

Master's Thesis (Academic Year 2023)

Fairness Guarantee and Quality of Service for Quantum Internet Applications

Keio University, Graduate School of Media and Governance
Nozomi Tanetani

修士論文要旨 - 2023 年度 (令和 5 年度)

量子インターネットアプリケーションのための公平性保証と QoE

ほげ。

キーワード:

1. 量子インターネット

慶應義塾大学大学院 政策・メディア研究科
種谷 望

Abstract of Master's Thesis - Academic Year 2023

Fairness Guarantee and Quality of Service for Quantum Internet Applications

hoge

Keywords :

1. Quantum Internet, 2. Fairness of Internet, 3. Resorce Allocation

Keio University, Graduate School of Media and Governance

Nozomi Tanetani

Contents

1	Introduction	1
1.1	Background	1
1.2	Research Contribution	1
1.3	Thesis Structure	1
2	Theory of Quantum Information and Quantum Network	3
2.1	History of Quantum Information and Quantum Network	3
2.2	Qubit	4
2.2.1	Dirac Notation	4
2.3	Multiple Qubit System	4
2.4	Quantum Gates	5
2.4.1	Single Qubit Gates	5
2.4.2	Controlled Gates	6
2.5	Superposition	7
2.6	Entanglement	7
2.7	Bell States	8
2.8	Quantum Teleportation	9
2.9	Quantum Network	11
2.9.1	Quantum Repeater Network	11
2.9.2	Quantum Internet Simulator	11
3	Theory of Classical Teletraffic	12
3.1	Telephone Traffic	12
3.2	Poisson Process	12
3.3	Actual Traffic Pattern in IP Network	13
3.4	Self-similar Model	14
3.5	Traffic Matrix	15
3.6	Gravity Model	17
4	Problem Definition and Proposal	18
4.1	Problem Definition	18
4.2	Proposal	18
5	Implementation	20
5.1	Implementation	20
5.1.1	Pseudo Code	20

5.1.2	Demo on QuISP	21
6	Evaluation	23
6.1	Evaluation	23
6.1.1	Unit Test	23
6.1.2	Traffic Matrix Generation Errors	24
7	Conclusion	25
7.1	Conclusion	25
	Acknowledgment	30
A	Appendix	31
A.1	Traffic Matrix Generation with Gravity Model Code for QUISP	31
A.2	Gravity Model Sample Code with Python	32
A.3	Unit Test Code	34

List of Figures

2.1	Quantum Teleportation Example, Alice to Bob	9
2.2	QuISP Display while running simulation	11
3.1	Exponential Traffic Pattern, adapted from [20] Fig.2	13
3.2	Actual Traffic Pattern, adapted from [20] Fig.2	14
3.3	Self-Similar Traffic Pattern, adapted from [20] Fig.2	15
3.4	4-node Network	16
5.1	Input Display in QuISP	21
5.2	Result Display in QuISP	21
6.1	Evaluation in Very Simple Case	23

List of Tables

3.1	The Features of Traffic, based on [23] Table.2	15
6.1	Generation Errors	24

List of Code Listings

A.1	Gravity Generator Function	31
A.2	inside of initialize function	32
A.3	Generate Simple Network with Python	32
A.4	Gravity Generator Function with Python	33
A.5	Test code for confirming the calculation	34

Chapter 1

Introduction

1.1 Background

The Quantum Internet is a new internet that uses quantum entanglement and is expected to be a technology that will make up for the weaknesses of the current Internet. Specifically, more robust cryptographic algorithms, faster solving of consensus problems and increased computational power through distributed quantum computation. However, the Quantum Internet has not yet reached the practical stage due to many hardware limitations. Nevertheless, many cryptographic algorithms and protocols are still being proposed so research activities on the Quantum Internet are flourishing.

In implementing the Quantum Internet, it is necessary to test the network. This is done to make sure that the protocols, etc. work properly for user use. In order to test the network, we need traffic data. To prepare traffic data, you need to use real data or generate it stochastically. However, since the Quantum Internet is not yet in practical use, it is not possible to use real data. Therefore, it needs to be generated stochastically. In the current Internet, traffic data is generated from the gravity model and the maximum entropy model, etc. The purpose of this paper is to establish a method of traffic data generation for the Quantum Internet.

1.2 Research Contribution

The main contribution of this project is the establishment of a basis for traffic matrix generation in the Quantum Internet. The traffic matrix generation method used in the current Internet has been applied to the Quantum Internet. Thereby, implemented traffic data generation for a Quantum Internet simulator to help simulate the Quantum Internet. In this way, It have been shown the basis for traffic testing in the Quantum Internet.

1.3 Thesis Structure

This thesis is constructed as follows.

Chapter 2 describes the fundamentals of quantum information and the Quantum Internet. In chapter 3, I will introduce the knowledge of the traffic in classical communications

and findings from previous studies. Chapter 4 details the problem definition of this research and the methods used to solve the problem. Chapter 5 describes how the method was implemented in a Quantum Internet simulator with actual code. Chapter 6 describes the evaluation of this research. It describes how to test the implemented functions. Chapter 7 provides a summary of this research and describes the future development of the research.

Chapter 2

Theory of Quantum Information and Quantum Network

2.1 History of Quantum Information and Quantum Network

Quantum computers were proposed by Richard Feynman in 1982[1]. It was proposed to simulate quantum systems with a computer based on quantum mechanics since classical computers could not simulate them accurately. Later, in 1992, the first algorithm for a quantum computer was proposed by Deutsch and Jozsa[2]. In addition, Shor proposed Shor's algorithm for performing factorization in polynomial time[3], and Grover proposed Grover's algorithm for performing data retrieval in polynomial time[4] on a quantum computer. Currently, research and discussions on quantum computers are very active, and even today, many technologies and new quantum algorithms are being announced for the realization of quantum computers.

On the other hand, along with quantum computers, the field of quantum networks has continued to grow to this day. In 1984, the first quantum cryptography protocol was proposed by Bennett and Brassard[5]. This algorithm is called BB84. This is a famous cryptographic algorithm that uses quantum mechanics, and it has brought quantum mechanics into the spotlight in the field of communications as well. Shor's algorithm introduced earlier has also attracted attention because of its potential to break the current cryptographic algorithm, the RSA cipher, which is used due to its computational security of prime factorization. In 1998, the "quantum repeater" was proposed, which is a quantum method of relaying for quantum communications over long distances[6][7].

Currently, there are two main types of quantum networks being actively researched. The first is called a quantum key distribution network, which uses algorithms such as BB84 and relays classically. This type of network is already in the experimental stage on a large scale and is also being used commercially[8][9][10]. The other type of network is called a quantum repeater network. It is also known as the Quantum Internet. This is an entanglement-based communication method using the quantum repeater. Quantum Internet can provide more robust security than quantum key distribution networks, but like quantum computers, there are still many hurdles to overcome before we can use it

practically, such as increasing the accuracy of relaying using quantum repeaters. This project is about the Quantum Internet.

2.2 Qubit

In classical information theory, information is represented by bits. In quantum information theory, information is represented by Qubit. A bit can take the value of 0 or 1. On the contrary, qubit can take on the superposition state of 0 and 1. While classical bits are implemented by turning on and off voltages, etc., qubits are implemented by quantum states.

2.2.1 Dirac Notation

Dirac notation is a notation for quantum states proposed by Dirac[11]. In Dirac notation, quantum states are represented by column vectors $|\psi\rangle$. $|\rangle$ is called "ket" and column vector is called "statevector". A single qubit can be represented by a Dirac notation as follows,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \quad (2.1)$$

$|\psi\rangle$ is a superposition of $|0\rangle$ and $|1\rangle$. $|0\rangle$ and $|1\rangle$ are computational basis states and defined as follows,

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad (2.2)$$

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (2.3)$$

In equation 2.1, α and β are complex numbers, meaning probability amplitudes. In simple terms, these numbers represent the probability of convergence to that computational basis. If you would like to know the probability of convergence to that computational basis, you can calculate it by squaring the probability amplitude. In other words, the following equation holds in the probability amplitude.

$$|\alpha|^2 + |\beta|^2 = 1 \quad (2.4)$$

In other words, the state of a qubit is a vector in a two-dimensional complex vector space. $|0\rangle$ and $|1\rangle$ form an orthonormal basis for this vector space[12].

2.3 Multiple Qubit System

Now, for 2 qubits state $|\psi\rangle$, it can be represented as follows.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle = \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix} \quad (2.5)$$

As you can see, there are four computational basis. The computational basis for N qubits, and the size of the statevector, is 2^N . In other words, the space that a qubit can represent increases exponentially.

If you would like to write multiple qubit states from multiple single qubit, you can represent them by taking the tensor product.

$$|\psi\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle \quad (2.6)$$

There are also quantum states that cannot be decomposed and written as independent one-qubit states, as shown below.

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \quad (2.7)$$

Such a state is called "entangled state"(section 2.6).

2.4 Quantum Gates

Classical computers have what are called logic gates, and quantum gates are their counterparts. All quantum gates can be represented by a matrix. Also, all quantum gates satisfy the unitary matrix, $U^\dagger U = U U^\dagger = I$. U is the unitary matrix, U^\dagger is the adjoint of U , and I is the identity matrix. Multiple quantum gates can be regarded as a single quantum gate by multiplying them together since they are a matrix.

There are two main types of quantum gates: single qubit gates and controlled gates (multiple qubit gates).

2.4.1 Single Qubit Gates

A single qubit gate is a quantum gate that apply to a single qubit[13]. There are many different types of gates.

X gate(NOT gate)

X gate (sometimes called the NOT gate) is the NOT gate in classical gates.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad (2.8)$$

This gate flip the bits like the classic NOT gate.

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \quad (2.9)$$

Z gate(Phase gate)

Z gate is also called phase gate.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (2.10)$$

This gate flip the phase when the qubit is $|1\rangle$.

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \quad (2.11)$$

It has no effect in $|0\rangle$.

$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle \quad (2.12)$$

In the $|1\rangle$ state, the phase flips so the state to $-|1\rangle$.

Hadamard gate(H gate)

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (2.13)$$

This gate is known to be a gate that, when applied to a qubit in the $|0\rangle$ or $|1\rangle$, creates a 50:50 superposition of states.

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad (2.14)$$

U gate

There are other famous single qubit gates as well. However, there is a constraint that quantum gates are unitary matrices, and we can use this to generalize. Its generalized single qubit gate is the U gate.

$$U(\theta, \phi, \lambda) = \begin{pmatrix} \cos \frac{\theta}{2} & -e^{i\lambda} \sin \frac{\theta}{2} \\ e^{i\lambda} \sin \frac{\theta}{2} & e^{i\lambda+i\phi} \cos \frac{\theta}{2} \end{pmatrix} \quad (2.15)$$

U gate requires three parameters: θ, ϕ, λ . This gate allows all single qubit gates to be represented.

2.4.2 Controlled Gates

A control gate is a gate that apply to two or more qubits[13].

CNOT (controlled-X) gate

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2.16)$$

The CNOT gate is a gate that causes the second qubit to flip when the first qubit is in the $|1\rangle$ state.

$$CNOT\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}, |0\rangle\right) = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (2.17)$$

A controlled gate can create an entanglement.

An important element of quantum circuits is that there can be any number of ways to implement the circuit. A controlled gate has a much higher error rate than a single qubit gate when operated in a real device. Therefore, one of the challenges for researchers is how to implement the same quantum circuit with fewer controlled gates.

Another important point is that all quantum gates can be decomposed into single qubit gates and CNOT gates. Since all single qubit gates can be represented by U gates, all quantum circuits can be made with U gates and CNOT gates. This is equivalent to saying that in classical circuits, all circuits can be made with NAND gates.

2.5 Superposition

A superposition is a state unique to quantum states in which $|0\rangle$ and $|1\rangle$ exist simultaneously.

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad (2.18)$$

The number in front of the two computational bases is called the probability amplitude and indicates the probability of convergence to that basis when measured. Superposition is a phenomenon unique to quantum states, and quantum states converge to a single state when measured. In other words, in the case of qubits, they converge to 0 or 1. If you would like to know the probability of convergence, you can calculate it by squaring the probability amplitude. In equation 2.18, $(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ so the probability of convergence to 0 and 1 is $1/2$. This superposition feature allows a quantum computer to perform calculations in a larger space than a classical computer.

2.6 Entanglement

Entanglement is a quantum state consisting of multiple qubits is a state in which the qubits are not independent and have some quantum mechanical correlation[12]. If a quantum state consisting of multiple qubits cannot be represented as individual qubits, then those qubits are said to be entangled.

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad (2.19)$$

The quantum state of the equation 2.19 is not an entangled state, since it can be split into individual qubits.

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (2.20)$$

The quantum state of the equation 2.20 is an entangled state, since it cannot be split into individual qubits.

The important point of entanglement is that these quantum states are correlated. For example, if you measure the quantum states of equation 2.20 one qubit at a time, if the first result is zero, the other result will always be zero, and if it is one, the other result will always be one. This happens no matter how far apart the two qubits are.

2.7 Bell States

The Bell state is the simplest pair of entanglements in two qubits. The following four states are called Bell states.

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (2.21)$$

$$|\phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \quad (2.22)$$

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \quad (2.23)$$

$$|\psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \quad (2.24)$$

Bell states use as a basis. This is called the Bell basis. Using Bell basis, the computational basis can be written as follows[14],

$$|00\rangle = \frac{1}{\sqrt{2}}(|\phi^+\rangle + |\phi^-\rangle) \quad (2.25)$$

$$|01\rangle = \frac{1}{\sqrt{2}}(|\psi^+\rangle + |\psi^-\rangle) \quad (2.26)$$

$$|10\rangle = \frac{1}{\sqrt{2}}(|\psi^+\rangle - |\psi^-\rangle) \quad (2.27)$$

$$|11\rangle = \frac{1}{\sqrt{2}}(|\phi^+\rangle - |\phi^-\rangle) \quad (2.28)$$

Bell basis is used in quantum teleportation and entanglement swapping, which is a technique of quantum repeater[14].

2.8 Quantum Teleportation

According to [12],

Quantum teleportation is a technique for moving quantum states around, even in the absence of a quantum communications channel linking the sender of the quantum state to the recipient.

When people hear the word teleportation, they think of instantaneous travel. In other words, we would expect objects or information to travel faster than the speed of light! The entanglement correlation happens no matter how far apart the two qubits are, which makes the expectation even higher. However, this is not the case with quantum teleportation. Quantum teleportation requires the transmission of classical information, which of course does not exceed the speed of light. Therefore, superluminal communication cannot be realized by quantum teleportation.

This section uses an example to explain quantum teleportation. The explanation is based on the derivation by [15] Suppose Alice would like to send Bob any one of the following qubits by quantum teleportation.

$$|\psi\rangle_{A_1} = \alpha|0\rangle + \beta|1\rangle \quad (2.29)$$

For quantum teleportation to work, Alice and Bob must share the Bell state. Therefore, it is assumed that the following Bell states are shared.

$$|\phi^+\rangle_{A_2B} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (2.30)$$

Let A_1 be the qubit that Alice would like to send, A_2 be the qubit of one of the Bell states that Alice has, and B be the qubit of one of the Bell states that Bob has. This is represented in the figure below.

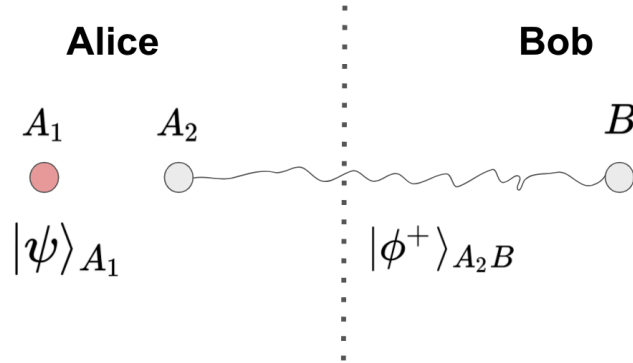


Figure 2.1: Quantum Teleportation Example, Alice to Bob

Equation 2.29 and 2.30 can be combined to be written as follows.

$$\begin{aligned} |\psi\rangle_{A_1} |\phi^+\rangle_{A_2B} &= (\alpha|0\rangle + \beta|1\rangle)_{A_1} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)_{A_2B} \\ &= \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle)_{A_1A_2B} \end{aligned} \quad (2.31)$$

This is the initial state. Here, Alice measures A_1 and A_2 on a Bell basis. Therefore, let's rewrite equation 2.31 in Bell basis.

$$\begin{aligned}
 |\psi\rangle_{A_1}|\phi^+\rangle_{A_2B} &= \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle)_{A_1A_2B} \\
 &= \frac{1}{\sqrt{2}}(\alpha(|\phi^+\rangle + |\phi^-\rangle)|0\rangle \\
 &\quad + \alpha(|\psi^+\rangle + |\psi^-\rangle)|1\rangle \\
 &\quad + \beta(|\psi^+\rangle - |\psi^-\rangle)|0\rangle \\
 &\quad + \beta(|\phi^+\rangle - |\phi^-\rangle)|1\rangle)
 \end{aligned} \tag{2.32}$$

This equation 2.32 can be further rewritten by regrouping it by Bell basis as follows.

$$\begin{aligned}
 |\psi\rangle_{A_1}|\phi^+\rangle_{A_2B} &= \frac{1}{\sqrt{2}}(\alpha(|\phi^+\rangle + |\phi^-\rangle)|0\rangle \\
 &\quad + \alpha(|\psi^+\rangle + |\psi^-\rangle)|1\rangle \\
 &\quad + \beta(|\psi^+\rangle - |\psi^-\rangle)|0\rangle \\
 &\quad + \beta(|\phi^+\rangle - |\phi^-\rangle)|1\rangle) \\
 &= \frac{1}{2}(|\phi^+\rangle_{A_1A_2}(\alpha|0\rangle + \beta|1\rangle)_B \\
 &\quad + |\phi^-\rangle_{A_1A_2}(\alpha|0\rangle - \beta|1\rangle)_B \\
 &\quad + |\psi^+\rangle_{A_1A_2}(\alpha|1\rangle + \beta|0\rangle)_B \\
 &\quad + |\psi^-\rangle_{A_1A_2}(\alpha|1\rangle - \beta|0\rangle)_B)
 \end{aligned} \tag{2.33}$$

Therefore, we can see that the state of Bob's qubit changes depending on the results of Alice's Bell state measurement. However, this can be undone by a simple gate operation with X gate and Z gate as follows.

$$|\psi\rangle_{A_1} = \alpha|0\rangle + \beta|1\rangle \tag{2.34}$$

$$Z|\psi\rangle_{A_1} = \alpha|0\rangle - \beta|1\rangle \tag{2.35}$$

$$X|\psi\rangle_{A_1} = \alpha|1\rangle + \beta|0\rangle \tag{2.36}$$

$$ZX|\psi\rangle_{A_1} = \alpha|1\rangle - \beta|0\rangle \tag{2.37}$$

This means that Alice has to inform Bob of the results of the Bell state measurement. Bob can reproduce the quantum state that Alice would like to send by manipulating the gate according to the results sent by Alice. This is how information is transmitted via quantum teleportation.

2.9 Quantum Network

2.9.1 Quantum Repeater Network

Quantum repeater network is an entanglement-based communication method using the quantum repeater. It is also called the Quantum Internet. Each end node shares a Bell pair and exchanges information through quantum teleportation.

Here is Quantum Internetworking flow briefly below.

1. Initiator sends a setup request to responder to share n Bell pairs
2. Perform an operation to share Bell pairs.
3. Someone(it depends on link type) sends the result of whether the sharing was a success or failure to the initiator and responder.

Entangled states are symmetric so there is no source and destination. Therefore, there is no such thing as ingress traffic and egress traffic as in the current Internet. However, there is such a thing as an initiator and a responder in a server-client type application. This is because when one node starts communication, it needs to send a connection setup request to another node. The node which sends the connection setup request is called the initiator, and the node which receives the request is called the responder.

2.9.2 Quantum Internet Simulator

Our group, AQUA, has released QuISP, a Quantum Internet simulator. [16]

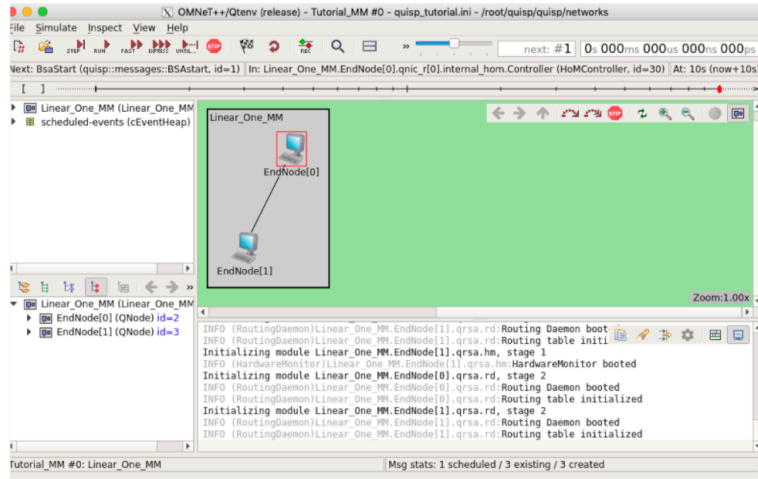


Figure 2.2: QuISP Display while running simulation

However, its functionality is limited so it does not allow us to test traffic using the gravity model, nor does it allow us to connect to multiple nodes or measure traffic.

Chapter 3

Theory of Classical Teletraffic

3.1 Telephone Traffic

Traffic is a word originally used in the fields of transportation and trade to describe the volume of traffic and trade being transacted. Teletraffic means traffic of communication. The first type of teletraffic is a telephone. There are two main definitions of traffic in telephony, one is simply the number of calls and the other is the duration of calls. In some cases, the synergistic product of these two values is called traffic density, and this is referred to as traffic[17].

3.2 Poisson Process

According to [18],

A Poisson Process is a model for a series of discrete event where the average time between events is known, but the exact timing of events is random. The arrival of an event is independent of the event before.

In terms of teletraffic, we know the amount of traffic in a unit of time, but we do not know at what interval it is communicated.

According to [18],

A Poisson Process meets the following criteria,

- Events are independent of each other. The occurrence of one event does not affect the probability another event will occur.
- The average rate (events per time period) is constant.
- Two events cannot occur at the same time.

The Poisson process can explain the characteristics of telephone traffic mathematically well. In particular, if we add an exponential distribution describing the time interval of events to the Poisson process, we can represent most of the actual telephone traffic. In fact, the beginning of teletraffic theory was based on empirical studies and measurements,

but with the observation of the Poisson nature of call arrivals, Poisson processes and exponential distributions became universal laws describing telephone traffic, and the curiosity of actually measuring it diminished. This has allowed us to generate a steady stream of traffic, and many performance indicators can be accurately predicted. As a result, the telephone tends to be less blocked as a communication network.[19]

3.3 Actual Traffic Pattern in IP Network

With the advent of the Internet, Internet traffic appeared in the field of teletraffic as well. Internet traffic is the amount of packets/data moving across a computer network at any given time. Therefore, Whereas telephone traffic is voice traffic, Internet traffic is data traffic.

While telephones occupy a single communication channel for communication, the Internet divides data into packets for transmission, and packets of other communications also flow in the communication channel. Therefore, the telephone is a centralized control system, while the Internet is a self-sustaining decentralized system. As a result, its overall behavior is very complex and difficult to understand. Since the Internet is a self-sustaining distributed system, data that exceeds the transfer limit may suddenly be sent over the network. A sudden increase in the amount of data is called a burst. One of the characteristics of Internet traffic is that, unlike telephone traffic, it cannot be explained by Poisson and exponential distributions[20]. This is because, as written earlier, the telephone and the Internet have different characteristics as communications.

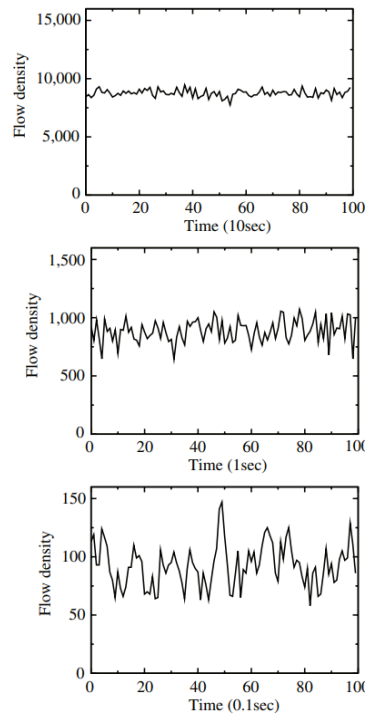


Figure 3.1: Exponential Traffic Pattern, adapted from [20] Fig.2

Figure. 3.1 is shows a graph of exponential traffic by timescale. As you can see, the exponential traffic is smoothed out as the timescale is increased.

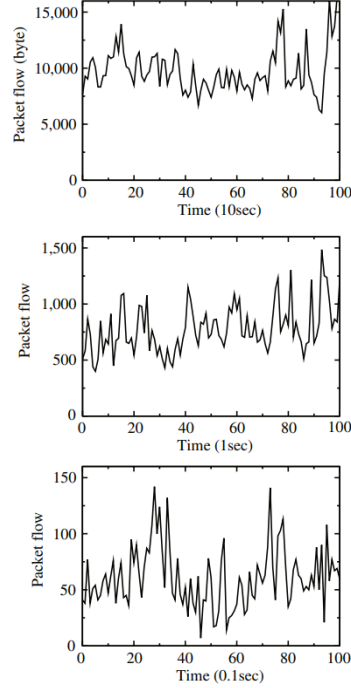


Figure 3.2: Actual Traffic Pattern, adapted from [20] Fig.2

Figure.3.2 is shows a graph of actual Internet traffic by timescale. As you can see, even with a larger timescale, the actual traffic is bumpy. This indicates that the traffic generation event occurrence is dependent on other events and that congestion can occur regardless of the timescale. Statistically, we know that the exponential nature of Internet traffic cannot be established[21][22].

3.4 Self-similar Model

The shape of Internet traffic is bumpy, even when the timescale is increased. In other words, it has a similar shape when the timescale is increased. This similarity in shape even when the scale is changed is called self-similarity or fractal. The characteristics of real Internet traffic are self-similar[21].

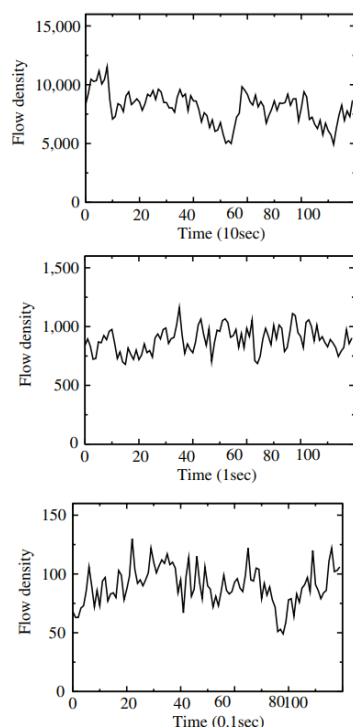


Figure 3.3: Self-Similar Traffic Pattern, adapted from [20] Fig.2

As comparing Fig.3.2, and Fig.3.3, we can see that the self-similar model is able to generate traffic similar to the actual traffic. Pseudo traffic using the self-similar model is more effective than the Poisson model in the Internet.

Table.3.1 below compares the characteristics of telephone and Internet traffic.

Table 3.1: The Features of Traffic, based on [23] Table.2

	Telephone	Internet
Link usage	Occupied, continuous	Shared, discrete
Event occurrence	Poisson	Depends on other events
Remarks	Traffic can be expressed as Poisson, so traffic testing is easy.	Events depends on other events so traffic pattern is self-similar. We can't use Poisson to test IP network.

3.5 Traffic Matrix

According to [24],

A Traffic Matrix is a matrix giving the traffic volumes between origin and destination in a network and has tremendously potential utility for IP network capacity planning and management.

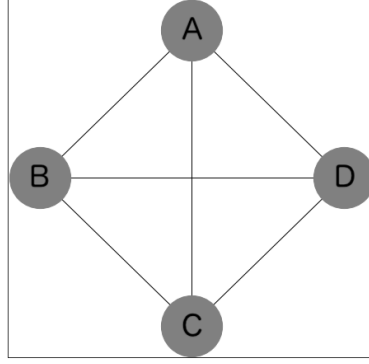


Figure 3.4: 4-node Network

Definition 1 (Traffic Matrix). *Consider a simple four-node, all-coupled network as shown in Fig.3.4. $T(i, j)$ means the amount of traffic from point j to point i .*

$$\begin{bmatrix} T(A, A) & T(A, B) & T(A, C) & T(A, D) \\ T(B, A) & T(B, B) & T(B, C) & T(B, D) \\ T(C, A) & T(C, B) & T(C, C) & T(C, D) \\ T(D, A) & T(D, B) & T(D, C) & T(D, D) \end{bmatrix} \quad (3.1)$$

In general, the traffic matrix does not have a time element. The general traffic matrix without the time element is called the static traffic matrix, and the traffic matrix with the time element is called the dynamic traffic matrix[25].

In order to perform traffic testing, we need a traffic matrix. However, It is practically impossible to prepare a real traffic matrix in current Internet. This is because it is necessary to measure all the packets, where each one comes from and where it goes. However, since there are countless packets flowing over the Internet, it is impossible to do so. So, the information that can be realistically obtained is the amount of traffic received and the amount of traffic sent at a node. The amount of traffic received is called "ingress traffic" and the amount of traffic sent is called "egress traffic". Therefore, in the current Internet, various model is used to estimate the real traffic matrix from the ingress traffic and egress traffic of each node. The most common estimation method is using the gravity model[26][27][28], but recently, estimation methods using deep learning have also been proposed[29].

Generating a traffic matrix is as difficult or more difficult than estimation. It is known that randomly generating a traffic matrix from a uniform distribution does not yield meaningful data[25]. Also, even if you are lucky enough to get the traffic matrix data of an actual network, it is unlikely that the topology will be the same as the network you want to test. The traffic matrix must be synthesized in a way that preserves the characteristics of the data so that it can be applied to the network we would like to test.

According to [30],

Traffic matrix synthesis method should satisfy the following these criteria below.

- Control
- Efficiency
- Consistency
- Simplicity

The gravity model is often used as a very simple method of generating traffic matrices[26][31]. There is also a slightly more complex max entropy model[32] and a spatio-temporal model[33][30] for generating traffic matrices.

3.6 Gravity Model

The gravity model is an application of Newton's law of universal gravitation, which is used in social sciences to estimate the amount of logistics between regions.[34] This model is one of the better ways to generate the traffic matrix as well as estimate.

Definition 2 (Simple Gravity Model[26]).

$$T(n_i, n_j) = \frac{T^{ingress}(n_i) * T^{egress}(n_j)}{T^{total}} \quad (3.2)$$

$T(n_i, n_j)$ means the amount of traffic from node j to node i. $T^{ingress}(n_i)$ is ingress traffic at node i. $T^{egress}(n_j)$ is egress traffic at node j. T^{total} is the amount of overall traffic movement in the network.

Chapter 4

Problem Definition and Proposal

4.1 Problem Definition

Create the base for traffic testing to make the Quantum Internet practical. Traffic tests in the Quantum Internet have yet to be devised. In this paper, I propose a definition of traffic in the Quantum Internet and a method to generate traffic data for traffic tests and show the base of traffic tests for the Quantum Internet. The method is implemented as a function of QuISP, the Quantum Internet simulator.

4.2 Proposal

First, the traffic in the Quantum Internet should be defined.

Definition 3 (A Traffic in Quantum Internet). *A traffic in Quantum Internet is one connection, not number of bell pairs.*

Traffic in the Quantum Internet is defined as one connection to one traffic. In other words, it does not matter how many bell pairs are shared in a single connection, or how long the connection takes. In the Quantum Internet, when a node wants to start a connection with another node, it sends a setup request to that node. Therefore, the number of setup requests sent by a node is the egress traffic, and the number of setup requests received by a node is the ingress traffic.

- Initiator address : Who start the connection?
- Responder address : Who is the responder in the connection?
- Start time : When start the connection?
- Connection length : How many Bell pairs sharing?

These are four basic elements for generating a traffic pattern. The goal of this project is to generate a static traffic matrix with the gravity model. The gravity model can calculate the amount of traffic between a particular initiator and responder, but it does

not determine the interval at which they are connected. Therefore, this project focus on the above two elements.

The specific goal of this project is to implement the function of traffic matrix generation in QuISP, a simulator of quantum Internet. For the specific simulation, assume a simple one where a single static traffic matrix is taken as input and communication is performed based on it.

The generation of the traffic matrix for the Quantum Internet is done using the gravity model. This is identical to the method traffic is generated on the current Internet. The method of the current Internet will be applied to the Quantum Internet. For the gravity model type, the simplest SimpleGravityModel is used. The reason for this is that this is the first attempt to apply the gravity model to Quantum Internet traffic generation, and in addition, the gravity model itself can be made more complex at will later. The equation for Simple Gravity Model[26][35], which represents the total amount of traffic going from point j to point i, is as follows.

Definition 4 (Simple Gravity Model for Quantum Internet).

$$T(n_i, n_j) = T^{initiator}(n_i) \times \frac{T^{responder}(n_j)}{T^{totalRes}(n_j)} \quad (4.1)$$

$T(n_i, n_j)$ means the amount of traffic from node j to node i. $T^{initiator}(n_i)$ is initiator traffic at node i. $T^{responder}(n_j)$ is responder traffic at node j. $T^{totalRes}(n_j)$ is the amount of total responder traffic expect node j.

The almost same equation for generating the traffic matrix is applied to the Quantum Internet. The only difference is in the way the total traffic amount is calculated. In the previous work, the total traffic is defined as the sum of the traffic of all the nodes including the own node. In this method, the total traffic is calculated as the sum of the traffic of the nodes other than the own node. This is because in the current QuISP, it is not expected that a node will send a setup request to its own node. Therefore, it is necessary to exclude its own node from the total traffic. The gravity model is based on the ratio of the size of the nodes to the amount of traffic. The amount of traffic is determined by the ratio of the relative sizes of other nodes.

Chapter 5

Implementation

5.1 Implementation

5.1.1 Psuedo Code

Here is the psuedo code for traffic matrix generation with simple gravity model for Quantum Internet.

Algorithm 1 Traffic Matrix Generation with Simple Gravity Model for Quantum Internet

Require: N (Number of nodes entire the network),

$T^{initiator}(n_i) : i = 1, \dots, N$; (Number of sent requests) ,

$T^{responder}(n_i) : i = 1, \dots, N$; (Number of received requests)

Ensure: $T(n_i, n_k) : i = 1, \dots, N; k = 1, \dots, N$ (Traffic matrix)

for $i = 1, \dots, N$ **do**

$T^{total}(n_k) \leftarrow 0$

for $k = 1, \dots, N$ **do**

if $i \neq k$ **then**

$T^{total}(n_k) \leftarrow T^{total}(n_k) + T^{responder}(n_k)$

end if

end for

for $k = 1, \dots, N$ **do**

if $i \neq k$ **then**

$T(n_i, n_k) \leftarrow T^{initiator}(n_i) \times (T^{responder}(n_k) / T^{total}(n_k))$


else

$T(n_i, n_k) \leftarrow 0$

end if

end for

end for

 $T(n_i, n_k)$

According to the QuISP specification, each node has its own information about where

it sent the setup request to. Therefore, the implementation does not directly generate the traffic matrix for the network as a whole. For this reason, in the gravity model, each node calculates the amount of traffic it sends from itself by retrieving data from other nodes. This is the specification in my implementation.

I implemented this in C++, the language of OMNeT++. You can see my actual code from code listings A.1.

5.1.2 Demo on QuISP

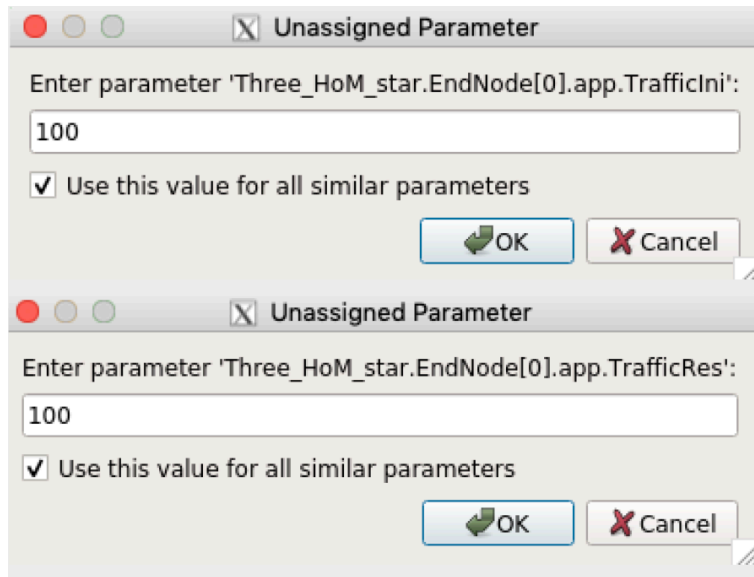


Figure 5.1: Input Display in QuISP

When you set up the simulation, you will be asked for node data, as shown in the figure above. You have to input the amount of initiator traffic and the amount of responder traffic for each node.

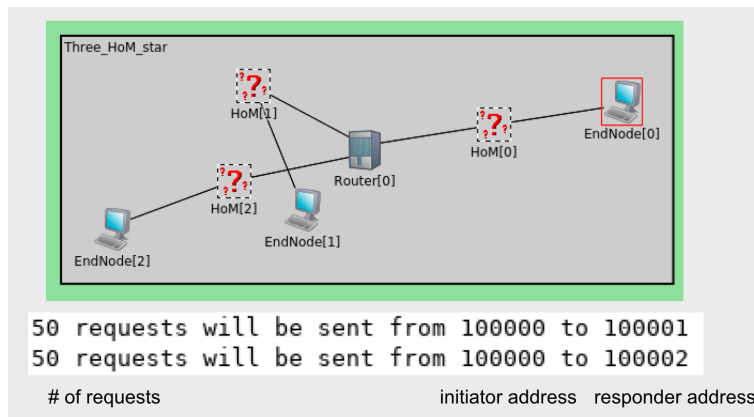


Figure 5.2: Result Display in QuISP

After entering the node data and starting the simulation, the addresses of the initiator and responder and the amount of traffic between the two will be output.

Chapter 6

Evaluation

6.1 Evaluation

In this chapter, the verification of the correctness of the actual implemented code is described. The verification method is based on the following two procedures.

1. Confirm whether my traffic matrices generation calculation is correct
2. Verify that the generated traffic matrix is correct.

QuISP still has limited functionality, so we cannot actually simulate and test a Quantum Internet.

6.1.1 Unit Test

First, confirm whether my traffic matrices generation calculation is correct. The gravity model calculates the traffic volume based on the relative ratio of each node. Therefore, I tested the implemented code in the case where the node data is very simple.

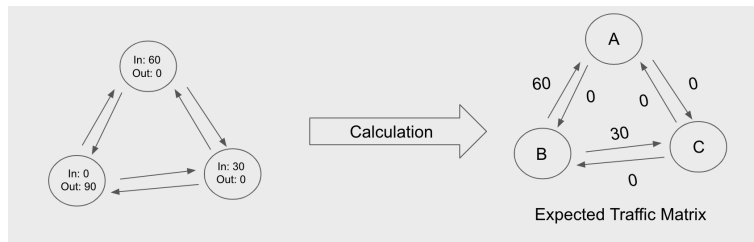


Figure 6.1: Evaluation in Very Simple Case

As shown in the figure, I use the case where the node data clearly shows the destination of all traffic. You can see my actual test code from code listings A.5. The test results correctly output the expected results, and the assertion test was passed.

For the unit test, I use Google Test to test the traffic matrix generation code that I have implemented.

6.1.2 Traffic Matrix Generation Errors

Next target is to verify that the generated traffic matrix is correct. To evaluate the generated traffic matrix, various measures of errors are used. The purpose of this is to compare the original traffic matrix and the generated traffic on an error scale to see how well the gravity model reproduces the characteristics of the original traffic.

The measures used in this project are Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). As functions, I used library functions from scikit-learn. See this page for definitions of measures of errors I used for this project.[36]

The following table transforms a randomly generated 12×12 traffic matrix from a uniform distribution into node data. Based on the node data, the traffic matrix is generated by the function I implemented. This randomly generated traffic matrix from a uniform distribution is taken as the true value, and I evaluate how well my traffic matrix generator represents the original features using measures of errors.

Table 6.1: Generation Errors

Error	Mean	Median	Std	Min	Max
MSE	632.4389	630.6597	61.1020	461.3056	815.0278
RMSE	24.7900	24.7868	1.2391	20.9479	28.3213
MAE	20.2920	20.2639	1.2122	16.2083	23.8056
MAPE	1.5686+e15	1.3448+e15	1.4074+e15	0.6073	9.3512+e15

The MAPE results are arbitrarily large. This is due to a feature of MAPE, which is that when the true value is close to zero, an extremely large value is output as the result.

In this project, there is no comparison to the gravity model. Therefore, it is important to note that this result does not imply that the gravity model is superior for generating quantum Internet traffic. This will be the basis for the evaluation of traffic generation models for the Quantum Internet, which will increase in the future.

Chapter 7

Conclusion

7.1 Conclusion

hoge

Bibliography

- [1] Richard P Feynman. Simulating physics with computers. *International journal of theoretical physics*, 21(6/7):467–488, 1982.
- [2] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.
- [3] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, Oct 1997.
- [4] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
- [5] C. H. Bennett and G. Brassard. Quantum cryptography: Public key distribution and coin tossing. In *Proceedings of IEEE International Conference on Computers, Systems, and Signal Processing*, page 175, 1984.
- [6] H-J Briegel, Wolfgang Dür, Juan I Cirac, and Peter Zoller. Quantum repeaters: the role of imperfect local operations in quantum communication. *Physical Review Letters*, 81(26):5932, 1998.
- [7] Wolfgang Dür, H-J Briegel, Juan Ignacio Cirac, and Peter Zoller. Quantum repeaters based on entanglement purification. *Physical Review A*, 59(1):169, 1999.
- [8] CH Bennett and G Brassard. The dawn of a new era for quantum cryptography: The experimental prototype is working,” sigact news 20, no. 4, 78 (1989); ch bennett et al. *Experimental Quantum Cryptography*, ” *J. Crypto*, 5(3), 1992.
- [9] Chip Elliott, Alexander Colvin, David Pearson, Oleksiy Pikalo, John Schlafer, and Henry Yeh. Current status of the darpa quantum network. In *Quantum Information and computation III*, volume 5815, pages 138–149. International Society for Optics and Photonics, 2005.
- [10] Masahide Sasaki, Mikio Fujiwara, H Ishizuka, W Klaus, K Wakui, M Takeoka, S Miki, T Yamashita, Z Wang, A Tanaka, et al. Field test of quantum key distribution in the tokyo qkd network. *Optics express*, 19(11):10387–10409, 2011.

- [11] Paul Adrien Maurice Dirac. A new notation for quantum mechanics. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 35, pages 416–418. Cambridge University Press, 1939.
- [12] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.
- [13] IBM Qiskit team. Summary of quantum operations. https://qiskit.org/documentation/locale/ja_JP/tutorials/circuits/3_summary_of_quantum_operations.html, 2022(Last Updated). Accessed: 2022-01-25.
- [14] QLEAP Education Project Keio University. Q-leap edu quantum communications 4-3 ベル状態. <https://youtu.be/rqgGsWaL9Hg>, 2021. Accessed: 2022-01-28.
- [15] QLEAP Education Project Keio University. Q-leap edu quantum communications 8-2 量子テレポテーションのプロトコル. <https://youtu.be/E6FwuDdYI7A>, 2021. Accessed: 2022-01-28.
- [16] Ryosuke Satoh, Michal Hajdušek, Naphan Benschasattabuse, Shota Nagayama, Kentaro Teramoto, Takaaki Matsuo, Sara Ayman Metwalli, Takahiko Satoh, Shigeya Suzuki, and Rodney Van Meter. Quisp: a quantum internet simulation package. *arXiv preprint arXiv:2112.07093*, 2021.
- [17] 辻 舛二. 電話トラフィック理論とその応用. 電子通信学会, 1954.
- [18] Will Koehrsen. The poisson distribution and poisson process explained. <https://towardsdatascience.com/the-poisson-distribution-and-poisson-process-explained-4e2cb17d459>, 2019. Accessed: 2021-12-18.
- [19] Walter Willinger and Vern Paxson. Where mathematics meets the internet. *Notices of the AMS*, 45(8):961–970, 1998.
- [20] 福田健介. ネットワークトラフィックの自己相似性とその生成モデル. **情報処理学会学会誌**, 45(6):603–609, 6 2004.
- [21] Will E Leland, Walter Willinger, Murad S Taqqu, and Daniel V Wilson. On the self-similar nature of ethernet traffic. *ACM SIGCOMM Computer Communication Review*, 25(1):202–213, 1995.
- [22] Vern Paxson and Sally Floyd. Wide area traffic: the failure of poisson modeling. *IEEE/ACM Transactions on networking*, 3(3):226–244, 1995.
- [23] 上田浩, 奈須野裕, 岩谷幸雄, 五十嵐隆治, 木下哲男, et al. 確率過程による lan トラフィックのモデル化における一考察. **情報処理学会論文誌数理モデル化と応用 (TOM)**, 48(SIG2 (TOM16)):167–174, 2007.
- [24] Matthew Roughan. Internet traffic matrices. https://roughan.info/project/traffic_matrix/, 2017. Accessed: 2021-12-18.

- [25] Antonio Nucci, Ashwin Sridharan, and Nina Taft. The problem of synthetically generating ip traffic matrices: Initial recommendations. *ACM SIGCOMM Computer Communication Review*, 35(3):19–32, 2005.
- [26] Yin Zhang, Matthew Roughan, Nick Duffield, and Albert Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):206–217, 2003.
- [27] Matthew Roughan, Mikkel Thorup, and Yin Zhang. Traffic engineering with estimated traffic matrices. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, pages 248–258, 2003.
- [28] Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring internet backbone traffic variability: Models metrics, measurements and meaning. In *Teletraffic Science and Engineering*, volume 5, pages 379–388. Elsevier, 2003.
- [29] Grigorios Kakkavas, Michail Kalntis, Vasileios Karyotis, and Symeon Papavassiliou. Future network traffic matrix synthesis and estimation based on deep generative models. In *30th International Conference on Computer Communication and Networks (ICCCN)*, 2021.
- [30] Paul Tune and Matthew Roughan. Spatiotemporal traffic matrix synthesis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 579–592, 2015.
- [31] Matthew Roughan. Simplifying the synthesis of internet traffic matrices. *ACM SIGCOMM Computer Communication Review*, 35(5):93–96, 2005.
- [32] Paul Tune and Matthew Roughan. Maximum entropy traffic matrix synthesis. *ACM SIGMETRICS Performance Evaluation Review*, 42(2):43–45, 2014.
- [33] Yin Zhang, Matthew Roughan, Walter Willinger, and Lili Qiu. Spatio-temporal compressive sensing and internet traffic matrices. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 267–278, 2009.
- [34] Suzanne P Evans. A relationship between the gravity model for trip distribution and the transportation problem in linear programming. *Transportation Research*, 7(1):39–61, 1973.
- [35] Matthew Roughan. How i learned to stop worrying and love traffic matrices. https://roughan.info/talks/tma_summer_school.pdf, 2016. Accessed: 2021-12-22.
- [36] scikit-learn developers. 3.3. metrics and scoring: quantifying the quality of predictions. https://scikit-learn.org/stable/modules/model_evaluation.html, 2021. Accessed: 2022-02-01.
- [37] University of TARTU Institute of Computer Science. Gravity models. <https://courses.cs.ut.ee/2011/graphmining/Main/GravityModels>, 2011. Accessed: 2021-12-22.

- [38] Bernard Fortz, Jennifer Rexford, and Mikkel Thorup. Traffic engineering with traditional ip routing protocols. *IEEE communications Magazine*, 40(10):118–124, 2002.
- [39] Takaaki Matsuo, Clément Durand, and Rodney Van Meter. Quantum link bootstrapping using a ruleset-based communication protocol. *Physical Review A*, 100(5):052320, 2019.
- [40] Takaaki Matsuo, Takahiko Satoh, Shota Nagayama, and Rodney Van Meter. Analysis of measurement-based quantum network coding over repeater networks under noisy conditions. *Physical Review A*, 97(6):062328, 2018.
- [41] Takaaki Matsuo. Simulation of a dynamic, ruleset-based quantum network. Master’s thesis, Keio University, Graduate School of Media and Governance, 2019.

Acknowledgment

hoge

Appendix A

Appendix

A.1 Traffic Matrix Generation with Gravity Model Code for QUISP

Code Listing A.1: Gravity Generator Function

```
1 void Application::gravityGenerator() {
2     int traffic_ini = par("TrafficIni");
3     int res;
4     double res_all = 0;
5
6     cTopology *topo = new cTopology("topo");
7     topo->extractByParameter("nodeType", provider.getQNode()->
8         par("nodeType").str().c_str());
9
10    for (int i = 0; i < other_end_node_addresses.size(); i++) {
11        cTopology::Node *node = topo->getNode(i);
12        auto app_module = node->getModule()->getModuleByPath(".
13            app");
14        if (app_module == nullptr) throw cRuntimeError("app_
15            module_not_found");
16        res = app_module->par("TrafficRes");
17        res_all += res;
18    }
19    for (int i = 0; i < other_end_node_addresses.size(); i++) {
20        cTopology::Node *node = topo->getNode(i);
21        auto app_module = node->getModule()->getModuleByPath(".
22            app");
23        if (app_module == nullptr) throw cRuntimeError("app_
24            module_not_found");
25        res = app_module->par("TrafficRes");
26        traffic_matrix.push_back((int)round(traffic_ini * (res /
27            res_all)));
28    }
29    int sum = std::accumulate(traffic_matrix.begin(),
30        traffic_matrix.end(), 0);
31    while (sum != traffic_ini) {
32        std::vector<int>::iterator iter = std::max_element(
33            traffic_matrix.begin(), traffic_matrix.end());
```

```

26     size_t max_index = std::distance(traffic_matrix.begin(),
27         iter);
28     if (sum > traffic_ini) {
29         traffic_matrix[max_index] -= 1;
30     } else if (traffic_ini < sum) {
31         traffic_matrix[max_index] += 1;
32     }
33     sum = std::accumulate(traffic_matrix.begin(),
34         traffic_matrix.end(), 0);
35 }
36 delete topo;

```

Code Listing A.2: inside of initialize function

```

1  if (traffic_pattern == 3) {
2      gravityGenerator();
3      for (int i = 0; i < other_end_node_addresses.size(); i++)
4          {
5              EV_INFO << traffic_matrix[i] << "requests will be
6                  sent from " << my_address << " to " <<
7                  other_end_node_addresses[i] << "\n";
8              printf("%d requests will be sent from %d to %d\n",
9                  traffic_matrix[i], my_address,
10                     other_end_node_addresses[i]);
11          }
12      return;
13  }

```

A.2 Gravity Model Sample Code with Python

Code Listing A.3: Generate Simple Network with Python

```

1  #Author: Nozomi Tanetani
2  import networkx as nx
3  import numpy as np
4  import matplotlib.pyplot as plt
5
6  def graph_generator(n, p, seed):
7      g = nx.random_graphs.fast_gnp_random_graph(n, p, seed,
8          directed = True) #Generate a random graph
9      g = g.to_undirected()
10
11     #Initialize a number of traffic per node as 0.
12     for i in g.nodes():
13         g.nodes[i]['in'] = 0
14         g.nodes[i]['out'] = 0
15
16     #Store a random number into number of traffic variable.
17     for i in g.nodes():
18         g.nodes[i]['in'] = np.random.randint(n*100,n*500)
19         tmp = g.nodes[i]['in']

```



```

19     for j in g.nodes():
20         if i is not j:
21             if j is n-1:
22                 g.nodes[j]['out'] += tmp
23             else:
24                 rand_tmp = np.random.randint(0,tmp)
25                 g.nodes[j]['out'] += rand_tmp
26                 tmp = tmp - rand_tmp
27     return g
28
29 g = graph_generator(4, 1, 13)
30
31 #Display a generated graph
32 plt.figure(figsize=(8,8))
33
34 pos = nx.circular_layout(g)
35 nx.draw_networkx_nodes(g, pos, node_size=5500, node_color='gray
36 ')
37 nx.draw_networkx_edges(g, pos)
38 nx.draw_networkx_labels(g, pos, font_size=20)
39
40 #Display
41 labeldict = nx.get_node_attributes(g,'in')
42 for i in g.nodes():
43     labeldict[i] = '\n\n' + str(labeldict[i])
44 nx.draw_networkx_labels(g,pos,labels=labeldict,font_size=12,
45     font_color='#FFAAAA')
46
47 labeldict = nx.get_node_attributes(g,'out')
48 for i in g.nodes():
49     labeldict[i] = '\n\n\n\n' + str(labeldict[i])
50 nx.draw_networkx_labels(g,pos,labels=labeldict,font_size=12,
51     font_color='#AAFFFF')
52 print("red_number_is_total_traffic_that_enters_the_node.")
53 print("blue_number_is_total_traffic_that_exits_the_node.")
54 plt.show()

```

Code Listing A.4: Gravity Generator Function with Python

```

1 def gravity_generator(graph):
2     tm = np.zeros((n,n))
3
4     for i in range(len(tm)):
5         for j in range(len(tm[i])):
6             if i is not j:
7                 tmp = 0
8                 for k in range(n):
9                     if k is not i:
10                        tmp += graph.nodes[k]['out']
11                        tm[i][j] = int(np.round(graph.nodes()[i]['in'] * (
12                            graph.nodes[j]['out'] / tmp))) # Slide Page 15
13
14     for i in range(len(tm)):
15         if sum(tm[i]) - graph.nodes()[i]['in'] == 1:
16             tm[i][np.argmax(tm[i])] -= 1
17         elif graph.nodes()[i]['in'] - sum(tm[i]) == 1:
18             tm[i][np.argmax(tm[i])] += 1
19     return tm

```

```

18
19 #Create a traffic matrix
20 tm = gravity_generator(g)
21 print(tm)

```

A.3 Unit Test Code

Test code to confirm whether my traffic matrices generation calculation is correct.

Code Listing A.5: Test code for confirming the calculation

```

1 TEST(AppTest, Gravity_Model_3node) {
2     auto *sim = prepareSimulation();
3     auto *mock_qnode = new TestQNode{123};
4     auto *mock_qnode2 = new TestQNode{456};
5     auto *mock_qnode3 = new TestQNode{789};
6
7     auto *app = new AppTestTarget{mock_qnode};
8     setParBool(app, "EndToEndConnection", true);
9     setParInt(app, "num_measure", 1);
10    setParInt(app, "TrafficPattern", 3);
11    setParInt(app, "distant_measure_count", 100);
12    setParInt(app, "LoneInitiatorAddress", 123);
13    setParInt(app, "TrafficIni", 90); //set the amount of
        initiator traffic
14    setParInt(app, "TrafficRes", 0); //set the amount of
        responder traffic
15    app->setName("app");
16    call_private::insertSubmodule(*dynamic_cast<cModule *>(
        mock_qnode), app);
17    sim->registerComponent(mock_qnode);
18    sim->registerComponent(app);
19
20    auto *app2 = new AppTestTarget{mock_qnode2};
21    setParBool(app2, "EndToEndConnection", true);
22    setParInt(app2, "num_measure", 1);
23    setParInt(app2, "TrafficPattern", 3);
24    setParInt(app2, "distant_measure_count", 100);
25    setParInt(app2, "LoneInitiatorAddress", 123);
26    setParInt(app2, "TrafficIni", 0); //set the amount of
        initiator traffic
27    setParInt(app2, "TrafficRes", 60); //set the amount of
        responder traffic
28    app2->setName("app");
29    call_private::insertSubmodule(*dynamic_cast<cModule *>(
        mock_qnode2), app2);
30    sim->registerComponent(mock_qnode2);
31    sim->registerComponent(app2);
32
33    auto *app3 = new AppTestTarget{mock_qnode3};
34    setParBool(app3, "EndToEndConnection", true);
35    setParInt(app3, "num_measure", 1);
36    setParInt(app3, "TrafficPattern", 3);
37    setParInt(app3, "distant_measure_count", 100);
38    setParInt(app3, "LoneInitiatorAddress", 123);

```

```

39  setParInt(app3, "TrafficIni", 0); //set the amount of
    initiator traffic
40  setParInt(app3, "TrafficRes", 30); //set the amount of
    responder traffic
41  app3->setName("app");
42  call_private::insertSubmodule(*dynamic_cast<cModule *>(
    mock_qnode3), app3);
43  sim->registerComponent(mock_qnode3);
44  sim->registerComponent(app3);
45
46  app->callInitialize();
47  app2->callInitialize();
48  app3->callInitialize();
49  ASSERT_EQ(app->getAddress(), 123);
50  ASSERT_EQ(app2->getAddress(), 456);
51  ASSERT_EQ(app3->getAddress(), 789);
52
53  ASSERT_EQ(app->gravityGenerator().size(), 2);
54  ASSERT_EQ(app->gravityGenerator()[0], 60); //Assertion test
    wheather the amount of traffic from node 1 to node 2 is 60
55  ASSERT_EQ(app->gravityGenerator()[1], 30); //Assertion test
    wheather the amount of traffic from node 1 to node 3 is 30
56 }

```
