section explains the important concepts of quantum networking.

2.6.1 Quantum Node

Quantum nodes are the nodes on a quantum network, which can be categorized into one of the following three categories, which was discussed in [32]

End nodes

MEAS measures the photons it receives. Although its functionality seems to be pretty limited, a pair of this type of node can perform quantum key distribution. In addition, a single node can be used as a terminal for blind quantum computation.

COMP represents quantum processor. This node also has the functionality of measuring qubits and also storing them in quantum memories.

SENS has sensing functionality by using quantum entanglement, which can be used for clock synchronization and a reference frame for interferometry.

Quantum repeaters and routers

REP1 plays the role of the 1st generation quantum repeater. It performs entanglement swapping and improves the fidelity of Bell pair by purification. The detail of each generation of quantum repeater network will be discussed in the later section.

REP2 plays the role of the 2nd generation quantum repeater. It performs entanglement swapping and perform quantum error correction on logical qubits composed of several physical qubits.

RTR behaves as the border between two different networks and also involves rewriting the given RuleSets into either 1st generation protocol and 2nd generation protocol based on what the network assumes.

Support nodes

EPPS, which stands for an entangled photon pair source, performs spontaneous parametric down conversion (SPDC). It creates pairs of entangled photons and send them to link end points. This node is used in terrestrial links or in satellite, which emits photons to telescopes on the ground.

BSA or Bell State Analyzer, generates a entangled state between two quantum memories by swapping two different entanglements between a single quantum memory and a single photon. The success probability of entanglement swapping with linear optics scheme does not exceed 50%.

RGSS generates multipartite photonic entangled state for memoryless quantum network. It sends each half of the generated repeater graph state to the neighboring nodes. The photons are measured at link end nodes.

ABSA performs both a single-photon measurement and two-photon measurements and their measurement basis changes based on previous measurement outcomes, logical encoding and the structure of repeater graph states.

OSW plays a role of optical switches and can exist independently or as a part of the type of nodes that are mentioned above. It switches photons from incoming links to outgoing ones.

Chapter 3

Problem Definition

3.1 Problem Definition

In order to maximize the overall performance and the aggregative use of resources in the entire network, it is desirable for several connections to be established in the real-time fashion. However, there are two major obstacles to overcome in the case of a quantum network.

One is the absence of a link management protocol for quantum network. There is previous work [33] that proposes and compares the performance of various multiplexing strategies, but it does not mention any concrete methods to establish multiple connections and allocate the available physical links to each of these connections.

The other one is the lack of interaction between connection management and the subsequent resource management. The current RuleSet-based communication protocol [18] only proposes the scheme to establish a single connection and it does not explain the method to tear it down and free the allocated physical links after the end of RuleSet execution.

This thesis tackles the first problem by proposing the link management protocol the involves the negotiation about the set of connections to establish and the one about when to start the establishment. It also discuss the messages and their properties that are required to run this protocol.

Additionally, this thesis explains how the link management scheme is going to be triggered when a new connection is established and the old one is torn down. This explanation includes the methods to implement in the relevant software components when RuleSet-based quantum network is simulated or deployed in the real world.

3.2 Assumptions

This protocol is proposed based on the following assumptions.

- The behavior of the entire network is determined by the collective decision made by each node.
- Each node allows the increase or decrease of the number of ongoing connections and does not refuse the setup of the new connection.

- The Bell pairs that are allocated but not used will be reallocated to one of the new connections.
- The link management protocol is triggered by the notification of either the setup of a new connection, or the teardown of an existing connection.
- The same amount of Bell pairs will be allocated to each RuleSet.

3.3 Requirements

This protocol has several requirements as follows.

3.3.1 Functional requirements

Common

- Two neighboring nodes MUST agree on the next set of RuleSets (which is called the link allocation policy) and its order.

Connection Setup

- Two neighboring nodes MUST also agree with when the new policy will be applied.
- If these negotiations are successful, available Bell pairs on the link level MUST be allocated to one of the RuleSets in the next link allocation policy.

Connection Teardown

- The execution of the old RuleSet MUST be terminated.
- Link Bell pairs that are allocated but not consumed MUST be released from the RuleSet after its connection is terminated.

Chapter 4

Proposal: Link Management For Quantum Network

4.1 Overview

This chapter proposes the protocol for the management of the link-level Bell pairs.

This protocol has two separated phases. The first phase is the negotiation about the set of RuleSets that are going to be established in the next round. The second phase is the negotiation about when to apply the next set of RuleSets. These negotiations will take place between the two end point of each link.

4.1.1 Link Allocation Policy

In order to establish multiple connections over a single link, both end nodes of the link need to make the coordinated decisions about what connections need to be established. This set of connections, to be more specific, the set of RuleSets, is called **Link Allocation Policy** in the rest of this thesis.

4.1.2 Link Allocation Policy Negotiation Phase

After the node receives a message that notifies the establishment of a new connection, or the termination of one of the existing connections, both nodes on a single link need to agree on the link allocation policy that is going to be executed in the next round. Therefore, this protocol involves the transmission of messages that include the information of the next link allocation policy in each node. It has to be mentioned that the order of arrival of RuleSets in the next policy might be different, so the protocol also requires a mechanism to determine which policy needs to be prioritized. This can be achieved by inserting a random integer to the message and adopting the order with the larger value.

4.1.3 The Timing Negotiation Phase

The end nodes of a physical link also need to align the timing of updating the link policy in order to assign the same Bell pair to the connection. Otherwise,they might

allocate the physical qubits of a different Bell pair to two different connections, which might end up with the failure of the entire connection.

4.1.4 Resource Management

The actual resource allocation process needs to take place before or during the execution of the RuleSets that were determined in the previous steps. In contrary, the release of physical resources that were allocated to the terminated RuleSets need to be executed after the notification of connection teardown, which the node receives from the networking layer.

4.2 Messages

This protocol involves the exchange of two kinds of messages, which are **LinkAllocationUpdateMessage** and **BarrierMessage**. This section proposes the required fields and their types in each message.

4.2.1 LinkAllocationUpdateMessage

This message contains the fields shown in table 4.1.

Field Name	Type	Explanation
srcAddress	integer	The source address
destAddress	integer	The destination address
activeLinkAllocations	unsigned long []	The array of IDs of the RuleSets in the active link allocation policy
nextLinkAllocations	unsigned long []	The array of IDs of the RuleSets in the upcoming link allocation policy
randomValue	integer	A random value

Table 4.1: The Message Fields in a LinkAllocationUpdateMessage

4.2.2 RejectLinkAllocationUpdateMessage

This message contains the fields shown in table 4.2.

4.2.3 BarrierMessage

This message contains the fields shown in table 4.3.

Field Name	Type	Explanation
srcAddress	integer	The source address
destAddress	integer	The destination address

Table 4.2: The Message Fields in a RejectLinkAllocationUpdateMessage

Field Name	Type	Explanation
srcAddress	integer	The source address
destAddress	integer	The destination address
sequenceNumber	integer	A sequence number of the first available physical Bell Pair

Table 4.3: The Message Fields in a BarrierMessage

4.3 Finite State Machine For Link Allocation Policy

A finite state machine (FSM) is commonly provides a simple and clear description about the behavior of the communication protocol [34]. Each state in the finite state machine represents the condition of a communication node, its events represents the change such as transmission and reception of messages, and the action represents the reaction to the event based on the previous condition. Each event triggers a state transition. This section explains the behavior of one of the end nodes of a link during the negotiation phase.

4.3.1 States

This subsection introduces all the states that a node can be during the negotiations related to the next link allocation policy.

Init

This is the initial state that each node starts with. In this state, neither the negotiation about the upcoming link allocation policy or the one about when to update the policy are happening. The FSM transits into either IncomingLAUWait state or LAUNotSent state by sending an LinkAllocationUpdateMessage or receiving the one from its neighboring nodes.

IncomingLAUWait

This is the state after a node sends a LinkAllocationUpdateMessage to its neighboring node. In this state, a node is waiting for the incoming LinkAllocationUpdateMessage from those nodes in return, so that the FSM can move to SyncNextPolicy state by coordinating the new link allocation policy.

LAUNotSent

This is the state when a node receives a LinkAllocationUpdateMessage from its neighboring node. In this state, a node is about to send LinkAllocationUpdateMessages back to those nodes, so that the FSM can move to SyncNextPolicy state by coordinating the new link allocation policy.

SyncNextPolicy

This is the state when both end nodes coordinated the next link allocation policy. The FSM transits into either BarrierNotSent state or IncomingBarrierWait state if the negotiation goes successfully, otherwise it transits back to Init if they fail.

IncomingBarrierWait

This is the state after a node sends a BarrierMessage to its neighboring node. In this state, a node is waiting for the incoming BarrierMessage from that node in return, so that the FSM can move to BarrierMessage state by coordinating from which Bell Pair the new link allocation policy should be applied.

BarrierNotSent

This is the state when a node receives a BarrierMessage from its neighboring node. In this state, a node is about to send BarrierMessage back to that node, so that the FSM can move to SyncNextSeqNum state by coordinating from which Bell Pair the new link allocation policy should be applied.

SyncNextSeqNum

This is the state when both end nodes of a link successfully coordinated from which Bell Pair the new link allocation policy will be updated. The FSM transits to the Init state until the next negotiation about the link allocation policy becomes triggered from the networking layer.

4.3.2 Events

This subsection introduces all the events that occur during the negotiation of the upcoming link allocation policy.

TxLAU

This event indicates the transmission of a LinkAllocationUpdateMessage.

RxLAU

This event indicates the reception of a LinkAllocationUpdateMessage.

TxBr

This event indicates the transmission of a BarrierMessage.

RxBr

This event indicates the reception of a BarrierMessage.

LAUSuccess

This event indicates the success in the coordination of the next link allocation policy.

LAUFail

This event indicates the failure in the coordination of the next link allocation policy.

BarrierSuccess

This event indicates the success in the coordination of the first sequence number that the new link allocation policy is applied.

BarrierFail

This event indicates the failure in the coordination of the first sequence number that the new link allocation policy is applied.

4.3.3 Description of Finite State Machine

Fig 4.1 shows the finite state machine that describes the change of states during the negotiations related to the next link allocation policy.

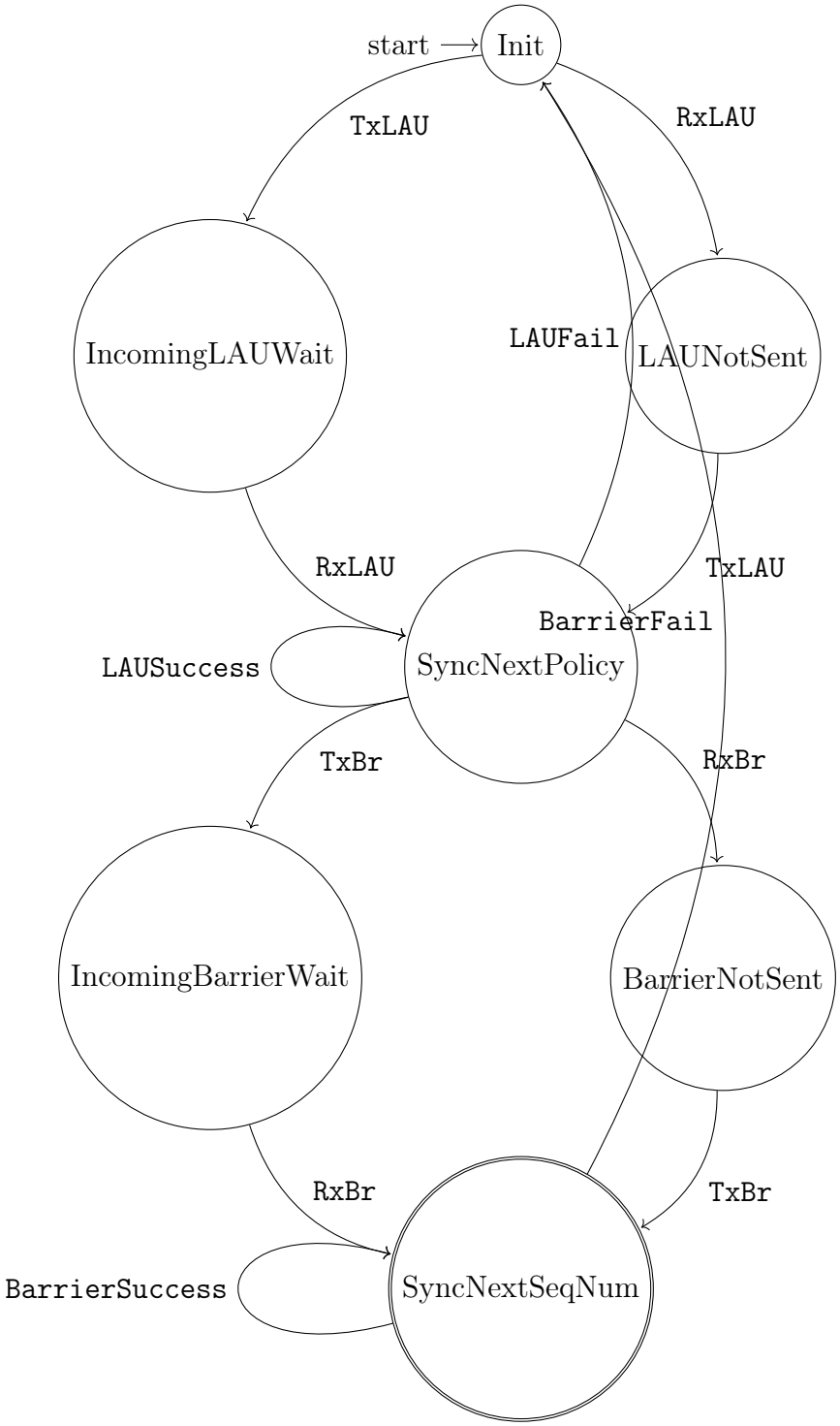

Figure 4.1: The FSM for the negotiation phase

4.4 Finite State Machine For Link Management

This section provides the different finite state machine for the link management phase. It focuses on the behavior of a link-level Bell pair for simplicity.

4.4.1 States

This subsection introduces all the states that a link-level Bell pair can be during the resource management.

Up

This is the state when a link-level Bell pair between its two end nodes. In this state, it is not allocated to any specific RuleSet. The FSM transits to Allocated if the link becomes allocated to one of the RuleSets in the active link allocation policy.

Down

This is the state when a physical Bell pair on a quantum link is not established between its two end nodes. In this state, it can be no Bell pair between two existing quantum memories in the case of a quantum repeaters with those memories, or the situation when incoming photons have not arrived to memoryless quantum repeaters. The FSM transits to Up if a Bell pair is established.

Allocated

This is the state when a physical Bell pair is allocated to one of the RuleSets in the active link allocation policy. The FSM transits to Up if the link becomes released after the connection that this link used to be allocated is terminated. It can also transits to Down if the physical qubits on the link are measured during execution of the RuleSet that this link is allocated to.

4.4.2 Events

This subsection introduces all the events that cause transition of states.

BellPairGen

This event indicates the generation of a Bell pair

Allocate

This event indicates the allocation of a given Bell pair to RuleSet.

Free

This event indicates the release of an available Bell pair from that RuleSet that it is used to be allocated to.

Measure

This event indicates the measurement of two physical qubits of the given Bell pair while the RuleSet is being executed.

4.4.3 Description of Finite State Machine

Fig 4.2 is the state machine that describes the transition for the state of a single link-level Bell pair.

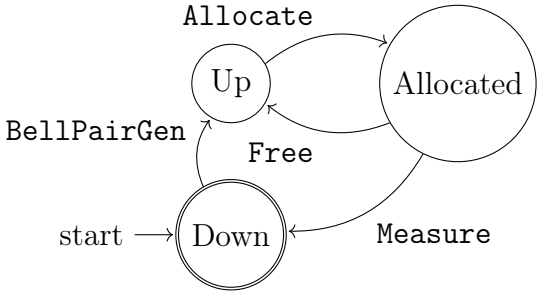

Figure 4.2: The FSM for the resource management phase

Chapter 5

Simulation

5.1 QuISP (Quantum Internet Simulation Package)

5.1.1 Overview

QuISP [35] is a quantum network simulator which aims to simulate the behavior of a large-scale quantum network. It is built on top of OMNeT++ [36], which is an event-driven network simulator. QuISP is built on top of OMNet++ because OMNet++ allows users to define their own networking layers. QuISP can simulate various types of errors, not only Pauli X error, Pauli Y error and Pauli Z error, but also relaxation error and excitation error. Physical noise on an actual quantum system with n qubits are usually simulated in the form of a density matrix, which would includes $2^n \times 2^n$ elements and soon becomes intractable as n becomes larger. QuISP realizes the scalable simulation of quantum network by simulating the physical error using an error probability vector, which would take the following form.

$$\overrightarrow{\pi}(t) = (\pi_I, \pi_X, \pi_Y, \pi_Z, \pi_R, \pi_E, \pi_L) \tag{5.1}$$

It contains $m+1$ elements (m is the number of simulated error types) The time evolution of error probability vector is provided by a transition error matrix Q.

$$\overrightarrow{\pi}(t) = \overrightarrow{\pi}(t-1)Q \tag{5.2}$$

The error probability vector above is the one for a single qubit, so the one for N qubit system contains $N(m+1)$ elements.

5.1.2 Hardware Components

Communication between two quantum nodes is achieved by transmission of photons via an optical fiber, and the fiber is mocked by an object called quantum link. QuISP supports three main link architecture.

Memory-Memory

The first one is Memory-Memory that two nodes are directly connected via a quantum link and the Bell State Analyzer is equipped in the receiver node.

Memory-Interference-Memory

The second one is Memory-Interface-Memory. Both end nodes of a quantum link emits photons to Bell State Analyzer located in the middle. After they become entangled, all the measurements results and required operations are sent back to both nodes.

Memory-Source-Memory

The last one Memory-Source-Memory. All the entanglement pairs are both generated and sent from the source of entangled photonic pair states in the middle.

5.1.3 Software Components

ConnectionManager

Connection establishment is done when connection manager at the Initiator nodes sends ConnectionSetupRequest to the Responder node and intermediate nodes sends additional information such as those about QNIC. After that, the connection manager at the responder node sends ConnectionSetupResponse to each node along the path of the connection.

HardwareMonitor

HardwareMonitor is the module that collects the information of a quantum link such as fidelity and generation rate and pass those information to the routing daemon and the connection manager.

BellPairStore

BellPairStore is the module that stores the entanglement pairs generated from a support node such as a Bell State Analyzer.

Runtime

Runtime is the program that executes each RuleSet.

RuntimeManager

RuntimeManager is the program that store Runtime for each RuleSet, which is a part of Rule Engine, which is explained in the next section.

RuleEngine

RuleEngine is the component that is in charge for executing the given RuleSets and monitor the conditions of physical qubits.

5.2 Implementation

This section discusses the mechanism and required methods that the author implemented in order to realize link management in QuISP.

5.2.1 Link Management in QuISP

Link Allocation Policy Negotiation

The first phase of link management in RuleSet-based quantum networking is the negotiation of the next set of RuleSets to execute. This is triggered by each update in the set of available runtimes, or RuleSets in the Rule Engine, such as the reception of a new RuleSet or the notification of connection teardown, which will be explained in detail in the later section.

First of all, LinkAllocationUpdateMessages are sent to all the neighboring nodes. 5.2.1 is the pseudocode of the transmission of those messages.

Algorithm 1 Algorithm For Sending LinkAllocationUpdateMessages

Require: The address of this node parent_address
Require: An array of available runtimes runtimes
Require: A map of node address of each neighboring nodes and the active link allocation policy node_address_active_link_allocations_map
Require: A map of node address of each neighboring nodes and the next link allocation policy node_address_next_link_allocations_map
Require: A map of node address of each neighboring nodes and a random number node_address_random_number_map
Require: A map of node address of each neighboring nodes and whether a LinkAllocationUpdateMessage is sent from this node node_address_lau_sent_map

```
1: function SendLinkAllocationUpdateMessages
2:     Initialize an array of node addresses of neighboring nodes partner_addresses
3:     for all runtime ← runtimes do
4:         for all partner ← the list of all the neighboring addresses that runtime needs to communicate do
5:             Append partner to partner_addresses if it has not been added
6:         end for
7:     end for
8:     for all runtime ← runtimes do
9:         for all partner ← the list of all the neighboring addresses that runtime needs to communicate do
10:            Initialize an array for RuleSet IDs in the active link policy active_link_allocations
11:            if partner is a part of the key of node_address_active_link_allocations_map then
12:                for all active_link_allocation ← node_address_active_link_allocations_map[partner] do
13:                    Append active_link_allocation to active_link_allocations
14:                end for
15:            end if
```

```
16:             Initialize an array for RuleSet IDs in the next link policy
    next_link_allocations
17:             if partner is a part of the key of
    node_address_next_link_allocations_map then
18:                 for all next_link_allocation ←
    node_address_next_link_allocations_map[partner] do
19:                     Append next_link_allocation to
    next_link_allocations
20:                 end for
21:             end if
22:             for all active_link_allocation ←
    active_link_allocations do
23:                 Append active_link_allocation to
    node_address_active_link_allocations_map[partner]
24:             end for
25:             for all next_link_allocation ← next_link_allocations
    do
26:                 Append next_link_allocation to
    node_address_next_link_allocations_map[partner]
27:             end for
28:         end for
29:         for all partner_address ← partner_addresses do
30:             Initialize an new LinkAllocationUpdateMessage pkt
31:             The source address of pkt ← parent_address
32:             The target address of pkt ← partner_address
33:             for all active_link_allocation ←
    node_address_active_link_allocations_map[partner] do
34:                 Append active_link_allocation to activeLinkAlloca-
    tions of pkt
35:             end for
36:             for all next_link_allocation ←
    node_address_next_link_allocations_map[partner] do
37:                 Append next_link_allocation to nextLinkAllocations
    of pkt
38:             end for
39:             Initialize a random integer rand_number
40:             The random number of pkt ← rand_number
41:             node_address_random_number_map[partner_address] ←
    rand_number
42:             node_address_lau_sent_map[partner_address] ← True
43:             Send pkt
44:         end for
45:     end for
46: end function
```

If the RuleEngine receives an incoming LinkAllocationUpdateMessage, it has to store the information for the further negotiation.

Algorithm 2 Algorithm For Storing the Information of an Incoming LinkAllocationUpdateMessage

Require: An incoming LinkAllocationUpdateMessage pkt
Require: A map of node address of each neighboring nodes and the incoming random number node_address_incoming_random_number_map
Require: A map of node address of each neighboring nodes and the incoming active link allocation policy node_address_incoming_active_link_allocations_map
Require: A map of node address of each neighboring nodes and the incoming next link allocation policy node_address_incoming_next_link_allocations_map
Require: A map of node address of each neighboring nodes and whether the RuleSet received LinkAllocationUpdateMessage node_address_lau_received_map

```
1: src_addr ← the source address of pkt
2: random_number ← the random number of pkt
3: node_address_incoming_random_number_map[src_addr] ← random_number
4: incoming_active_link_allocations_count ← the number of RuleSets in the active link allocation policy
5: for i ← 0 to incoming_active_link_allocations_count do
6:     incoming_active_link_allocation ← ith element of the ActiveLinkAllocations of pkt
7:     Append incoming_active_link_allocation to node_address_incoming_active_link_allocations_map[src_addr]
8: end for
9: incoming_next_link_allocations_count ← the number of RuleSets in the next link allocation policy
10: for i ← 0 to incoming_next_link_allocations_count do
11:     incoming_next_link_allocation ← ith element of the NextLinkAllocations of pkt
12:     Append incoming_next_link_allocation to node_address_incoming_next_link_allocations_map[src_addr]
13: end for
14: node_address_lau_received_map[src_addr] ← True
```

If both the incoming and outgoing LinkAllocationUpdateMessages have the same random numbers, the negotiation will be rejected by sending a RejectLinkAllocationUpdateMessage to each other.

Algorithm 3 Algorithm For Sending a RejectLinkAllocationUpdateMessage

Require: The address of the this node this_address
Require: The address of the destination node dest_address
1: Initialize a new RejectLinkAllocationUpdateMessage pkt
2: The source address of pkt ← this_address
3: The destination address of pkt ← dest_address
4: Send pkt

After exchanging LinkAllocationUpdateMessages, each pair of nodes need to synchronize the contents and their order of the next link allocation policy. The policy from the LinkAllocationUpdateMessage with the bigger random value will be prioritized.

Algorithm 4 Algorithm For Synchronizing the Link Allocation Policy

Require: The address of the destination node dest_address
Require: A map of node address of each neighboring nodes and the incoming random number node_address_incoming_random_number_map
Require: A map of node address of each neighboring nodes and the random number in this node node_address_random_number_map
Require: A map of node address of each neighboring nodes and whether the RuleSet sent BarrierMessage node_address_barrier_sent_map
Require: A map of node address of each neighboring nodes and whether the RuleSet received BarrierMessage node_address_barrier_received_map
1: incoming_random_number ← node_address_incoming_random_number_map[dest_address]
2: random_number ← node_address_random_number_map[dest_address]
3: **if** incoming_random_number == random_number **then**
4: Send a RejectLinkAllocationUpdateMessage
5: **else if** incoming_random_number > random_number **then**
6: node_address_incoming_random_number_map[dest_address] ← node_address_random_number_map[dest_address]
7: node_address_barrier_sent_map[dest_address] ← False
8: node_address_barrier_received_map[dest_address] ← False
9: **end if**

Link Allocation Timing Negotiation

After the next link allocation policy is synchronized between two neighboring nodes, it is time to determine when to update the link allocation policy.

Algorithm 5 Algorithm For Sending a BarrierMessage

Require: The address of the this node this_address
Require: The address of the destination node dest_address
1: Initialize a new BarrierRequest pkt
2: The source address of pkt ← this_address
3: The destination address of pkt ← dest_address
4: sequence_number ← the smallest value among sequence numbers of the available link Bell pairs
5: The sequence number of pkt ← sequence_number
6: Send pkt

Algorithm 6 Algorithm For Storing Information About the Incoming BarrierMessage

Require: The incoming BarrierMessage pkt
Require: A map of node address of each neighboring nodes and the smallest sequence number in this node node_address_barrier_sequence_number_map
Require: A map of node address of each neighboring nodes and whether the RuleSet received BarrierMessage node_address_barrier_received_map
1: sequence_number ← the smallest value among sequence numbers of the available link Bell pairs
2: node_address_incoming_sequence_number_map[src_addr] ← sequence_number

Algorithm 7 Algorithm For Synchronizing the Next Sequence Number

Require: The node address of one of the neighboring nodes src_address
Require: A map of node address of each neighboring nodes and the smallest sequence number in this node node_address_sequence_number_map
Require: A map of node address of each neighboring nodes and the incoming smallest sequence number node_address_incoming_sequence_number_map
1: sequence_number ← node_address_sequence_number_map[src_address]
2: incoming_sequence_number ← node_address_incoming_sequence_number_map[src_address]
3: **if** incoming_sequence_number > sequence_number **then**
4: node_address_sequence_number_map[src_addr] ← incoming_sequence_number
5: **end if**

Link Management in QuISP

Allocation of link Bell pairs starts after two neighboring nodes coordinates the first sequence number to start applying the new link allocation policy. Release of link Bell pairs that were allocated to the terminated connection will be performed after the termination of execution of the associated RuleSet.

Algorithm 8 Algorithm For Allocating Link Bell pairs

Require: The QNIC type qnic_type
Require: The QNIC index qnic_index
Require: The first sequence number first_sequence_number

Initialize a map of node addresses of neighboring nodes and the indices of Runtimes partner_addr_runtime_indices_map
index ← 0
for all runtime ← runtimes **do**
    partners ← the addresses of all nodes that runtime needs to communicate
    for all partner ← partners **do**
        Append partner to partner_addr_runtime_indices_map[partner]
    end for
    index ← index + 1
end for
for all [partner_addr, value] ← partner_addr_runtime_indices_map **do**
    runtime_indices ← partner_addr_runtime_indices_map[partner_addr]
    bell_pair_range ← the list of Bell pairs on the side of this node
    bell_pair_num ← 0
    for all bell_pair ← bell_pair_range **do**
        bell_pair_num ← bell_pair_num + 1
    end for
    Initialize a map of Runtime indices and the number of Bell pairs runtime_index_bell_pair_number_map
    number ← 0
    for all bell_pair ← bell_pair_range **do**
        sequence_number ← sequence_number of bell_pair
        if first_sequence_number <= sequence_number **then**
            qubit_record ← The object of the physical qubit in bell_pair
            if qubit_record is not allocated **then**
                qubit_record.is_allocated ← True
                index ← ⌊number × the size of runtime_indices / bell_pair_num⌋
                i ← runtime_indices[index]
                Allocate qubit_record to the ith Runtime
                if i is a part of the key list of runtime_index_bell_pair_number_map **then**
                    runtime_index_bell_pair_number_map[runtime_index] ← runtime_index_bell_pair_number_map[runtime_index] + 1
                else
                    runtime_index_bell_pair_number_map[runtime_index] ← 0
                end if

```
            end if
        end if
        number ← number + 1
    end for
end for
```

Algorithm 9 Algorithm For Releasing Link Bell pairs

Require: The QNIC type qnic_type
Require: The QNIC index qnic_index
Require: The first sequence number first_sequence_number
Require: A map of sequence numbers of Bell pairs and associated RuleSet IDs sequence_number_ruleset_id_map

```
Initialize a map of node addresses of neighboring nodes and the indices of terminated Runtimes partner_addr_terminated_runtime_indices_map
index ← 0
terminated_runtimes ← the array of terminated runtime from RuntimeManager
for all terminated_runtime ← terminated_runtimes do
    partners ← the addresses of all nodes that terminated_runtime needs to communicate
    for all partner ← partners do
        Append partner to partner_addr_terminated_runtime_indices_map[partner]
    end for
    index ← index + 1
end for
for all [partner_addr, value] ← partner_addr_terminated_runtime_indices_map do
    runtime_indices ← partner_addr_terminated_runtime_indices_map[partner_addr]
    bell_pair_range ← the list of Bell pairs on the side of this node
    for all bell_pair ← bell_pair_range do
        sequence_number ← sequence_number of bell_pair
        qubit_record ← The object of the physical qubit in bell_pair
        ruleset_id ← sequence_number_ruleset_id_map[sequence_number]
        if qubit_record is already allocated then
            qubit_record.is_allocated ← False
            Reinitialize the qubit record
            if qubit_record is already allocated then
                if qubit_record is busy then
                    qubit_record.is_busy ← False
                end if
```

```
25:                 for all terminated_runtime ← terminated_runtimes
    do
26:                     if ID of terminated_runtime is ruleset_id then
27:                         Release qubit_record
28:                         runtime_index                                    ←
    sequence_number_runtime_index_map[sequence_number]
29:                         runtime_index_bell_pair_number_map[runtime_index] ←
    runtime_index_bell_pair_number_map[runtime_index] - 1;
30:                     end if
31:                 end for
32:             end if
33:         end if
34:     end for
35: end for
```

5.2.2 Relationship with Connection Management

Connection Setup

There are three steps for connection setup in RuleSet-based quantum networking. The first one is the transmission of a ConnectionSetupRequest from the Initiator to the Responder.

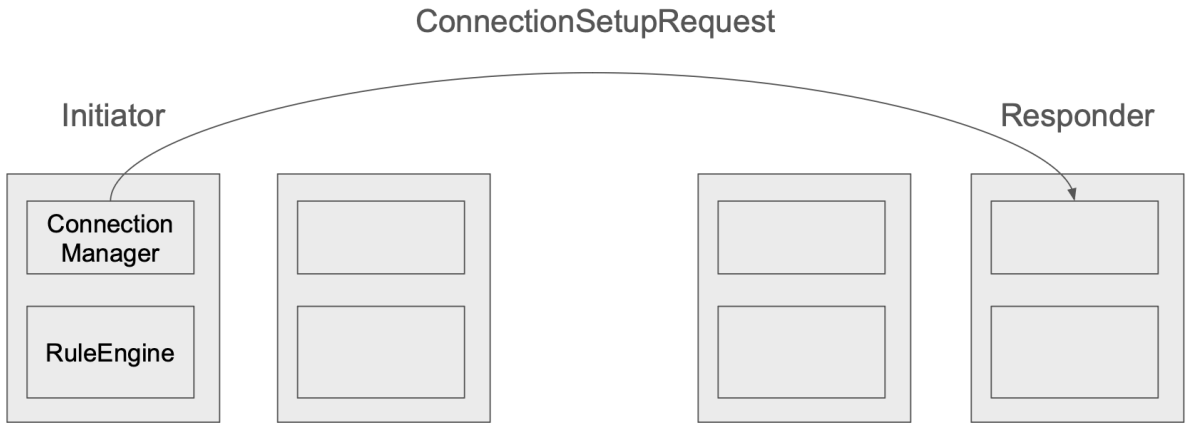

Figure 5.1: Transmission of a ConnectionSetupRequest

After the responder receives ConnectionSetupRequest, it sends Ack and RuleSet to each node along the path of the connection.

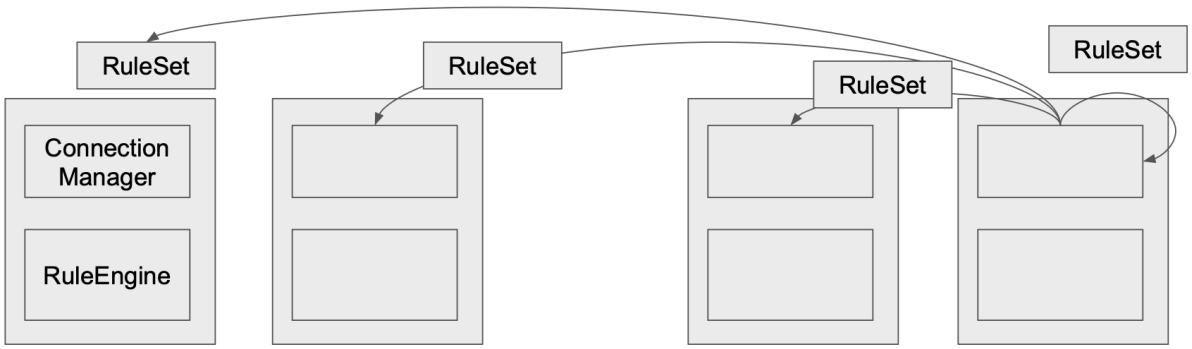

Figure 5.2: Transmission of RuleSets

The RuleSet received by the ConnectionManager will be stored in RuleEngine in each node.

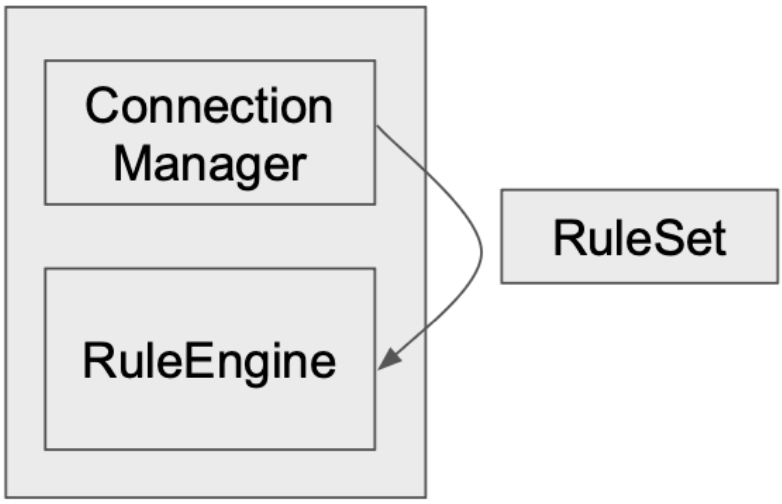

Figure 5.3: Storing RuleSet in the RuleEngine

The reception of RuleSet triggers the transmission of LinkAllocationUpdateMessage to each of the neighboring nodes. After each node both sent and received LinkAllocationUpdateMessages, it determines the next link allocation policy with each of the neighboring nodes.

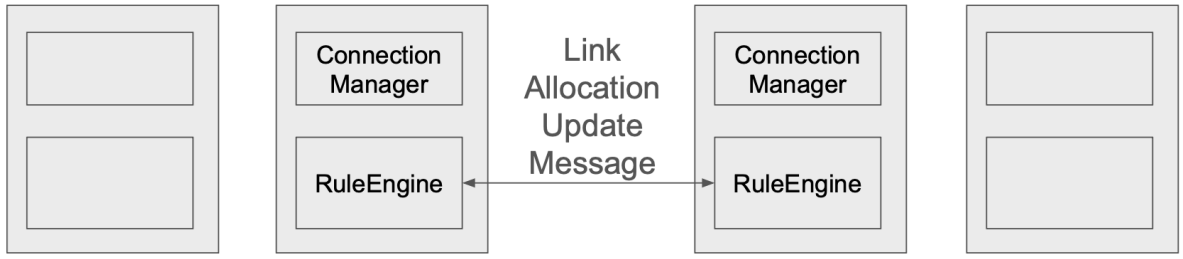

Figure 5.4: Exchange of LinkAllocationUpdateMessages

Generation of link Bell pairs are performed independently from exchange of messages.

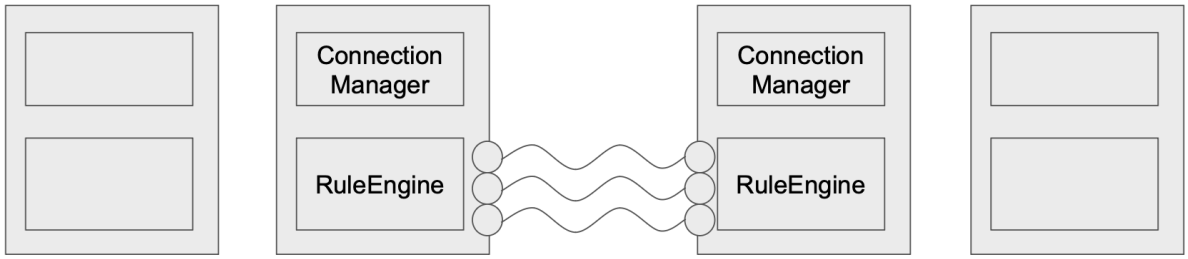

Figure 5.5: Generation of Link-level Bell pairs

BarrierMessages will be exchanged only if the next link allocation policy is determined and available link Bell pairs exist with each of the neighboring nodes. After each node sent and receives BarrierMessages, the first sequence number for the next link allocation is determined.

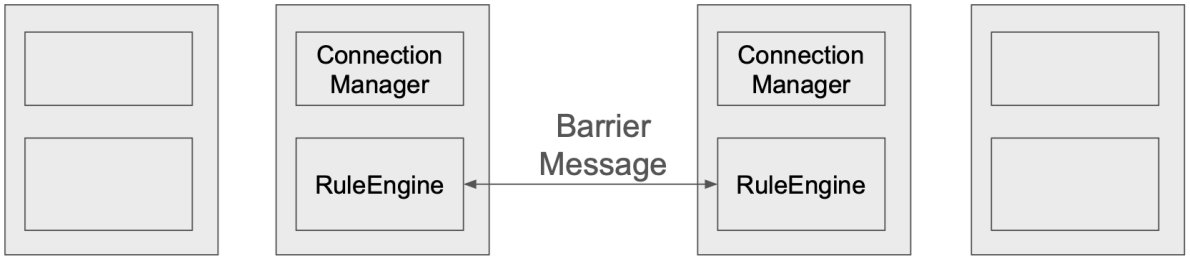

Figure 5.6: Exchange of BarrierMessages

the first sequence number for the next link allocation is negotiated, each link Bell pairs will be allocated to one of the runtimes for the RuleSets in the next link allocation policy.

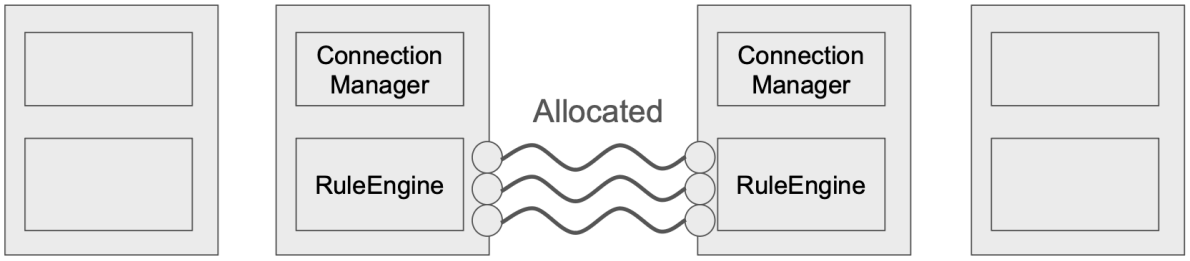

Figure 5.7: Allocation of Link Bell pairs

Connection Teardown

After the RuleEngine detects the termination of execution of the RuleSet, the responder sends a ConnectionTeardownNotifier to its ConnectionManager.

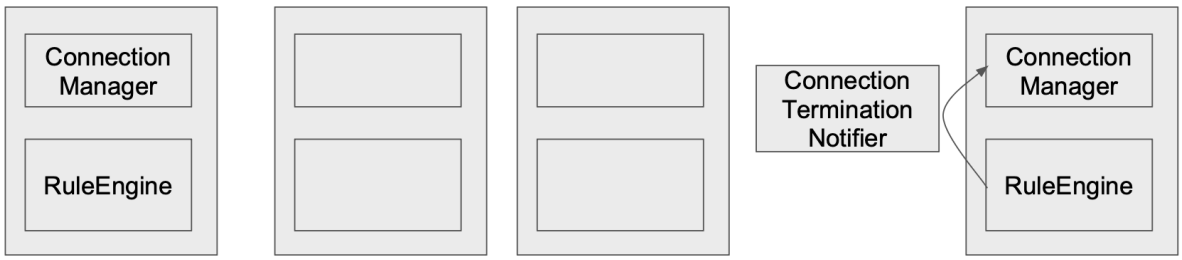

Figure 5.8: Transmission of ConnectionTeardownNotifier

After the ConnectionManager receives a ConnectionTeardownNotifier, it sends a ConnectionTeardownMessage to each node along the path of the connection.

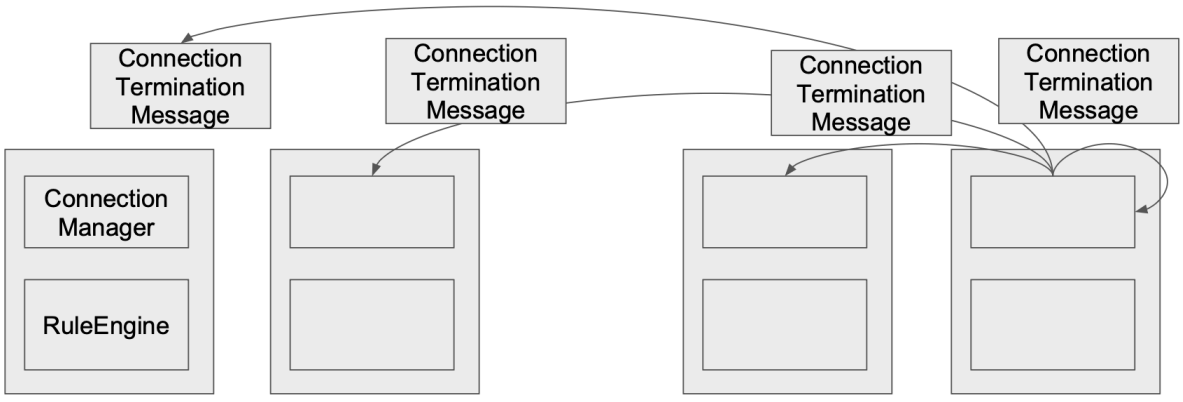

Figure 5.9: Transmission of ConnectionTeardownMessages

After each node receives a ConnectionTeardownMessage, it stores the message to the RuleEngine in the same node. The name of the message is changed to an InternalConnectionTeardownMessage for a better readability.

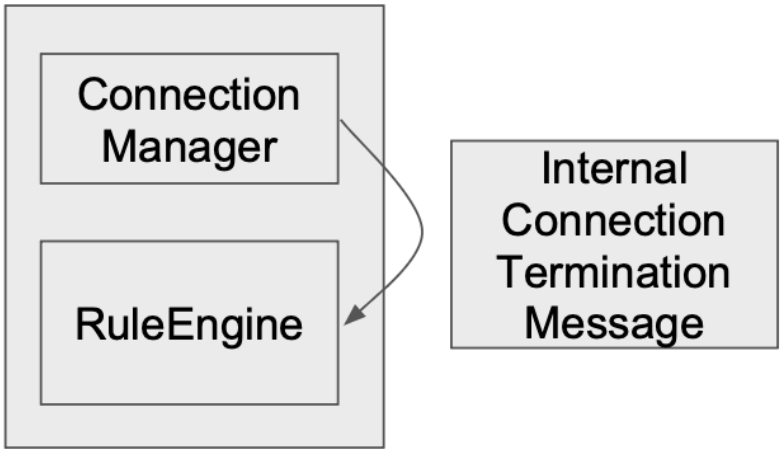

Figure 5.10: Storing a ConnectionTeardownMessage

After the RuleEngine receives an InternalConnectionTeardownMessage, it terminated the execution of the RuleSet for the connection that was torn down.

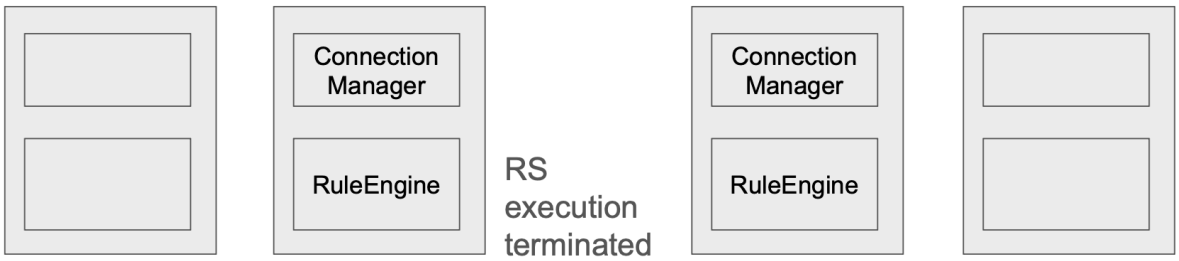

Figure 5.11: Termination of the execution of RuleSets

Lastly, all the link-level Bell pairs that are allocated but not consumed are released so that they can be reused for new RuleSets.

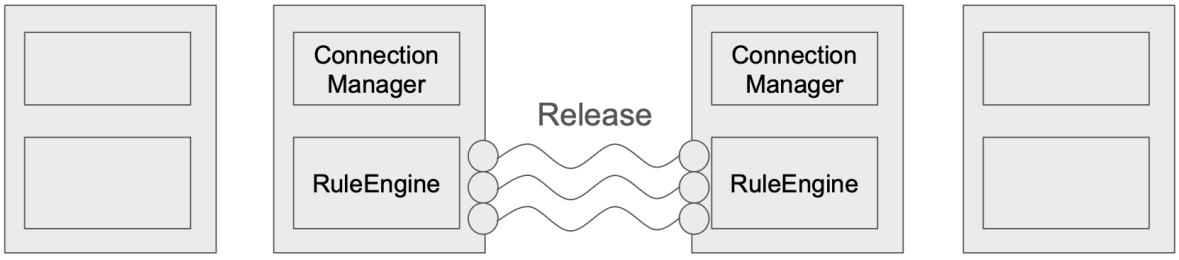

Figure 5.12: Release of allocated Link Bell pairs

Chapter 6

Related Works

6.1 Protocol Stack For Quantum Network

This section discusses the protocol stack for quantum network. The protocol stack is a collection protocols that supports various levels of communication. Table 6.1 is the comparison of the protocol stack for quantum network that are presented in the previous works.

Name	Functionality
Application	Run an application on an E2E connection
Purification Control	Perform purification to E2E connection
Entanglement Swapping Control	Perform entanglement swapping to establish an E2E connection
Purification Control	Perform purification to a physical bell pair
Entanglement Control	Provide robustness to the bell pair establishment
Physical Entanglement	Establish a physical bell pair

Table 6.1: Protocol Stack for Quantum Network in [2]

The work [2] is the first study that proposed the quantum protocol stack and its proposal assumes the quantum repeater protocol that manages error using entanglement purification for both a link between two neighboring nodes and an end-to-end connection between two end nodes. It has to be mentioned that this work assumes the number of hops for entanglement swapping and purification is assumed to be $N = 2^n$ (n is a positive integer) in a linear topology. Also, this work does not assume the routing functionality in any protocol layer.

Another work [3] proposes the different stack of quantum networking protocols that assumes the existence of transport layer that teleports a qubit using an end-to-end connection that is established by up to the network layer. Also, it mentions the future outlook that the functionalites of routing and entanglement management may be separated from the network layer.

Name	Functionality
Application	Run an application on an E2E connection
Transport	Qubit transmission
Network	Long distance entanglement
Link	Robust entanglement generation
Physical	Attempt entanglement generation

Table 6.2: Protocol Stack for Quantum Network in [3]

Although several previous works present different protocol stacks in terms of those in the upper layer, those protocol stacks still have some common features, which are

- Establishment of an actual physical Bell pair
- Robust entanglement generation
- Extension of physical bell pairs in order to establish an end-to-end Bell pair

These three elements will be the foundation of quantum network. This thesis will introduce specific protocols in the physical layer that is responsible for establishing the physical bell pair and link layer that adds robustness in the process of entanglement generation.

6.2 Physical Layer Protocols For Quantum Network

A previous work [3] proposes the communication protocol for the physical layer for two of the four different use cases of a quantum network that it defines. The protocol is called the midpoint heralding protocol (MHP) in short.

MHP for Create and Keep (CK)

Create and Keep is the use case when multiple entanglements should be stored simultaneously, such as quantum sensing [37], metrology [38] and distributed quantum computing [39]. The process of entanglement generation is triggered by the reception of the message from the link layer, which includes the following parameters.

- An ID for the entanglement generation attempt
- Generation parameters
- Qubits on the physical device which entanglements will be stored.
- The detail of microwave and laser pulse sequence

Then, the GEN message, which asks for the entanglement generation with the ID in the given message and the timestamp is sent to the support node in the middle. The support node uses the given timestamp to see if it receives the same IDs from the both

side within a certain amount of time. Also, it sends a REPLY message which includes the result, which is either success or failure, the generated state, and a sequence of IDs of entangled qubits after the measurement. Then the end node performs an additional gate operation on the physical qubit depending on what state is generated, and redirected the received information to the link layer.

MHP for Measure Directly (MD)

Measure Directly is the usecase when multiple entanglements need to be created sequentially such as quantum key distribution [7] and secure identification [40]. The basic procedure is the same as the one above, but there are two main differences. One is the operations that the end nodes perform on qubits. Instead of performing additional state, it performs measurement on a specific basis. The othe one is the timing of measurement. These nodes perform these measurement only they receive successful responses.

6.3 Link Layer Protocols For Quantum Network

Communication protocols for link layer have been proposed in several previous studies [2, 3, 4]. The biggest difference between the communication protocol for the physical layer protocol and the one for the link layer is reliability. The former involves the process of actual entanglement generation and classical communication that triggers the process. On the other hand, the latter requires the additional classical communication that indicates the beginning and end of entanglement generation.

The first protocol [2] assumes that end nodes of a physical link are directly connected via an optical fiber. First of all, multiplexed optical pulses are sent to the receiver and they are demultiplexed and measured at the receiver. After several entanglements are generated, the ACK or NACK "keep" flags for each physical qubit are sent back to the sender.

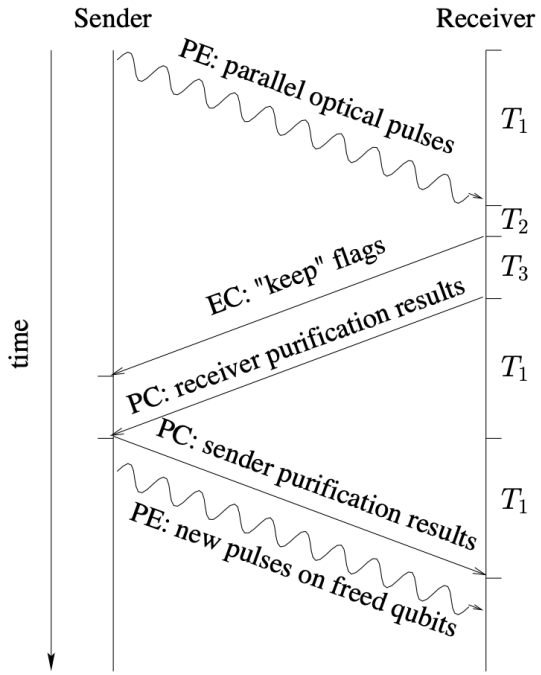

Figure 6.1: Message sequences in the link layer protocol in [2]

The next protocol [3] assumes several components, which are Distributed Queue to store requests, Quantum Memory Management (QMM) to decide which physical qubits to use, Fidelity Estimation Unit (FEU) to estimate hardware capabilities, and Scheduler that schedules the timing of incoming requests. The link layer receives a CREATE operation from the upper networking layer with the number of entangled pairs it requires, the minimum required fidelity, and the amount of time it can wait. After that, the FEU estimates the hardware capabilities and the amount of time it would take to generate the entanglement pairs. It the request will be rejected if the estimated time exceeds the given amount of time. If it is accepted, the link layer send the "yes" response with the unique ID and the number of requested entanglement pairs.

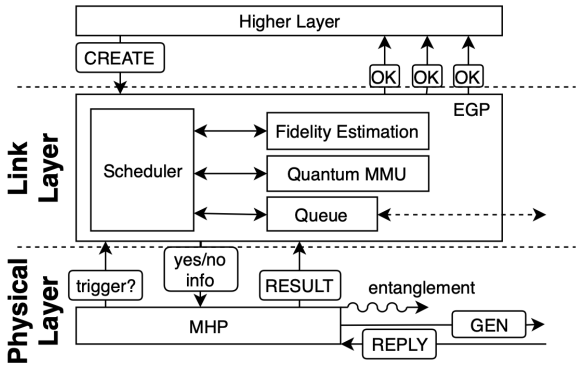

Figure 6.2: Message sequences in the link layer protocol in [3]

The last protocol [4] assumes either SendReceiver or MeetInTheMiddle for its link architecture. First of all, each end node sends Boot Up Notification to its neighboring BSA nodes. After these notifications are received, the BSA node in the middle calculates the emission timing and send it back to its neighboring end nodes. Then end nodes emit a bulk of photons and send the message to notify the end of photon emission. Lastly, the BSA node transmits the measurement results either sequentially or in batch and the message that includes the next emission timing.

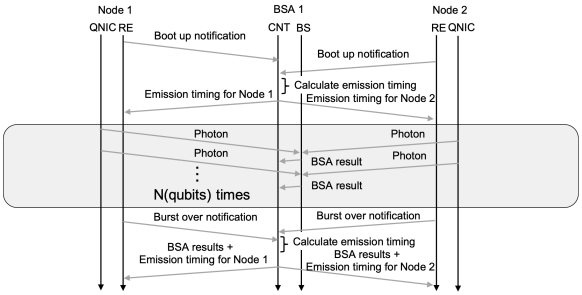

Figure 6.3: Message sequences in the link layer protocol in [4]

6.4 RuleSet-Based Quantum Network

This section explains the essential features of RuleSet-based quantum networking. Before the connection is established, the initiator sends the ConnectionSetupRequest and the information about each link along the path to the responder. After the responder receives the request, it sends the ConnectionSetupResponse and an object called RuleSet. RuleSet is a collection of Rule, which contains both one or more Condition clauses and Action clause. Condition clause is the condition to meet in order to perform a specific operation, and an Action clause is the operation itself, such as entanglement swapping and purification. Connection establishment is performed by executing the RuleSet in each node instead of performing synchronization with neighboring nodes for each operation, so that it can reduce the number of message exchange and eventually build a more scalable quantum network.

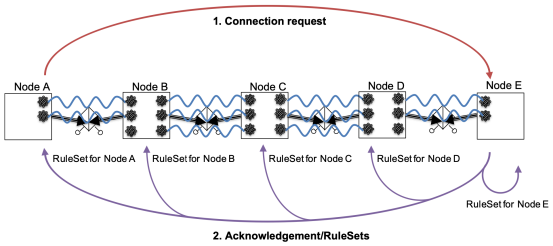

Figure 6.4: Connection Setup in the RuleSet-based quantum network from [4]

Chapter 7

Evaluation

7.1 Experiment

The author performed several experiments using QuISP (Quantum Internet Simulation Package) and validated the behavior of the proposed protocol.

7.1.1 Validation of the Protocol

Setting

Table 7.1 is the table of parameters that are used in this experiment.

Parameter	Value
Total simulation time (s)	20
Waiting time for the initial notification of link generation (s)	10
Interval for sending each ConnectionSetupRequest (s)	2
Number of qubits in the buffer	100
Number of link-level Bell pairs required by each RuleSet	100
The type of RuleSet	Tomography

Table 7.1: A table of parameters in the experiment

The author performs a two-node topology with the SenderReceiver link architecture.

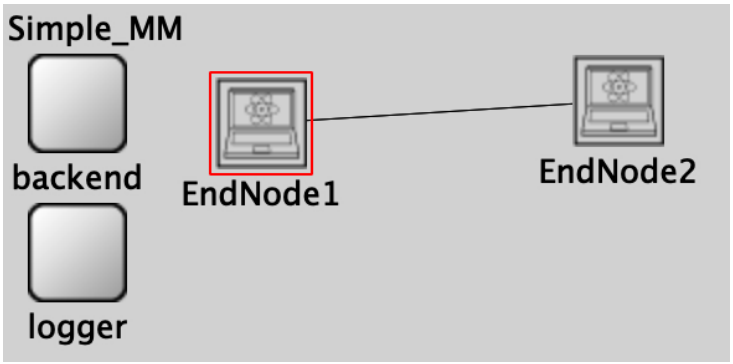

Figure 7.1: The figure of the network topology used in the experiment

Result

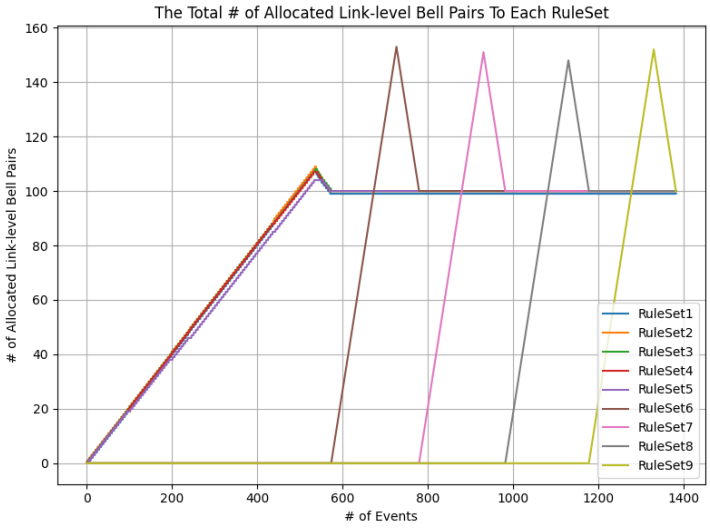

Figure 7.2: The number of link-level Bell pairs allocated to each RuleSet

Fig 7.2 shows the number of link-level Bell pairs that are allocated to each RuleSet. This figure shows that several RuleSets are already stored before the the generation of link-level Bell pairs is notified, and link-level Bell pairs are allocated to those RuleSet once the generation starts, which is a clear sign of multiplexing on a RuleSet-based quantum network. It also shows that the number of allocated link-level Bell pairs declines. This clearly indicates that link-level Bell pairs that are allocated but not consumed by each RuleSet were released after the relevant connection is torn down.

7.1.2 The Case When More Link-level Bell Pairs Are Required

Setting

Table 7.2 is the table of parameters that are used in this experiment.

Parameter	Value
Total simulation time (s)	20
Waiting time for the initial notification of link generation (s)	10
Interval for sending each ConnectionSetupRequest (s)	2
Number of qubits in the buffer	100
Number of link-level Bell pairs required by each RuleSet	200
The type of RuleSet	Tomography

Table 7.2: A table of parameters in the experiment

Result

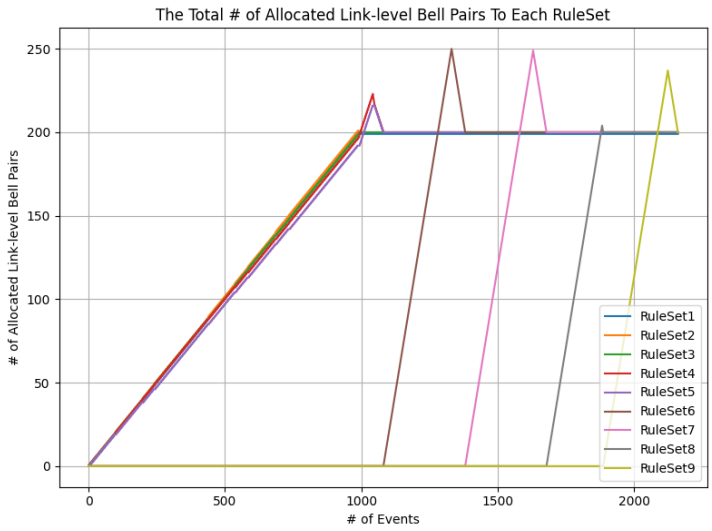

Figure 7.3: The figure of the number of link-level Bell pairs allocated to each RuleSet

Each RuleSet in this experiment requires twice as many link-level Bell pairs as how many Bell pairs generated in each round. Fig 7.3 shows that it takes more than one thousands events to finish multiplexing, which is more than twice as many events as how many it took in the previous experiment.

7.1.3 The Case When More Link-level Bell Pairs Are Generated

Setting

Table 7.3 is the table of parameters that are used in this experiment.

Parameter	Value
Total simulation time (s)	20
Waiting time for the initial notification of link generation (s)	10
Interval for sending each ConnectionSetupRequest (s)	2
Number of qubits in the buffer	200
Number of link-level Bell pairs required by each RuleSet	100
The type of RuleSet	Tomography

Table 7.3: A table of parameters in the experiment

Result

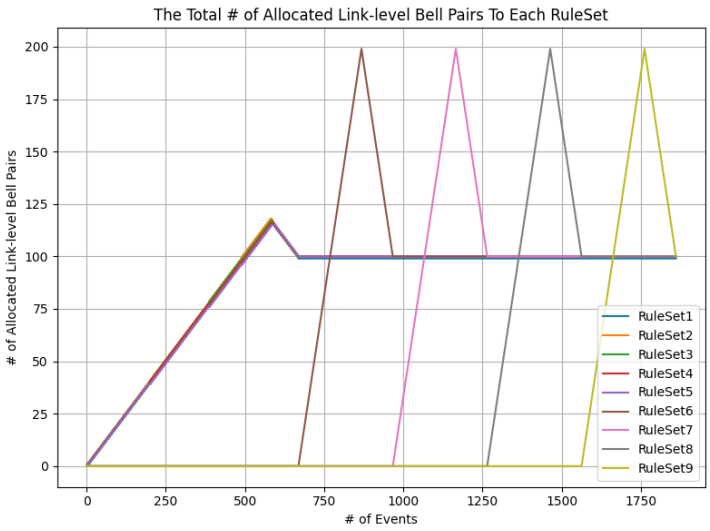

Figure 7.4: The figure of the number of link-level Bell pairs allocated to each RuleSet

The number of link-level Bell pairs generated in each round is twice as many as how many it is required by each RuleSet. In the two previous experiments, one hundred link-level Bell pairs are generated in each round and more than one hundred link-level Bell pairs are allocated to a single RuleSet, which means that it takes more than one round of the generation of link-level Bell pairs until the execution of each RuleSet is terminated. However, this experiment takes only one round of the generation of link-level Bell pairs until a single RuleSet is successfully executed. This means that it is more time-efficient if the number of link-level Bell pairs generated in each round exceeds that of the total number of those required by a single RuleSet.

Chapter 8

Conclusion

8.1 Conclusion

This thesis proposed the link management protocol for quantum network, which provides coordinated decision about two aspects. One is the upcoming link allocation policy, which are the list of RuleSets, or operations each node needs to execute in order to establish an end-to-end connection. The other one is the particular Bell pair that the next link allocation policy should be applied to. Also, the author validated the approach by performing several simulation for a RuleSet-based quantum network.

8.2 Future work

The quantum network requires more advanced multiplexing methods, such as a prioritized multiplexing, in order to enhance the overall performance of the entire network. It is also desirable to develop congestion control techniques for a quantum network to figure out the better use of resources and void congestion control on the overall network.

Acknowledgement

First and foremost, I would like to express my deepest gratitude to professor Rodney Van Meter, who has supervised my research activities in the last six years in AQUA. I would also like to show my gratitude Dr.Takahiko Satoh, who is now an associate professor at the School of Science and Engineering at Keio University, for his patient to nurture my ability in terms of all the aspects of my research, such as problem definition, proposal, implementation and presentation. Next, I would like to appreciate Dr.Michal Hajdušek for guiding me to the world of quantum nonlinear physics and offering me an opportunity of simulating quantum synchronization, which is an intriguing phenomena which can be observed in open quantum system. Almost, I feel grateful Dr.Shota Nagayama for supervising both my presentation and thesis almost every night, even at 1:00 a.m. If my thesis deserves my Master's degree, I would not have written this without his huge support. Furthermore, I would like to thank Kentaro Teramoto, who has patiently taught me how to use QuISP and helped me with debugging every time I became stuck. I would like to thank all the faculties members and all the students I have met in the Nakamura lab.

Furthermore, I'm extremely grateful to Nobufumi Fukumoto, who is my private mentor and has supervised my activities as a software developer for the last six years. I became interested in the field of infrastructure, especially network architecture, after I heard so many interesting episodes when he was designing large-scale enterprise networks. I would not have chosen to work on quantum networking for the research topic for my Master's thesis, and even pursue my career as a site reliability engineer if I have not met him six years ago.

Lastly but not least, I would like to extend my deepest gratitude to all of my family members, including my cats Maru and Uni, who have supported me for the last twenty five years.

Reference

[1] Elias Riedel Gårding, Nicolas Schwaller, Chun Lam Chan, Su Yeon Chang, Samuel Bosch, Frederic Gessler, Willy Robert Laborde, Javier Naya Hernandez, Xinyu Si, Marc-André Dupertuis, et al. Bell diagonal and werner state generation: Entanglement, non-locality, steering and discord on the ibm quantum computer. *Entropy*, 23(7):797, 2021.

[2] R. Van Meter, T.D. Ladd, W.J. Munro, and K. Nemoto. System design for a long-line quantum repeater. *IEEE/ACM Transactions on Networking*, 17(3):1002–1013, JUN 2009.

[3] Axel Dahlberg, Matthew Skrzypczyk, Tim Coopmans, Leon Wubben, Filip Rozpędek, Matteo Pompili, Arian Stolk, Przemysław Pawełczak, Robert Knegjens, Julio de Oliveira Filho, Ronald Hanson, and Stephanie Wehner. A link layer protocol for quantum networks. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19. ACM, AUG 2019.

[4] Takaaki Matsuo. Simulation of a dynamic, ruleset-based quantum network, 2019.

[5] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6/7):467–488, 1982.

[6] Charles H. Bennett and Gilles Brassard. Quantum cryptography: Public key distribution and coin tossing. *Theoretical Computer Science*, 560:7–11, Dec 2014.

[7] Artur K. Ekert. Quantum cryptography based on bell's theorem. *Phys. Rev. Lett.*, 67:661–663, Aug 1991.

[8] I K Kominis, T W Kornack, J C Allred, and M V Romalis. A subfemtotesla multichannel atomic magnetometer. *Nature*, 422(6932):596–9, Apr 2003.

[9] Rodney Van Meter and Simon J. Devitt. The path to scalable distributed quantum computing. *Computer*, 49:31–42, 2016.

[10] Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 4(05):883–898, 2006.

[11] Richard Jozsa, Daniel S Abrams, Jonathan P Dowling, and Colin P Williams. Quantum clock synchronization based on shared prior entanglement. *Physical Review Letters*, 85(9):2010, 2000.

[12] E. T. Khabiboulline, J. Borregaard, K. De Greve, and M. D. Lukin. Optical interferometry with quantum networks. *Phys. Rev. Lett.*, 123:070504, Aug 2019.

[13] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299(5886):802–803, 1982.

[14] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller. Quantum repeaters: The role of imperfect local operations in quantum communication. *Phys. Rev. Lett.*, 81:5932–5935, Dec 1998.

[15] M Horne. " event-ready-detectors" bell experiment via entanglement swapping. *Physical Review Letters*, 71(26), 1993.

[16] Charles H. Bennett, Herbert J. Bernstein, Sandu Popescu, and Benjamin Schumacher. Concentrating partial entanglement by local operations. *Phys. Rev. A*, 53:2046–2052, Apr 1996.

[17] 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.*, 70:1895–1899, Mar 1993.

[18] Takaaki Matsuo, Clément Durand, and Rodney Van Meter. Quantum link bootstrapping using a ruleset-based communication protocol. *Physical Review A*, 100(5):052320, 2019.

[19] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, USA, 10th edition, 2011.

[20] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Phys. Rev.*, 47:777–780, May 1935.

[21] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001.

[22] N. Lütkenhaus, J. Calsamiglia, and K.-A. Suominen. Bell measurements for teleportation. *Phys. Rev. A*, 59:3295–3300, May 1999.

[23] Ji-Gang Ren, Ping Xu, Hai-Lin Yong, Liang Zhang, Sheng-Kai Liao, Juan Yin, Wei-Yue Liu, Wen-Qi Cai, Meng Yang, Li Li, et al. Ground-to-satellite quantum teleportation. *Nature*, 549(7670):70–73, 2017.

[24] Angela Sara Cacciapuoti, Marcello Caleffi, Rodney Van Meter, and Lajos Hanzo. When entanglement meets classical communications: Quantum teleportation for the quantum internet. *IEEE Transactions on Communications*, 68(6):3808–3833, 2020.

[25] Jian-Wei Pan, Dik Bouwmeester, Harald Weinfurter, and Anton Zeilinger. Experimental entanglement swapping: Entangling photons that never interacted. *Phys. Rev. Lett.*, 80:3891–3894, May 1998.

[26] H. de Riedmatten, I. Marcikic, J. A. W. van Houwelingen, W. Tittel, H. Zbinden, and N. Gisin. Long-distance entanglement swapping with photons from separated sources. *Phys. Rev. A*, 71:050302, May 2005.

[27] Xiaojun Jia, Xiaolong Su, Qing Pan, Jiangrui Gao, Changde Xie, and Kunchi Peng. Experimental demonstration of unconditional entanglement swapping for continuous variables. *Physical review letters*, 93(25):250503, 2004.

[28] Bogdan-Cãlin Ciobanu, Voichiţa Iancu, and Pantelimon George Popescu. Entanglenetsat: A satellite-based entanglement resupply network. *IEEE Access*, 10:69963–69971, 2022.

[29] Pei-Shun Yan, Lan Zhou, Wei Zhong, and Yu-Bo Sheng. Feasible time-bin entanglement purification based on sum-frequency generation. *Opt. Express*, 29(2):571–583, Jan 2021.

[30] Rainer Reichle, Dietrich Leibfried, Emanuel Knill, Joseph Britton, Brad Blakestad, John Jost, C. Langer, R Ozeri, Signe Seidelin, and David Wineland. Experimental purification of two-atom entanglement. (443), 2006-10-19 2006.

[31] Norbert Kalb, Andreas A Reiserer, Peter C Humphreys, Jacob JW Bakermans, Sten J Kamerling, Naomi H Nickerson, Simon C Benjamin, Daniel J Twitchen, Matthew Markham, and Ronald Hanson. Entanglement distillation between solid-state quantum network nodes. *Science*, 356(6341):928–932, 2017.

[32] Rodney Van Meter, Ryosuke Satoh, Naphan Benchasattabuse, Kentaro Teramoto, Takaaki Matsuo, Michal Hajdušek, Takahiko Satoh, Shota Nagayama, and Shigeya Suzuki. A quantum internet architecture. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 341–352. IEEE, 2022.

[33] Luciano Aparicio and Rodney Van Meter. Multiplexing schemes for quantum repeater networks. In *Quantum Communications and Quantum Imaging IX*, volume 8163, pages 59–70. SPIE, 2011.

[34] Gregor V. Bochmann. Finite state description of communication protocols. *Computer Networks (1976)*, 2(4):361–372, 1978.

[35] Ryosuke Satoh, Michal Hajdušek, Naphan Benchasattabuse, Shota Nagayama, Kentaro Teramoto, Takaaki Matsuo, Sara Ayman Metwalli, Poramet Pathumsoot, Takahiko Satoh, Shigeya Suzuki, et al. Quisp: a quantum internet simulation package. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 353–364. IEEE, 2022.

[36] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, Brussels, BEL, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[37] Daniel Gottesman, Thomas Jennewein, and Sarah Croke. Longer-baseline telescopes using quantum repeaters. *Phys. Rev. Lett.*, 109:070503, Aug 2012.

[38] Peter Komar, Eric M Kessler, Michael Bishof, Liang Jiang, Anders S Sørensen, Jun Ye, and Mikhail D Lukin. A quantum network of clocks. *Nature Physics*, 10(8):582–587, 2014.

[39] Michael Ben-Or and Avinatan Hassidim. Fast quantum byzantine agreement. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 481–485, New York, NY, USA, 2005. Association for Computing Machinery.

[40] Ivan B Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner. Secure identification and qkd in the bounded-quantum-storage model. In *Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27*, pages 342–359. Springer, 2007.

[41] Cody Jones, Danny Kim, Matthew T Rakher, Paul G Kwiat, and Thaddeus D Ladd. Design and analysis of communication protocols for quantum repeater networks. *New Journal of Physics*, 18(8):083015, 2016.

[42] *Parameter Estimation for Scientists and Engineers, Appendix C: Positive Semidefinite and Positive Definite Matrices*, pages 259–263. John Wiley & Sons, Ltd, 2007.

[43] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2 edition, 2012.