Leiden University

Computer Science

Waiting Time in Quantum Repeater Chains

Name: Sebastiaan Brand

Date: 06/06/2019

1st supervisor: Dr. David Elkouss (TU Delft)
2nd supervisor: Dr. Alfons Laarman (Leiden University)
Daily supervisor: Tim Coopmans MSc. (TU Delft)

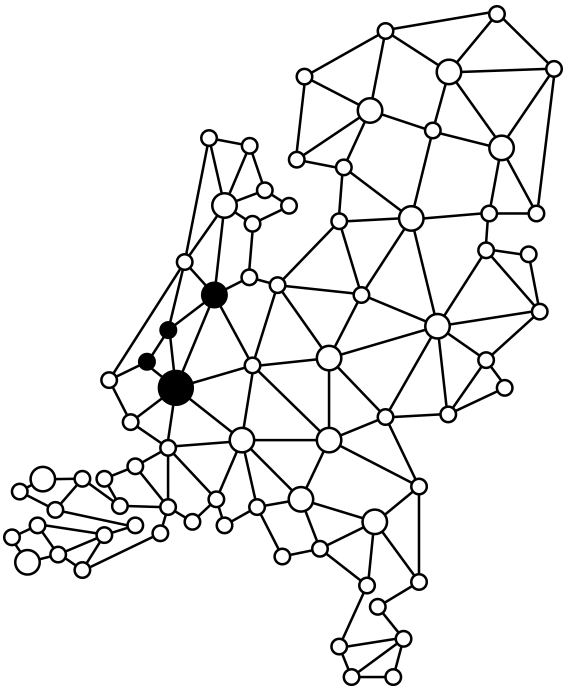
MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Waiting Time in Quantum Repeater Chains

Sebastiaan Brand

May 30, 2019



Supervisors

Dr. David Elkouss	TU Delft
Dr. Alfons Laarman	Leiden University
Tim Coopmans MSc.	TU Delft

Preface

A little over two years ago I took a class in Quantum Computing in Leiden as part of my master's, purely out of curiosity, not knowing that I would be drawn so much to the topic that I would be taking extra classes in Delft, and ultimately even end up doing my master's thesis in this direction.

I want to thank David for his supervision and for giving me the opportunity to do this project, QuTech and the Quantum Internet and Networked Computing theory group for providing a great environment to work in, and Alfons for taking on the supervision of my project from the Leiden side.

I also want to thank the people I was able to share an office with for many months, especially Kenneth and Filip, not just for very helpful discussions (yes, sometimes actual science was discussed), but also for the persistent stream of quantum related puns (of varying quality ;).

Most importantly, I want to thank Tim for having been incredibly helpful in the supervision of this project, putting in a lot of time and effort, and for helping to bring this project to a successful close.

Abstract

Quantum entanglement shared between remote nodes in a quantum network can be used as a resource for numerous new quantum information applications, otherwise impossible on classical networks. However, because of exponential losses in optical fibers, the distance over which this entanglement can be generated directly is limited. Connecting the end nodes indirectly via quantum repeaters aims to solve this problem. Such a chain of quantum repeater nodes can distribute entanglement between its end nodes through probabilistic (but heralded) operations. Because these operations are probabilistic, it is not immediately clear how long one would have to wait for this entanglement, which is the problem we investigate. For this we assume that the entanglement distribution within these repeater chains follows the commonly used BDCZ protocol. Aside from this waiting time, we also investigate the fidelity, or quality, of the produced entanglement.

Previous work on this probabilistic waiting time in repeater chains has largely made use of approximating assumptions to obtain an expression for the average waiting time. Exact solutions to this problem have been found using Markov chains, but because these Markov chains grow exponentially with the number of repeater nodes this approach can only be practically applied to repeater chains of limited size (up to 17 nodes).

We present two methods for characterizing both this waiting time probability distribution as well as the fidelity of the produced entanglement. The first method is a Monte Carlo simulation which can be used to produce approximations of this waiting time distribution and the corresponding fidelities. The second second method is an algorithm for numerically calculating this waiting time distribution, and also compute fidelities under limited circumstances. Both these methods have a time complexity which is polynomial in the number of repeater nodes, and allow for the analysis of repeater chains with hundreds or even thousands of nodes.

Contents

Chapter 1

Introduction

Quantum computation and communication Quantum computers and the ability to manipulate quantum information offer the potential of efficiently solving problems which cannot be efficiently solved on classical computers. This quantum information behaves fundamentally different than classical information. Where classical bits are either 0 *or* 1, quantum bits (qubits) can exist in a superposition of both 0 *and* 1 at the same time. It is also possible to entangle two qubits, which creates a correlation between the qubits states while making it impossible to describe each of the qubits isolated from the other.

Quantum networks are networks which allow for the exchange of quantum information between remote parties, and open the door to numerous new applications, otherwise impossible on classical networks. For one, exchanging quantum states between multiple remote quantum processors enables distributed quantum computation [CEHM99] and secure cloud quantum computing [BFK09]. However there are also network orientated applications which do not require (large) quantum computers to be practically useful, the most well-known one being quantum key distribution [BB84, Eke91], which allows two remote parties to agree on a secret cryptographic key while having high confidence no eavesdropper intercepted this key. Other applications of a quantum network include, but are not limited to, leader election [TKM05] and more accurate synchronization of clocks [KKB$^+$14].

These quantum networks can be physically implemented by providing the ability to generate quantum entanglement between the nodes in this network. Two nodes sharing entanglement can use this entanglement as a resource to transmit arbitrary quantum information between them [BBC$^+$93].

Quantum repeaters A major hurdle in creating this remote entanglement is that it requires the remote qubits to (indirectly) interact. This indirect interaction can in practice be facilitated by exchanging photons, particles of light, between the nodes in the network. The ability to carry quantum information and the speed at which they travel makes these photons well suited for the task quantum communication. However, when these photons are sent through optical fibers the probability of them reaching their destination decreases exponentially with the length of the fiber. Because of this, for increasing distances, it becomes practically unfeasible to connect these network nodes directly using these fibers.

Quantum repeaters aim to solve this problem by dividing a single connection into multiple segments connected via repeater nodes. While different kinds of quantum repeaters have been proposed, here we

focus exclusively on entanglement-based quantum repeaters following the protocol from [BDCZ98]. In these types of repeaters entanglement between neighboring repeater nodes is then generated, after which local operations can connect this entanglement through a process called entanglement swapping, the final result being entanglement between the end nodes of this repeater chain.

Waiting time for entanglement In many implementations of entanglement-based repeaters, the steps in the process of distributing entanglement between two end nodes of a repeater chain are probabilistic, and so also is the time it takes to distribute this entanglement. We are interested in obtaining information about the probability distribution of this waiting time in relation to a set of parameters describing the properties of the repeater chain. In our model these parameters consist of the probabilities that certain basic operations in the repeater chain fail or succeed.

There are two things we hope to obtain from analyzing this waiting time. First is to gain a better insight into how certain parameters affect the waiting time, and especially how this effect scales when moving to longer (say intercontinental) repeater chains. The ability to characterize these waiting time distributions for our model efficiently can also help simulations of more detailed models focus on more narrow, interesting regions of the total parameter space. Secondly we want to use this waiting time information to obtain some information about the fidelity, or quality, of the entanglement the repeater chain produces. This fidelity can decrease over time when generated entanglement is idling, and the waiting time probability distribution can tell us for how long the entanglement idles. The motivation for including fidelity in our investigation is that waiting time alone does not give a complete picture of the performance of a repeater chain. Often, trade-offs between fidelity and waiting time can be made, and so obtaining information about both is useful.

State-of-the-art We have found that analytically evaluating the waiting time even of simple repeater chain models is difficult. Most work on the waiting time for entanglement in repeater chains uses the so-called 3-over-2 approximation for the average waiting time [SDRA$^+$07]. However, methods have been developed which are able to produce exact results using Markov chains [SSvL17]. The computational cost of this approach grows exponentially with the number of nodes in the repeater chain. The authors find that in practice this Markov chain approach can analyze the waiting time behavior of repeater chains with up to 17 nodes. In more recently work [VK19], bounds on the fidelity of the produced entanglement are obtained using a similar Markov Chain approach

Our contributions We present two methods with which we are able to characterize the waiting time probability distributions, as well as the fidelity of the produced entanglement as a function of time.

The first method is a Monte Carlo simulation which produces numerical estimates on the mean waiting time, as well as the complete probability distribution, with an average time complexity polynomial in the number of repeater nodes. This method is also capable of obtaining numerical estimates on the fidelity of produced entanglement as a function of time, while considering the effects of state decay over time, probabilistic entanglement swapping, and (fidelity dependent) probabilistically entanglement distillation.

The second method is a numerical approach with which we are able to numerically compute the waiting time probability distribution exactly on a finite interval, and obtain tight bounds on the mean, with a polynomial time complexity in the the number of repeater nodes. In practice this method is much faster than the Monte Carlo approach and with this method we can produce the waiting time

distribution for repeater chains with thousands of nodes with in the order of minutes of computation time on consumer hardware. With this numerical approach we are also able to calculate fidelities as a function of time while considering the effects of decay over time and deterministic entanglement swapping.

Outline Here we briefly outline the content of the other chapters in this thesis. Chapter 2 covers some general preliminaries, both regarding quantum information as well as probability theory. Chapter 3 explains the problem we investigate in detail, as well as all the assumptions we make for our model, while Chapter 4 gives an overview of relevant literature which also tackles this problem or related instances. Chapter 5 covers three different approaches towards finding the probability distribution of waiting time: First in Section 5.1 we outline a method for obtaining numerical approximations using a Monte Carlo simulation. Then Section 5.2 treats what we can (and cannot) learn about this problem analytically, and after that Section 5.3 describes an exact numerical method which patches the holes in our analytical capabilities. Chapter 6 shows how the calculation or estimation of the output fidelity can be integrated in two of these methods. Finally Chapter 7 summarizes the results, and gives an outlook for potential future work.

Chapter 2

Preliminaries

Before going into more detail regarding our specific repeater chain problem, it is useful to explain and formalize a number of basic concepts in both quantum information, as well as probability theory. These are covered in Sections 2.1 and 2.2 respectively.

2.1 Quantum information

To give a full explanation of why quantum repeaters work the way they do, it is necessary to go into some detail about quantum information in general. While Chapter 3 covers the specifics of quantum repeater chains, this section covers the required quantum information theory leading up to that point.

Qubit states and measurements Similar to a classical bit, a quantum bit, or qubit, can be found in two states when measured. For classical bits we call these states 0 and 1. For qubits we add a bit of notation and call them $|0\rangle$ and $|1\rangle$. Qubits however can also exist in a superposition of $|0\rangle$ *and* $|1\rangle$ at the same time. When a qubit is measured it is forced to give up this superposition and randomly "pick" either $|0\rangle$ or $|1\rangle$. The probability of any given measurement outcome depends on the superposition before the measurement. A qubit can for example be be in an equal superposition of $|0\rangle$ and $|1\rangle$, with both outcomes having probability 50%, but it could also be in a state where it is 95% likely to be measured as $|0\rangle$, and 5% likely to be measured as $|1\rangle$. When a qubit is measured multiple times in a row using the same measurement, the following outcomes will not be probabilistic, but instead the it will same as the first measurement. This means that the act of measuring a qubit in superposition changes, or "collapses", this superposition to a state which is definitively $|0\rangle$ or $|1\rangle$.

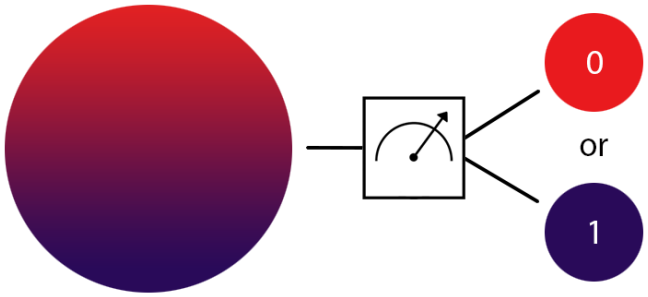


Figure 2.1: *A qubit can exist in a superposition of $|0\rangle$ and $|1\rangle$ at the same time. This superposition collapses when the qubit is measured.*

Entanglement Two or more qubits can, through interaction, become entangled with each other. This entanglement means that the qubits are correlated in such a way that one qubit cannot be completely described isolated from the other. For example, when two qubits A and B are in a super position of $|01\rangle_{AB}$ and $|10\rangle_{AB}$, we know exactly how they are correlated, i.e. if we measure qubit A to be some value, we instantly know that qubit B must have the opposite value. However, writing the states of A and B independently of each other will not give a complete description of the combined system.

This entanglement can, in theory, exist over any distance. In practice, because two qubits require (indirect) interaction to get entangled, it can be difficult to generate this entanglement over long distances. Here we illustrate two remote qubits sharing an entangled state by a (wavy) line connecting them (see Fig. 2.2). Maximally entangled states of two qubits, i.e. entangled states where the qubits are completely (anti)correlated, are often called Bell states or EPR pairs.

Figure 2.2: *When Alice and Bob share a maximally entangled state, neither of them holds any information about their individual qubits, however their measurement outcomes will always be perfectly correlated.*

Quantum communication While it is possible to encode qubit information directly into a photon and then send that photon between two parties, through for example an optical fiber, this is difficult to do in practice. Individual photons are likely to get lost along the way, and if it was carrying the result of a long quantum computation the entire computation needs to be done again. Instead, a quantum communication channel can be implemented using a scheme called quantum teleportation [BBC$^+$93], where two parties use a shared EPR pair to teleport an arbitrary qubit state between them. Generating these shared EPR pairs still requires photons to be sent between the two parties, but these photons are more easily generated than the information which is to be transmitted, and hence trying again after losing them is less of a problem. The generated EPR pairs are used up when they are used to teleport quantum states, so they need to be constantly regenerated.

2.2 Probability theory

Our repeater chain model, outlined in Section 3.3, is build up out of probabilistic processes. This section covers a number of definitions and lemmas needed to describe and analyze these processes.

Discrete random variables A discrete random variable is a variable which can take on values from a totally ordered countable set. All the discrete random variables we discuss take on non-negative integer values, that is from $\mathbb{N}_{>0} = \{1, 2, 3, \ldots\}$, which we will call *count random variables* and in our case these will be counting the number of time steps until successful entanglement distribution. The probability that a random variable X takes on some value is described by its probability function (Definitions 2.1 and 2.2). The set of values a random variable can take on we call its support (Def. 2.3).

Definition 2.1. *The probability mass function (PMF) of some discrete random variable X is defined as the probability that X takes on the value x. We denote*

$$\Pr(X = x).$$

Definition 2.2. *The cumulative distribution function (CDF) of some random variable X is defined as the probability that X takes on a value smaller or equal to x. We denote*

$$\Pr(X \leq x).$$

Corollary 2.2.1. *For a count random variable X, we can translate between its probability mass function and its cumulative distribution function in the following manner:*

$$\Pr(X = x) = \Pr(X \leq x) - \Pr(X \leq x - 1)$$

and

$$\Pr(X \leq x) = \sum_{k=1}^{x} \Pr(X = k).$$

Definition 2.3. *In general we define the support R_X of a discrete random variable X as the set of elements x for which $\Pr(X = x) > 0$, i.e.*

$$R_X = \{x \in \mathbb{R} : \Pr(X = x) > 0\}.$$

The random variables we investigate count discrete events, and only take on values in $\mathbb{N}_{>0} = \{1, 2, 3, \ldots\}$. For these random variables the support can also be defined as

$$R_X = \{x \in \mathbb{N}_{>0} : \Pr(X = x) > 0\}.$$

Definition 2.4. *The expected value, also called the expectation, the mean, or the average, of a discrete random variable X is defined as*

$$\mathrm{E}[X] = \sum_{x \in R_X} x \cdot \Pr(X = x)$$

where R_X is the support (Def. 2.3) of X.

Lemma 2.5. *The expected value of a count random variable X can be computed by summing over its complementary cumulative distribution function:*

$$\mathrm{E}[X] = \sum_{x=1}^{\infty} \Pr(X \geq x).$$

Proof.

$$\begin{aligned}
\mathrm{E}[X] &= \sum_{k=1}^{\infty} k \cdot \Pr(X = k) \\
&= \sum_{k=1}^{\infty} \sum_{j=1}^{k} \Pr(X = k) \\
&= \Pr(X = 1) + \\
&\quad \Pr(X = 2) + \Pr(X = 2) + \\
&\quad \Pr(X = 3) + \Pr(X = 3) + \Pr(X = 3) + \\
&\qquad \vdots
\end{aligned}$$

The terms in this nested sum form a lower triangular matrix, and if we sum them over the columns first instead of the rows we get:

$$\begin{aligned}
&= \sum_{j=1}^{\infty} \sum_{k=j}^{\infty} \Pr(X = k) \\
&= \sum_{j=1}^{\infty} \Pr(X \geq j)
\end{aligned}$$

□

Geometric random variables A number of processes in our repeater chain model are described by geometric distributions. A geometric distribution is the probability distribution of the number of consecutive Bernoulli trials needed until the first success. These Bernoulli trials can be seen as coin flips which land on the desired side with a fixed probability p. In Definition 2.6 we define a geometric distribution as the *total number* of Bernoulli trials until the first success. Note that this distribution sometimes is defined as the *number of failures* instead, which would be the total number of trials minus 1.

Definition 2.6. *We define a geometric distribution as the number of independent, identically distributed (i.i.d.) Bernoulli trials (coin flips) needed until the first success. For a random variable X which is geometrically distributed we write*

$$X \sim \mathrm{geom}(p),$$

where p is the success probability of a single Bernoulli trial. Because for the first success at least one trial is needed, the support of X is given by

$$R_X = \{1, 2, 3, \ldots\} = \mathbb{N}_{>0}.$$

For $x \in \mathbb{N}_{>0}$, the probability mass function of X is given by

$$\Pr(X = x) = (1 - p)^{x-1} p,$$

and the cumulative distribution function of X is given by

$$\Pr(X \leq x) = 1 - (1 - p)^x.$$

The expected value of X is

$$\mathrm{E}[X] = \frac{1}{p}.$$

Sums of random variables When looking at waiting time, it is possible for multiple random processes to need to run sequentially. In order to learn something about the probability distribution of when the last of these sequential processes finishes we look at sums of random variables. Lemma 2.7 shows how the PMF of the sum of two independent count random variables can be computed. Lemma 2.8 shows how the PMF can be computed for the sum of a fixed number of i.i.d. random variables, while Lemma 2.9 shows how the PMF can be computed if the length of this sum is also a random variable.

Lemma 2.7. *Given $Z = X + Y$, with X and Y independent, count random variables, the probability mass function of Z is given by the convolution of the probability mass functions of X and Y:*

$$\Pr(Z = z) = \sum_{y=0}^{z} \Pr(X = z - y) \Pr(Y = y).$$

Lemma 2.8. *Given $Z_n = X_1 + X_2 + \cdots + X_n$, with X_k i.i.d. count random variables, and n some fixed value in $\mathbb{N}_{>0}$, the probability mass function of Z_n can be computed iteratively through repeated convolutions (Lem. 2.7), starting with*

$$\Pr(Z_1 = z) = \Pr(X = z)$$

for $n = 1$, and

$$\Pr(Z_n = z) = \sum_{j=0}^{z} \Pr(Z_{n-1} = z - j) \Pr(Z_1 = j)$$

for $n \geq 2$.

Proof.

$$\begin{aligned} \Pr(Z_n = z) &= \Pr(Z_{n-1} + Z_1 = z) && \text{using Lem. 2.7:} \\ &= \sum_{j=0}^{z} \Pr(Z_{n-1} = z - j) \Pr(Z_1 = j) \end{aligned}$$

□

Lemma 2.9. *Given $Z = X_1 + X_2 + \cdots + X_N$, with X_k i.i.d. count random variables, and N a discrete random variable with support $\mathbb{N}_{>0}$, independent of X_k, the probability mass function of Z is given by*

$$\Pr(Z = z) = \sum_{n=1}^{z} \Pr(N = n) \Pr(Z_n = z)$$

where Z_n the sum over a fixed number of X_k as in Lemma 2.8. This sum over the possible values of N only needs to range from 1 to z because for $n < 1$ we have

$$\Pr(N = n) = 0$$

since the support of N is $\mathbb{N}_{>0}$, and for $n > z$ we have

$$\Pr(Z_n = z) = 0$$

because the support of X_k is $\mathbb{N}_{>0}$ and therefor the lowest value Z_n can take on is n.

Maximum of random variables When looking at waiting time, it is possible for multiple random processes to run in parallel, and we might be interested in the probability distribution of when all of these processes have finished. To this end we look at the properties of the maximum of multiple independent random variables. Lemma 2.10 shows how the probability distribution of the maximum of n independent random variables can be computed from their cumulative distribution functions, while Lemma 2.12 shows how of the maximum of two random variables can be stochastic ordered (see Def. 2.11) and how its mean can be bound.

Lemma 2.10. *Given $Z = \max\{X_1, \ldots, X_n\}$, with all X_k independent random variables, the CDF of Z is given by the product of the CDF's of all X_k:*

$$\Pr(Z \leq z) = \Pr(X_1 \leq z) \Pr(X_2 \leq z) \ldots \Pr(X_n \leq z).$$

Proof.

$$\begin{aligned} \Pr(Z \leq z) &= \Pr(\max\{X_1, \ldots, X_n\} \leq z) \\ &= \Pr(X_1 \leq z, X_2 \leq z, \ldots X_n \leq z) && \text{all } X_k \text{ independent:} \\ &= \Pr(X_1 \leq z) \Pr(X_2 \leq z) \ldots \Pr(X_n \leq z) \end{aligned}$$

□

Corollary 2.10.1. *When all X_k in Lem. 2.10 are, aside from independent, also identically distributed, the result becomes:*

$$\Pr(Z \leq z) = \Pr(X \leq z)^n .$$

Definition 2.11. *For two random variables X and Y, we call X stochastically greater than Y if*

$$\forall k, \Pr(Y > k) \leq \Pr(X > k).$$

We can also write $Y \leq_{\text{st}} X$.

Lemma 2.12. *The maximum of two non-negative, i.i.d. random variables X_k can be stochastically ordered as*

$$X_1 \leq_{\text{st}} \max\{X_2, X_3\} \leq_{\text{st}} X_4 + X_5,$$

i.e.

$$\forall x, \Pr(X_1 > x) \leq \Pr(\max\{X_2, X_3\} > x) \leq \Pr(X_4 + X_5 > x).$$

Proof. First the left inequality:

$$\begin{aligned} \Pr(\max\{X_2, X_3\} > x) &= 1 - \Pr(\max\{X_2, X_3\} \leq x) \\ &= 1 - \Pr(X \leq x)^2 && \text{(using Corollary 2.10.1)} \\ \Pr(X_1 > x) &= 1 - \Pr(X \leq x) \end{aligned}$$

Since $\forall x, 0 \leq \Pr(X \leq x) \leq 1$, we have that

$$\forall x, \Pr(X \leq x) \geq \Pr(X \leq x)^2,$$

and so

$$\forall x, \Pr(X_1 > x) \leq \Pr(\max\{X_2, X_3\} > x).$$

For the inequality on the right, consider two sets $S_{\max}$ and S_{sum} of tuples (a, b), for a and b being elements in the support R_X of X.

$$S_{\max}(x) = \{(a, b) : a, b \in R_X, \max\{a, b\} > x\}$$
$$S_{\text{sum}}(x) = \{(a, b) : a, b \in R_X, a + b > x\}$$

Since all the elements in the support of X are non-negative, for any tuple for which $\max\{a, b\} > x$ holds, $a + b > x$ will also hold, so

$$S_{\max}(x) \subseteq S_{\text{sum}}(x).$$

We can compute the probabilities in the inequality like

$$\Pr(\max\{X_2, X_3\} > x) = \sum_{(a,b)\in S_{\max}} \Pr(X_2 = a) \Pr(X_3 = b),$$
$$\Pr(X_4 + X_5 > x) = \sum_{(a,b)\in S_{\text{sum}}} \Pr(X_4 = a) \Pr(X_5 = b).$$

And since $\Pr(X_2 = a) = \Pr(X_4 = a)$, and $\Pr(X_3 = b) = \Pr(X_5 = b)$, and $S_{\max} \subseteq S_{\text{sum}}$ we have that

$$\Pr(\max\{X_2, X_3\} > x) \leq \Pr(X_4 + X_5 > x).$$

□

Corollary 2.12.1. *From Lemma 2.12 follows, using Lemma 2.5, that*

$$\mathrm{E}[X_1] \leq \mathrm{E}[\max\{X_2, X_3\}] \leq \mathrm{E}[X_4 + X_5].$$

Chapter 3

Problem specification

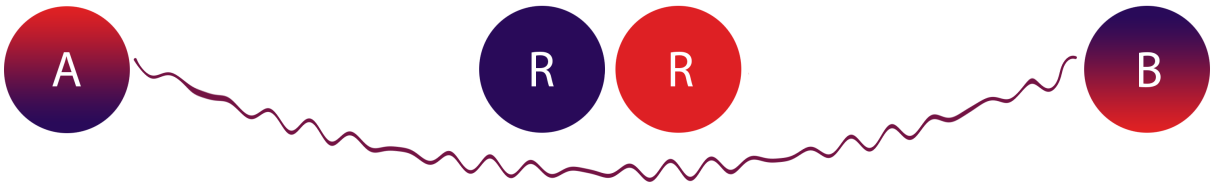
This section defines the exact problem we investigate, i.e. finding waiting time probability distributions for entanglement distribution in quantum repeater chains. Section 3.1 explains how a single quantum repeater can be used to generate entanglement between two nodes, after which Section 3.2 outlines a protocol [BDCZ98] which can be used to create this end-to-end entanglement when many intermediate repeaters are involved. Finally, in Section 3.3 we define the waiting for entanglement distribution in a repeater chain by making specific assumptions about the hardware on which the protocol in Section 3.2 would run.

3.1 Quantum repeaters

A classical network, e.g. the current internet, is a collection of classical machines which can exchange classical information between themselves. Similarly, a quantum network is a collection machines which can exchange quantum information. Quantum information between two nodes in the network can be exchanged via quantum teleportation [BBC$^+$93] if these nodes share entanglement. This shared entanglement is generally generated by exchanging photons between the nodes through optical fibers. Because of exponential losses of photons in optical fibers it is not feasible to connect two distant quantum nodes directly. A solution to this problem is to divide the communication channel between these nodes into multiple segments, connected to each other via quantum repeaters. These quantum repeaters serve as intermediate nodes between the end nodes which are to be connected. Entanglement is then generated between the neighboring nodes in this chain of repeaters. Figure 3.1a shows this for a repeater chain with one repeater node. In order for these repeaters to improve the probability that two end nodes successfully generate entanglement in a given amount of time, it is necessary to be able to store the generated entangled states over a period of time. This means that not all the entanglement covering the entire repeater chain needs to be generated at the same time. Being allowed to generate some entanglement on the first attempt, some more on the next, etc., decreases the overall number of attempts required compared to a direct channel where the probability of success decreases exponentially with the distance.

Connecting all the nodes individually does not yet constitute a connection between the end nodes. This individual entanglement can be tied together by means of entanglement swapping, also called entanglement teleportation, as depicted in Figure 3.1b. The quantum part of these swaps can be executed locally on the nodes, however still requiring some classical communication between the repeater and its neighbors.

(a) *The qubits in the network nodes of Alice (A) and Bob (B) first generate entanglement with the repeater node (containing two qubits) in the middle.*



(b) *A local operation on the repeater node allows the entanglement to be "swapped" such that Alice and Bob are now entangled with each other. Classical communication is still required for this operation.*

Figure 3.1: *A quantum repeater connecting Alice and Bob.*

For many physical implementations of repeaters, like [DLCZ01] where qubits interact using linear optics, neither generating entanglement between neighboring nodes nor performing swaps are deterministic operations. For the entanglement generation this means it is necessary to keep trying until there is a success. What we are interested in is (the probability distribution of) the number of required attempts. For the entanglement swapping, aside from the number of swap attempts, there is an additional thing to take into account: if a swap fails all involved entanglement is lost, and before the swap can be attempted again, the entanglement needs to be regenerated. We assume the successful events for both the entanglement generation, as well as the entanglement swapping are heralded. This means that even though the operations succeed probabilistically, it is announced (and known to the repeater protocol) when they do succeed.

Because of the probabilistic nature of this system, it is not immediately clear how long it will take before entanglement between the end nodes is generated. We are interested in characterizing the probability distribution describing this waiting time, or latency, as a function of the number of repeater segments, as well as the probability that entanglement generation between neighbors succeeds, and the probability an entanglement swap succeeds. To that end we specify next how this entanglement swapping works in longer repeater chains.

3.2 The BDCZ protocol

The idea of generating entanglement and performing entanglement swapping as shown in Figure 3.1 could, in theory, simply be extended to repeater chains of arbitrary numbers of nodes: first generate entanglement between all neighboring nodes, then perform swaps on all the repeaters. It would not really matter in which order the swaps happen, as long as there is some coordination with regard to the classical communication.

There is however a problem with this in practice. Because none of the physical operations are perfect, the quality, or fidelity, of the entanglement link resulting from a swap (Fig. 3.1b) will be lower than either of the fidelities of the original two links (Fig. 3.1a), and when there are many repeater nodes these imperfections would stack up quickly. It is possible to create a single higher fidelity link from two or more lower fidelity links. This process is called entanglement distillation, and is also probabilistic. Just like with entanglement swapping, if a distillation attempt fails, all involved entanglement is lost. The success probability of this entanglement distillation depends on the fidelity of the input links. If these fidelities are too low, it becomes impossible to distill entanglement of useful quality. For long repeater chains the fidelity of the final entanglement would be far too low to be usable for distillation.

A solution to this problem is provided by the repeater chain protocol proposed in [BDCZ98, DBCZ99]. Here it is proposed to perform the entanglement swaps in a nested structure. Figure 3.2 gives an overview of this structure. Layer 0 in this structure represents the physically neighboring repeater nodes, between which entanglement is generated directly. Entanglement swapping, just as in Figure 3.1, is performed on every second node, with the result that the nodes as in layer 1 are now "neighbors" in the sense that they share entanglement. The repeater chain on layer 1 behaves very similar to the repeater chain at layer 0, except now generating links between "neighbors" is a more complicated process. From this a recursive structure can be recognized , shown in Figure 3.3. This recursive structure continues until finally the end nodes are entangled directly. For simplicity we assume that the repeater chain has exactly 2^n segments, for some positive integer n, and entanglement swapping is always performed on pairs of two links in every transition from layer k to layer $k+1$.

The reason for doing the swaps in this nested structure is to allow for entanglement distillation steps to happen between the different nesting layers of the protocol, in order to keep the fidelity of the links high enough for them to remain of sufficient quality. Figure 3.4 shows how these distillation steps are integrated in the nested swapping structure. For every link to be used in a swap, two or more links are first generated and distilled. Entanglement swapping is then done using the the distilled, higher fidelity links, which results in a new link with lower fidelity. This is done for all the swaps happening on all the nesting layers of the protocol, keeping the entanglement fidelity from becoming too low.

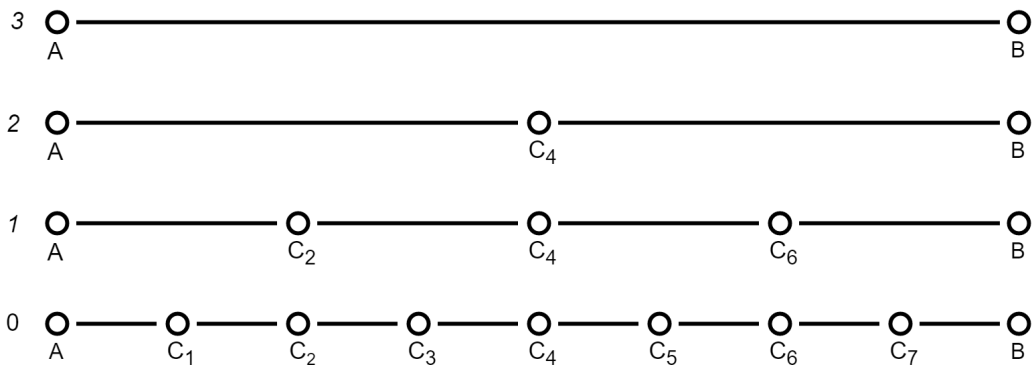


Figure 3.2: *An overview of the swapping in the protocol for a repeater chain with $2^3 = 8$ segments. The numbers on the left indicate the nesting layers. At layer 0 the nodes (e.g. A and C_1) are physical neighbors. After C_1 has successfully generated entanglement with its neighbors A and C_2, an entanglement swap is performed by node C_1, which (if successful) results in nodes A and C_2 now being entangled with each other. This process continues for all the nodes, on all the layers, until eventually A and B share entanglement.*

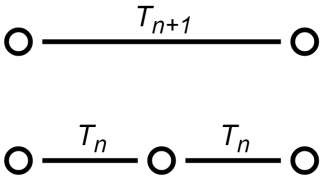


Figure 3.3: *The recursive structure present in the BDCZ protocol. The waiting time for entanglement at layer $n+1$ can be described as a function of the waiting at layer n.*

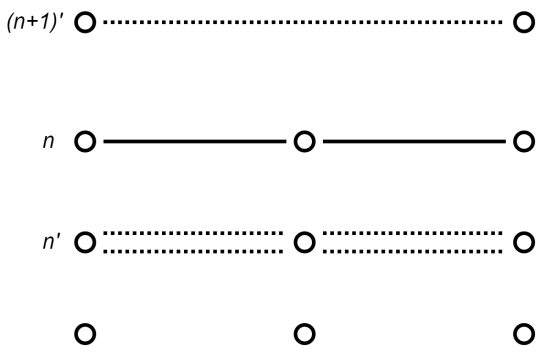


Figure 3.4: *The distillation steps happen between the swap steps in the protocol. In this case two links with lower fidelity (dotted) are first generated (n') and then probabilistically distilled into a single higher fidelity link (n). The obtained link after the swap, $(n+1)'$, has again a lower fidelity.*

3.3 Waiting time model

As mentioned before, we are interested in the waiting time for distributing entanglement between the end nodes of a repeater chain. Towards solving this problem we first specify a basic model of the BDCZ protocol running on physical hardware. after which we show an extension to this model in the form of entanglement timeouts.

3.3.1 Basic model

As a basis for our waiting time model we take the nested entanglement swapping structure of the BDCZ protocol covered in Section 3.2, where swaps are performed on 2 segment at each time. For simplicity we limit the number of repeater chain segments to some integer power of 2. We assume a single entanglement generation attempt between neighbors succeeds with probability p_{gen}, which is set beforehand and remains constant during the execution of the protocol. The same goes for the probability that a swap succeeds, which we call p_{swap}, and the probability a distillation attempt succeeds p_{dist}. Table 3.1 gives an overview of these parameters. We also assume that the repeater nodes have the ability to generate links to both neighbors in parallel, as opposed to needing to generate them one after the other.

Because in our model the nested structure of distillation is exactly the same as that for entanglement swapping, distillation does not fundamentally add any complexity to the problem. In fact when doing m distillation rounds for every layer of swaps (a total of n layers), the problem of finding the waiting time could be rephrased as doing no distillation rounds and having $(m+1) \times n$ layers of swaps. Because of this we generally assume the number of distillation rounds $m = 0$ in the analysis of this problem.

number of equal size repeater chain segments	$N = 2^n$
number of swap layers	n
entanglement generation probability	p_{gen}
swap success probability	p_{swap}
distillation success probability	p_{dist}
number of distillation rounds per swap layer	m

Table 3.1: *Overview of the model*

When it comes to the waiting time of generating entanglement we count in discrete number of timesteps. On each repeater segment a single entanglement generation attempt can be made at each timestep. This effectively means that each discrete timestep corresponds to the amount of time a single heralded entanglement generation attempt takes: L_0/c, where L_0 is the distance between two neighboring repeater nodes and c is the speed with which they can exchange information (i.e. the speed of light). We assume entanglement swapping is done instantly when both the required links are available, and does not take any time. By doing this we are ignoring the classical communication time between nodes required for a swap operation. This is a valid approximation when p_{gen} is small (more specifically $p_{\text{gen}}^{-1} \gg n$ [BS08], see also Section 4.1), which it often is in practice, however our main reason for leaving out this communication time is so that we can streamline the problem to a minimal problem statement which still captures the core difficulties of this problem.

From these assumptions we can extract a recursive definition of the waiting time for such repeater chains. This recursive recursive definition is given in Definition 3.3.1. Below follows a more detailed motivation of this definition.

Let the random variable T_n be the waiting time for a repeater chain with 2^n segments. Let us first look at T_0, the waiting time for a repeater chain with $2^0 = 1$ segment, so no repeater nodes. The probability of generating entanglement between the end nodes in this repeater chains on the first attempt is p_{gen}. The probability of the first success to be on the second attempt (and therefore not on the first) is $(1 - p_{\text{gen}})p_{\text{gen}}$. For the first success to be on the third attempt we get $(1 - p_{\text{gen}})^2 p_{\text{gen}}$, etc. We can see that T_0 is geometrically distributed with parameter p_{gen}.

Let the random variable T'_k describe the waiting time needed in order to have the entanglement ready to perform a swap operation to move from layer k to layer $k+1$. Under the assumption that both required links can be generated in parallel, T'_k is the maximum of two random variables giving the waiting time on layer k, T_k. Once the entanglement for the swap is ready the swap needs to be done. As specified before, this is an operation which succeeds with probability p_{swap} at each attempt. Just like for T_0, the number of times we need to try a specific swap in the protocol before it succeeds is a geometric random variable, this time with parameter p_{swap}. When the swaps fails the involved entanglement needs to be regenerated, and so the waiting times are added.

Definition 3.3.1. *Let T_n denote the waiting time for a repeater chain with 2^n segments, entanglement success probability p_{gen}, swap success probability p_{swap}, and no distillation. T_n has the following recursive definition:*

$$\begin{aligned} T_{n+1} &= (T'_n)_1 + (T'_n)_2 + \cdots + (T'_n)_S \\ T'_n &= \max\{(T_n)_1, (T_n)_2\} \\ T_0 &\sim \text{geom}(p_{\text{gen}}) \\ S &\sim \text{geom}(p_{\text{swap}}) \end{aligned}$$

3.3.2 Entanglement timeouts

In the model described in the previous section, one assumption we make is that entanglement can be stored for however long is necessary while waiting for other parts of the protocol to finish. In reality however, the storage of these entangled states is not perfect, and the quality of the generated entanglement reduces over time. To limit the effect of this decay, entanglement timeouts, or cut-offs, can be introduced [RGR+18]. What this means is that when an earlier generated link has been stored in memory for a while we assume its fidelity has decreased to the point where it is not useful anymore, and throw the link away and generate a new one from scratch.

Fixed timeout bins A simple way to model entanglement timeouts is with fixed bins. Intuitively, one could think of this as starting a timer, and needing to generate two entanglement links before the timer reaches some predetermined cut-off value τ. If both links have been generated before this time a swap is attempted when the cut-off time has been reached. If either one or both links are missing everything is discarded and restarted from scratch. Figure 3.5 illustrates this idea. Waiting to do the swap at time τ is not the most efficient with respect to waiting time or the quality of the state because we potentially spend a lot of time idling, but it makes it easier to deal with analytically, as we will see in Section 5.2.3.

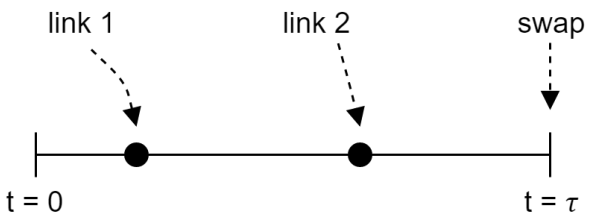


Figure 3.5: *Fixed bin entanglement timeouts. Two links are attempted to be generated before some time τ. At time τ, if both links have been generated, an entanglement swap is attempted. If one (or both) of the links are missing, all the current entanglement is discarded and the process starts from scratch. This guarantees that links which have been generated will never spend more than τ time steps in memory waiting for other links.*

3.4 Fidelity model

Aside from computing waiting times, we would also like to predict the fidelity, or quality, of the entanglement produced by the repeater chain. In order to model the fidelity of the entanglement links we make two assumptions. First we assume all our two-qubit states are Werner states, and secondly we assume all our noise is depolarizing noise. A more detailed description and motivation of these assumptions can be found in Appendix A. From these assumptions we create three functions for updating fidelities.

- swap(F_1, F_2) gives the fidelity after a successful swap on two links with fidelities F_1 and F_2, according to Lemma A.4.
- distill(F_1, F_2) gives the fidelity after a successful distillation attempt on two links with fidelities F_1 and F_2, as well as the probability of success, according to Lemma A.3.
- decay$(F, \Delta t)$ gives the fidelity after a link with initial fidelity F has spent Δt timesteps in memory, by applying a depolarizing channel with parameter $p = e^{-\lambda \Delta t}$ according to Lemma A.6. Here λ^{-1} is the mean lifetime and needs to be set as predetermined parameter to the model.

In Chapter 6 we show how these functions can be used to calculate the average fidelity of produced entanglement, conditioned on the generation time.

Chapter 4

Literature overview

The original proposal of the BDCZ protocol also includes an analysis of the amount of time needed to entangle the end nodes of a repeater chain [BDCZ98, DBCZ99]. This analysis relies on a number of simplifying assumptions and serves mostly to show how different types of entanglement distillation schemes behave in terms of generation time as well as required resources. Since then, there have been numerous works on repeater chains which include an analysis of the end-to-end entanglement rate. Often an approximation for the average waiting time is used, which is discussed in detail in Section 4.1. However exact results have been produced, both for the model we define in Section 3.3.1, as well as for variations, which are discussed in Sections 4.2 and 4.3 respectively.

4.1 3-over-2 approximation

The 3-over-2 approximation for the average generation time, first appearing in [SDRA$^+$07], has been used in the analysis of entanglement generation rate for quantum repeaters by many [SDRA$^+$07, SSM$^+$07, SSZ$^+$08, BS08, SDS09, SSDRG11, BPvL11, MATN15, ALW$^+$17]. The following is the 3-over-2 approximation stated in terms of the parameters defined for the model in Section 3.3.1.

$$\mathrm{E}\left[T_n\right] \approx \frac{1}{p_{\mathrm{gen}}} \cdot \left(\frac{3}{2p_{\mathrm{swap}}}\right)^n \tag{4.1}$$

Model This approximation is used for models similar to the model we describe in Section 3.3.1, with repeater chains with 2^n segments and probabilistic entanglement generation and swapping. These models also take the entanglement generation and swapping probabilities to be fixed, although often allow for the swapping probability to be different for different nesting levels. The main difference is that these models take the communication time for entanglement swapping into account. This communication time increases for swaps at higher nesting levels of the protocol because the repeater nodes involved in the swaps will be further apart, and the speed at which they can communicate over distance is limited by the speed of light (c). However, as we will see in the next paragraph, under certain approximating conditions this communication time on higher nesting levels can be neglected, leaving only the communication time between neighboring nodes. When this communication time between neighbors (L_0/c) (plus the amount of time it takes to prepare and measure states) is set to 1, the analysis becomes the same as for the random variable given in Definition 3.3.1.

Approximation In order to obtain the result in Eq. (4.1) two approximating assumptions are used. First of all, as stated in [BS08], the communication time for entanglement swapping can be neglected if $p_{\text{gen}}^{-1} \gg n$. The intuition here is that when the average amount of attempts to generate a single link on the lowest nesting level ($\mathrm{E}[T_0] = p_{\text{gen}}^{-1}$) is much higher than the number of swaps that link will be involved in (at most n), the time spent on these swaps becomes insignificant compared to the time spent on generating the elementary links. The subsequent analysis of the average waiting time in most of these works is similar and can be summarized as

$$\mathrm{E}[T_n] \approx \frac{1}{p_{\text{gen}}} \cdot \left(\frac{1}{p_{\text{swap}}}\right)^n \cdot \nu_0 \nu_1 \cdots \nu_{n-1} \tag{4.2}$$

where

$$\nu_k = \frac{\mathrm{E}[\max\{(T_k)_1, (T_k)_2\}]}{\mathrm{E}[T_k]}.$$

This result can also be obtained by expanding the recursive expression for T_n found in Definition 3.3.1. Since T_k are non-negative random variables we get

$$\mathrm{E}[T_k] \leq \mathrm{E}[\max\{(T_k)_1, (T_k)_2\}] \leq \mathrm{E}[(T_k)_1 + (T_k)_2]$$

and so, as also mentioned in [SSDRG11], for all ν_k we have

$$1 \leq \nu_k \leq 2.$$

The second approximation is made when setting the values of ν_k. Since $T_0 \sim \text{geom}(p_{\text{gen}})$, for ν_0 we have that when $p_{\text{gen}} \to 0$, $\nu_0 = \frac{3}{2}$. Although ν_1 has been computed analytically [SSvL17], no general expression for ν_k is known. However empirical results [JTL07] suggest that, in the regime of small p_{gen} and p_{swap}, setting all $\nu_k = \frac{3}{2}$ is a good approximation. Setting all $\nu_k = \frac{3}{2}$ in Eq. (4.2) produces the approximation given in Eq. (4.1).

Limitations While the approximation in Eq. (4.1) is good for small p_{gen} and p_{swap}, it has also been shown in [SSvL17] that, in comparison with exact results, for larger p_{gen}, p_{swap}, or n this approximation becomes increasingly worse (up to a factor 5 too high for $p_{\text{gen}} = p_{\text{swap}} = 1$ and $2^n = 16$). Currently, in many physical implementations these success probabilities are low, making this approximation applicable. However, physical implementations of quantum repeaters where these probabilities are higher are possible. A method allowing for near-deterministic entanglement swapping using linear optics is presented in [BRP+10], and while solutions have been obtained in the case of deterministic swapping (see Section 4.3), it might also be desirable to have accurate solutions for systems which are still probabilistic, but which have higher success probabilities for operations.

4.2 Exact results with Markov chains

Exact results for waiting times have been obtained by modeling the random process of entanglement generation and swapping using Markov chains [SSvL17].

Markov chains are used to model random processes and are often described by directed graphs. The vertices in these graphs represent the state of the system, in this case the state of the repeater chain, while directed weighted edges describe the probability of transitioning from one state to another. Often there is one state (or possibly multiple states) which represent the random process being finished, and

has no outgoing edges. Such a state is called an absorbing state. In [SSvL17] the states making up the Markov chain capture which repeaters share entangled pairs, with the state where the end nodes of the repeater chain share entanglement as the absorbing state of the Markov chain. These are discrete-time Markov chains, which means that transitioning from one state to a directly connected state takes exactly one time step.

From such a Markov chain the waiting time probability distribution can be extracted, as well as the mean and higher order moments. Closed form expressions for the mean waiting time for up to 4 repeater segments are obtained, while numerically the mean and probability distribution can be found for up to 16 segments.

The number of vertices $|V|$ in these Markov chains (and therefore also the computational complexity) grows exponentially ($|V| = 2^N$) with the number of repeater segments $N = 2^n$. Exploiting symmetries and lumping together equivalent states reduces the size of the Markov chains but the growth remains exponential in the number of repeater segments ($|V| \approx 2 \cdot 1.346^N$). This exponential growth makes this approach practically unsuitable for analyzing waiting time for longer repeater chains.

More recent work [VK19] shows how the complexity of computing the probabilities $P(T_n = t)$ can be improved by a factor $|V|$ by computing these via the corresponding so-called probability generating function rather than directly from the Markov matrix, but this method still remains exponential in the number of repeater segments N. In addition, they are also able to produce tighter bounds on secrete key rates (a function of both waiting time and fidelity) than were previously available. Their methods for obtaining bounds on fidelity can be summarized as follows: First upper bounds on the waiting time for each link are obtained, and then using this waiting time the average fidelity for each layer are computed under very similar assumptions we make (see Appendix A) with regard to how operations and decay over time changes the fidelity.

4.3 Different models

Variations on the model described in Section 3.3.1 have been investigated which simplify the analysis of the average waiting time. Here we take a look at two of them.

Deterministic swapping When analyzing a model in which entanglement swapping succeeds deterministically and takes no time, and distillation is omitted, finding the average waiting time reduces to finding the mean of $N = 2^n$ i.i.d. geometric random variables. The following expression for this average waiting time is obtained in [BPvL11]:

$$\mathrm{E}\left[T_n^{\mathrm{det}}\right] = \sum_{k=1}^{N} \binom{N}{k} \frac{(-1)^{k+1}}{1-(1-p_{\mathrm{gen}})^k}. \tag{4.3}$$

Expressions which reduce to Eq. (4.3) are also obtained in [VK17, Eq.(5)] and in [KMSD19, Eq.(7)]. This expression can also be tightly bounded by expressions of which the behavior for large N is easier to analyze (see Section 5.2.2, Eq. (5.5)). The probability distribution of T_n^{det}, found in [KMSD19], is given by

$$\Pr\left(T_n^{\mathrm{det}} = t\right) = \left(1-(1-p_{\mathrm{gen}})^t\right)^N - \left(1-(1-p)^{t-1}\right)^N. \tag{4.4}$$

Entanglement timeouts In [SJM18] it has been shown that it is possible to introduce entanglement timeouts to the protocol, i.e. discarding entangled states after a certain period of time, in such a way that the analysis of the average waiting time can be done completely analytically. These timeouts serve to limit the amount of time generated entangled states can spend in memory, and so also the amount their fidelity decreases over time, but discarding entanglement also has an effect on the waiting time.

The entanglement timeouts used in [SJM18] are similar to the fixed timeout model described in Section 3.3.2. It differs from these fixed timeouts in that the number of steps τ_n before a timeouts happens at level n is a function of the output times of level $n-1$, rather than being fixed to the same value for every level. Because of these timeouts the probability of producing an entangled pair at nesting layer n is given by

$$p_{\text{out}}^{(n)} = p_{\text{swap}} \cdot \left(1 - \left(1 - p_{\text{out}}^{(n-1)}\right)^{\tau_n}\right)^2 \tag{4.5}$$

where $p_{\text{out}}^{(0)} = p_{\text{gen}}$. From this it follows that the average time to generate entanglement while using these timeouts is given by

$$\mathrm{E}\left[T_n^{\text{timeout}}\right] = \frac{1}{p_{\text{out}}^{(n)}} \tau_n \tau_{n-1} \cdots \tau_0 \tag{4.6}$$

In their comparison with the BDCZ protocol without timeouts they again use an expression equivalent to the 3-over-2 approximation given in Eq. (4.1).

Chapter 5

Results: waiting time

In order to characterize the waiting time distribution we take three different approaches. The first approach, presented in Section 5.1, uses a Monte Carlo simulation and is the most flexible when it comes to handling changes or extensions to the repeater chain model. The drawback of this approach is that by simulating the random processes in the repeater chain directly, it lends little insight into how the ultimate waiting time distribution comes to be, and how we might calculate it more efficiently. In Section 5.2 we describe an analytical approach which, on its own, does not result in a complete solution to our problem, but does however provide a foundation for the numerical approach, which is presented in Section 5.3.

5.1 Monte Carlo simulation

A straightforward way of characterizing the waiting time random variable as defined in Definition 3.3.1 is to generate samples from this random variable, and from these samples extract a numerical approximation of the probability distribution. Generating these samples can be done by means of a Monte Carlo simulation. In general a Monte Carlo simulation starts by generating random samples from a known distribution, and then performs some (deterministic) operations on these samples, producing some output. For us the random inputs are the initial waiting times of generating links between neighbors, as well as coin flips determining when a swap is successful. The operations performed on these inputs are given by the repeater chain protocol our model assumes (Section 3.2).

Sampling from $\mathbf{T_0}$ As input for this Monte Carlo method we have waiting time samples of the generation time between neighboring repeater nodes, which are samples from the random variable T_0. Since the distribution of T_0 is known, i.e. $T_0 \sim \text{geom}(p_{\text{gen}})$, samples from this distribution can be obtained by taking uniform random samples from $[0, 1]$, and transforming these into samples from a geometric distribution by using the inverse transform method [Dev86, Ch. 2].

Sampling from $\mathbf{T_n}$ Equation (5.1) is a recursive function $T(n)$ describing how a single waiting time sample can be obtained from the random variable as in Definition 3.3.1. Algorithm 1 shows how this can be implemented. For $n \geq 1$, the swap from layer $n-1$ to layer n succeeds with probability p_{swap}. If this swap succeeds the waiting time is the maximum of two waiting time samples of a repeater

chain with half the number of segments (2^{n-1}), which are given by $T(n-1)$. If the swap fails all involved entanglement is lost, which means we count the time for the failed attempt and (recursively) try again until success. Determining whether a swap succeeds can be done via a virtual coin flip: a uniform random number from $[0, 1]$ is drawn, and if this number is lower than p_{swap} (which occurs with probability p_{swap}) the swap is considered successful, and the algorithm proceeds accordingly.

$$T(n) = \begin{cases} \text{sample from } T_0 \sim \text{geom}(p_{\text{gen}}), & \text{if } n = 0. \\ T_{\text{swap}}(n), & \text{if } n \geq 1. \end{cases} \tag{5.1}$$

$$T_{\text{swap}}(n) = \begin{cases} \max\{T(n-1), T(n-1)\}, & \text{with prob. } p_{\text{swap}}. \\ \max\{T(n-1), T(n-1)\} + T_{\text{swap}}(n), & \text{with prob. } 1 - p_{\text{swap}}. \end{cases}$$

Algorithm 1: Function $T(n)$, generates waiting time sample from T_n.

Input : Probabilities p_{gen}, p_{swap}, $n = \log_2$(repeater chain segments)
Output: Single sample from T_n
if $n = 0$ **then**
  return sample from T_0
else if $n \geq 1$ **then**
  left $\leftarrow T(n-1)$
  right $\leftarrow T(n-1)$
  $r \leftarrow$ uniform random sample from $[0, 1]$
  if $r \leq p_{\text{swap}}$ **then**
    return max(left, right) `// swap was successful`
  else
    return max(left, right) $+ T(n)$ `// swap failed, retry`
  end
end

Computational complexity For deterministic entanglement swapping (i.e. $p_{\text{swap}} = 1$) the function $T(n)$ would make exactly two recursive calls $T(n-1)$, each of which would make another two, etc, until $T(0)$. This would ultimately result in 2^n samples being drawn from T_0. However, because a swap fails with probability p_{swap}, on average the function $T(n)$ would need to redo these two calls p_{swap}^{-1} times. This ultimately results in $(2p_{\text{swap}}^{-1})^n$ samples being drawn from T_0 on average. Because samples from T_0 can be obtained in constant time, the average time complexity of this method is proportional to

$$\text{number of samples} \cdot \left(\frac{2}{p_{\text{swap}}}\right)^n,$$

which is polynomial in the number of repeater segments N, as $n = \log_2 N$.

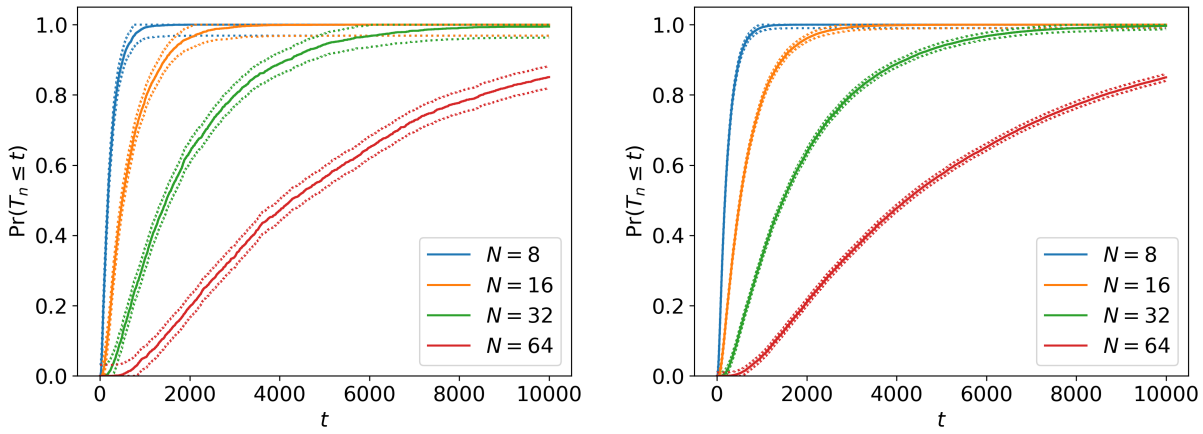
Error bounds We can use the Dvoretzky-Kiefer-Wolfowitz (DKW) inequality to create confidence bands on for the empirically obtained cumulative distribution function, as can be seen in Lemma 5.1. We can also use the bound in Lemma 5.1 to determine how many samples we need to achieve a desired precision. For example, if we want the empirical CDF to be within $\epsilon = 0.01$ of the actual CDF for all t with $1 - \alpha = 99\%$ confidence, we would need approximately $k = 27{,}000$ samples. Figure 5.1 illustrates these confidence bands on the empirical CDF.

Lemma 5.1. *Given some empirical CDF $\hat{F}_n(t)$ obtained from k samples, and the actual CDF $F_n(t) = \Pr(T_n \leq t)$, the probability that the difference between $\hat{F}_n(t)$ and $F_n(t)$ is greater than some ϵ can be bounded using the Dvoretzky-Kiefer-Wolfowitz inequality [Was06, p.14] as follows:*

$$\forall t, \Pr\left(\hat{F}_n(t) - \epsilon \leq F_n(t) \leq \hat{F}_n(t) + \epsilon\right) \leq \alpha,$$

where

$$\epsilon = \sqrt{\frac{\ln(2/\alpha)}{2k}}.$$



(a) *2,700 samples ($\epsilon = 0.03$) (runtime $\approx$ 20 sec. for $N = 64$)*

(b) *27,000 samples, ($\epsilon = 0.01$) (runtime $\approx$ 200 sec. for $N = 64$)*

Figure 5.1: *The empirical cumulative distribution functions of the waiting time generated from samples produced by Algorithm 1 for different number of repeater segments N. Confidence bands from Lemma 5.1 are shown (dotted) for $\alpha = 0.01$ for different sample sizes. The repeater chain parameters here are $p_{\text{gen}} = 0.1$, $p_{\text{swap}} = 0.5$, and no distillation. The given runtimes are for single-threaded computations on commodity hardware.*

5.2 Analytical approach

In this section we investigate how far we can get with solving our probabilistic waiting time problem with analytical methods alone. While we are not able to obtain a complete expression for the entire probability distribution, the results obtained in Section 5.2.1 are useful components for our numerical approach later in Section 5.3. What we *are* able to obtain through analytical means are bounds on the average waiting time (in Section 5.2.2), as well as an expression for the waiting time probability distribution for a modification to the model where entanglement timeouts are applied (in Section 5.2.3).

5.2.1 Analysis of the waiting time distribution

The recursive expression given in Def. 3.3.1 (restated below) contains two distinct operations: Taking the maximum of two random variables, and a sum of i.i.d. random variables where the length of the sum is also a random variable. In the next two paragraphs we analyze these operations.

Definition 3.3.1. *Let T_n denote the waiting time for a repeater chain with 2^n segments, entanglement success probability p_{gen}, swap success probability p_{swap}, and no distillation. T_n has the following recursive definition:*

$$\begin{aligned} T_{n+1} &= (T'_n)_1 + (T'_n)_2 + \cdots + (T'_n)_S \\ T'_n &= \max\{(T_n)_1, (T_n)_2\} \\ T_0 &\sim \text{geom}(p_{\text{gen}}) \\ S &\sim \text{geom}(p_{\text{swap}}) \end{aligned}$$

Waiting for parallel processes As stated before, we assume that a single repeater node can generate entanglement with both its neighbors in parallel. In order to determine what happens to the total waiting time when waiting for two parallel random processes to finish, we want to find the distribution of the maximum of two i.i.d. random variables. This is straightforward to do when describing the random variables in terms of their cumulative distribution functions. Using Corollary 2.10.1 we find that if we know the CDF of the waiting time of a process, we can compute the CDF of the maximum of any number of such processes. This gives us the distribution of when the last process of a number of parallel processes will finish. So in terms of the variables in Def. 3.3.1, where exactly two links need to be generated before a swap can be performed, we have that

$$\Pr(T'_n \leq t) = \Pr(T_n \leq t)^2. \tag{5.2}$$

Probabilistically succeeding operations After having generated the required entanglement a swap can be attempted. When a swap operation fails all involved entanglement is lost and needs to be regenerated. The waiting time for every failed swap attempt is added to the total waiting time, and so the need arises to compute the distribution of sums of random variables. The probability distribution of the sum of two random variables can be computed through the convolution of their probability mass functions (Lem. 2.7). It is however not enough to compute the distribution of the sum of two random variables. Since entanglement swapping succeeds probabilistically, the length of this sum is also a random variable, S. Using Lemma 2.9 we can express the probability mass function of T_{n+1} as

$$\Pr(T_{n+1} = t) = \sum_{s=1}^{t} \Pr(S = s) \Pr\left(\sum_{k=1}^{s} (T'_n)_s = t \right). \tag{5.3}$$

If T'_n were to be geometric, like S already is, T_{n+1} would also be a geometric random variable. However, in our case T'_n is not geometric and this expression fails to be simplified. In terms of analytically solving this problem these convolutions do not provide the results we would hope for.

It turns out that this random sum can be expressed more simply when writing it in terms of the so-called probability generating functions of S and T'_n. However, this approach (see Appendix B) yields no solution to our problem, as there is no expression for the maximum of random variables in terms of their probability generating functions.

5.2.2 Bounds on the mean waiting time

Having difficulty analytically evaluating the waiting time of a repeater chain as in Def. 3.3.1, we look at bounding the average waiting time. It turns out we can find an upper and lower bound on the mean waiting time. These bounds, given in Lemma 5.2, are rather wide, but especially the upper bound is of practical use in the numerical approach later on.

Lemma 5.2. *The mean waiting time* $\mathrm{E}[T_n]$ *can be bounded as*

$$\left(\frac{1}{p_{\text{swap}}}\right)^n \frac{1}{p_{\text{gen}}} \leq \mathrm{E}[T_n] \leq \left(\frac{2}{p_{\text{swap}}}\right)^n \frac{1}{p_{\text{gen}}}. \tag{5.4}$$

Proof. We consider two random variables, Y_n and Z_n, related to T_n, as given in Definitions 5.3 and 5.4. In Corollary 5.5.1 we find that

$$\mathrm{E}[Z_n] \leq \mathrm{E}[T_n] \leq \mathrm{E}[Y_n].$$

The means of Y_n and Z_n can be computed analytically. Starting with $\mathrm{E}[Z_n]$ we can simplify Z_n's definition to

$$Z_n = (Z_{n-1})_1 + (Z_{n-1})_2 + \cdots + (Z_{n-1})_S.$$

Using Wald's identity [Wal45] for the expectation of the sum of a random number of i.i.d. random variables, we find that

$$\mathrm{E}[Z_n] = \mathrm{E}[S] \cdot \mathrm{E}[Z_{n-1}] = \mathrm{E}[S]^n \cdot \mathrm{E}[Z_0] = \left(\frac{1}{p_{\text{swap}}}\right)^n \frac{1}{p_{\text{gen}}}.$$

For $\mathrm{E}[Y_n]$ we get

$$\begin{aligned}
\mathrm{E}[Y_n] &= \mathrm{E}[S] \cdot \mathrm{E}\left[Y'_{n-1}\right] \\
&= \mathrm{E}[S] \cdot (\mathrm{E}[Y_n] + \mathrm{E}[Y_n]) \\
&= \mathrm{E}[S] \cdot 2 \cdot \mathrm{E}[Y_{n-1}] \\
&= (2 \cdot \mathrm{E}[S])^n \cdot \mathrm{E}[Y_0] \\
&= \left(\frac{2}{p_{\text{swap}}}\right)^n \frac{1}{p_{\text{gen}}}.
\end{aligned}$$

□

Definition 5.3. *Let the random variable* Y_n *denote the total number of entanglement attempts for the same repeater chain as in Def. 3.3.1.* Y_n *is similar to* T_n*, but where* T_n *counts parallel entanglement attempts as single waiting time time-steps,* Y_n *counts the total number of entanglement attempts, with parallel attempts counted separately, resulting in the following recursive definition:*

$$\begin{aligned}
Y_n &= \left(Y'_{n-1}\right)_1 + \left(Y'_{n-1}\right)_2 + \cdots + \left(Y'_{n-1}\right)_S \\
Y'_n &= (Y_n)_1 + (Y_n)_2 \\
Y_0 &\sim \text{geom}(p_{\text{gen}}) \\
S &\sim \text{geom}(p_{\text{swap}})
\end{aligned}$$

Definition 5.4. *Let* Z_n *be a random variable similar to* T_n *in Def. 3.3.1, but where* T_n *would need two (in parallel generated) links to perform a swap, for* Z_n *we imagine we would only need one link. This does not have any physical meaning, but we can write a recursive definition regardless:*

$$\begin{aligned}
Z_n &= \left(Z'_{n-1}\right)_1 + \left(Z'_{n-1}\right)_2 + \cdots + \left(Z'_{n-1}\right)_S \\
Z'_n &= Z_n \\
Z_0 &\sim \text{geom}(p_{\text{gen}}) \\
S &\sim \text{geom}(p_{\text{swap}})
\end{aligned}$$

Lemma 5.5. *For T_n, Y_n, Z_n as in Definition 3.3.1 and Defs. 5.3 and 5.4 respectively, we have that*

$$Z_n \leq_{\text{st}} T_n \leq_{\text{st}} Y_n,$$

i.e.

$$\forall t, \Pr(Z_n > t) \leq \Pr(T_n > t) \leq \Pr(Y_n > t).$$

Proof. The recursive definition of Y_n replaces $T'_n = \max\{(T_n)_1, (T_n)_2\}$ with $Y'_n = (Y_n)_1 + (Y_n)_2$, while Z_n replaces it with $Z'_n = Z_n$. Because the rest of the definition is the same, we have that, using Lemma 2.12

$$Z_n \leq_{\text{st}} T_n \leq_{\text{st}} Y_n.$$

□

Corollary 5.5.1. *From Lemma 5.5 follows, using Lemma 2.5, that*

$$\mathrm{E}[Z_n] \leq \mathrm{E}[T_n] \leq \mathrm{E}[Y_n].$$

Another upper bound on the waiting time would be obtained by first generating all entanglement, followed by all the swaps, and if one of the swaps fails discard all the entanglement in the entire repeater chain. This is formalized in Definition 5.6. Discarding all entanglement if any swap fails is described by a geometric distribution, where the probability of a successful event is the probability that all $2^n - 1$ swaps succeed in a single attempt, which is $p_{\text{swap}}^{2^n-1}$. We still assume all the required entanglement, in this case 2^n links, can be generated in parallel.

Definition 5.6. *For a repeater chain of length 2^n, let random variable W'_n be the waiting time to prepare all entanglement between neighboring nodes, and W_n the waiting time to successfully perform all swaps at the same attempt after having generated all entanglement.*

$$\begin{aligned} W_n &= (W'_n)_1 + (W'_n)_2 + \cdots (W'_n)_S \\ W'_n &= \max\{(W_0)_1, (W_0)_2, \cdots, (W_0)_{2_n}\} \\ W_0 &\sim \text{geom}(p_{\text{gen}}) \\ S &\sim \text{geom}(p_{\text{swap}}^{2^n-1}) \end{aligned}$$

Since in this case we regenerate *all* entanglement (instead of just the involved entanglement) when a single swap fails, the average waiting time of W_n will be greater or equal to that of T_n, so

$$\mathrm{E}[T_n] \leq \mathrm{E}[W_n]$$

Work on repeater chains with deterministic swapping already provides expressions for the mean of 2^n geometric(p_{gen}) random variables, i.e. $\mathrm{E}[W'_n]$, as seen in Eq. (4.3). However, as shown in [Eis08] this mean can also be tightly bounded, as shown below. These bounds make it easier to study the behavior of this waiting time for large n.

$$\frac{1}{\lambda}\sum_{k=1}^{2^n}\frac{1}{k} < \mathrm{E}\left[W'_n\right] < 1 + \frac{1}{\lambda}\sum_{k=1}^{2^n}\frac{1}{k} \tag{5.5}$$

where

$$(1-p) = e^{-\lambda}$$

and

$$\sum_{k=1}^{2^n} \frac{1}{k} = H_{2^n}$$

is the 2^n-th harmonic number. So for a repeater chain with 2^n segments, in order to generate all entanglement, the waiting time is upper bounded by

$$\mathrm{E}\left[W_n'\right] < 1 + \frac{H_{2^n}}{-\ln(1-p_{\mathrm{gen}})}$$

Using Wald's equation we find the mean of the sum of W_n''s in the definition of W_n as

$$\mathrm{E}\left[W_n\right] = \mathrm{E}\left[S\right] \cdot \mathrm{E}\left[W_n'\right]$$

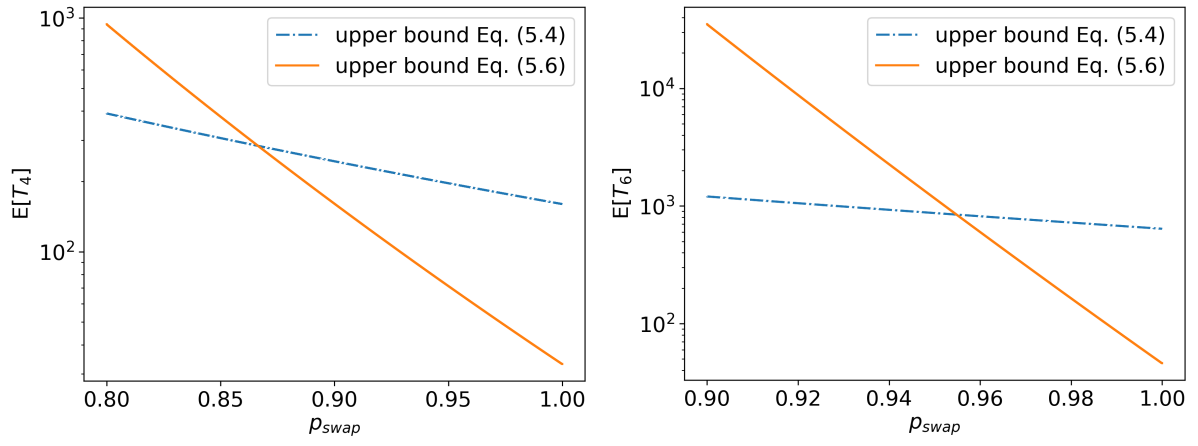
and since

$$\mathrm{E}\left[S\right] = \left(\frac{1}{p_{\mathrm{swap}}}\right)^{2^n-1}$$

we get

$$\mathrm{E}\left[T_n\right] < \left(\frac{1}{p_{\mathrm{swap}}}\right)^{2^n-1} \cdot \left(1 + \frac{H_{2^n}}{-\ln(1-p_{\mathrm{gen}})}\right) \tag{5.6}$$

The harmonic numbers behave asymptotically as $\ln(x) + \gamma$, where γ is the Euler-Mascheroni constant [GKP94, p. 278]. Since the harmonic numbers grow logarithmically, the 2^n-th harmonic number grows linearly in n. For small p_{gen}, $-\ln(1-p_{\mathrm{gen}})$ is very close to p_{gen}. So compared to the $1/p_{\mathrm{gen}}$ term we had in the previous upper bound this is somewhat of the same order. The first term makes a bigger difference. For deterministic or near-deterministic swapping, i.e. when p_{swap} is very close to 1, this upper bound is better than the one previously found in Eq. (5.4), but for smaller p_{swap} this grows much faster than this previous upper bound. Figure 5.2 gives a comparison between these bounds for different swapping probability.



(a) *$N = 16$ repeater segments.*

(b) *$N = 64$ repeater segments.*

Figure 5.2: *A comparison between the upper bound on the mean waiting time from Equations* (5.4) *and* (5.6) *for different values of p_{swap}, $p_{\mathrm{gen}} = 0.1$, and no distillation.*

5.2.3 Entanglement timeouts

When applying entanglement timeouts at fixed intervals (defined in Section 3.3.2), it is possible to compute the waiting time distribution entirely analytically.

Generating the required entanglement to attempt a swap operation on layer n requires swaps to succeed on layer $n-1$ first. Entanglement swapping only happens at the end of each bin if the required entanglement is available, otherwise any current entanglement is discarded. So if the cut-off point is set to the same moment in time for every nesting layer a situation is created where all the swaps in the repeater chain need to succeed on the same attempt. After all, if even one of them fails the repeater chain fails to generate entanglement between the end nodes, and at this point all the current entanglement is discarded because of the timeout.

The probability of generating entanglement between the end nodes of a repeater chain with 2^n segments depends on two things. First all the required entanglement on layer 0 needs to be generated before the cut-off time τ is reached. The probability of generating entanglement between physically neighboring nodes is described, as per Def. 3.3.1, by $T_0 \sim \text{geom}(p_{\text{gen}})$, and there are 2^n links to be generated. From this, in combination with Corollary 2.10.1, it follows that the probability of generating the required entanglement before the cut-off time τ is given by

$$\Pr(T_0 \leq \tau)^{2^n} = (1 - (1 - p_{\text{gen}})^{\tau})^{2^n}.$$

The probability that a single swap succeeds is p_{swap}, so the probability that all $2^n - 1$ swaps succeed on a single attempt is $p_{\text{swap}}^{2^n-1}$. The probability of successfully entangling the end nodes at the end of the timeout bin becomes

$$p_c(n) = \Pr(T_0 \leq \tau)^{2^n} \cdot p_{\text{swap}}^{2^n-1}.$$

This value $p_c(n)$ describes the probability a single attempt being successful, an attempt being the generation of entanglement between the end nodes before time τ. The probability distribution of the first successful event is then given by a geometric distribution. Since swaps (and therefore also entanglement delivery) always happen at the cut-off points, the waiting time will be some multiple of the cut-off value. Ultimately, for the total waiting time we get a geometric distribution multiplied by some scalar τ.

$$T_n^{\text{cutoff}} = \tau \cdot X_n \tag{5.7}$$

with

$$X_n \sim \text{geom}(p_c(n)),$$

and

$$p_c(n) = (1 - (1 - p_{\text{gen}})^{\tau})^{2^n} \cdot p_{\text{swap}}^{2^n-1}.$$

5.3 Numerical approach

As discussed in the previous section, we fail to obtain a completely analytical result for the repeater chain model with probabilistic swapping and without timeouts. In Section 5.3.1 we show how we can compute this probability distribution numerically instead. The obtained numerical results are exact for $0 \leq t \leq t_{\text{trunc}}$, where t_{trunc} is the maximum number of time steps we want to find $\Pr(T_n = t)$ for, and can

be increased at the expense of greater computational costs (elaborated on in Section 5.3.2). We also show, in Section 5.3.3, how the bounds on the average waiting time found in Section 5.2.2 can be improved by including results from our numerical calculation of the waiting time distribution.

5.3.1 Calculating the waiting time distribution

The core idea of this approach is to keep track of the probability mass function of the waiting time numerically, i.e. as a list of probabilities for different waiting times t. Every step in the swapping protocol has an effect on the probability distribution of the waiting time. These effects are numerically computed for all the steps of the protocol, at the end of which we obtain the waiting time distribution for connecting the end nodes of the repeater chain in question. Algorithm 2 gives a high level overview of this process when distillation is omitted, while Algorithm 3 shows how m rounds of distillation per round of swaps would be included in this calculation.

As seen in Definition 3.3.1, the probability distribution of T_0, i.e. the waiting time for a repeater chain with $2^0 = 1$ segment, is geometric. Since this distribution has infinite support, it is impossible to store the entire PMF numerically in a finite amount of space, and so this tail needs to be truncated. We call this truncation point t_{trunc}. For simplicity we fix this truncation point at the beginning of the algorithm.

The next paragraphs describe how we can numerically obtain the distribution of T_{n+1} from the distribution of T_n. In order to compute the distribution of T_n we start at T_0, of which we know the distribution, and iterate over the steps in the algorithm n times.

We can verify the output of the algorithm against the statistics produced by generating waiting time samples using the Monte Carlo method (Alg. 1), an example of which is given in Figure 5.3.

Algorithm 2: Numerically calculate the distribution of waiting time T_n for the basic model (Section 3.3.1) without distillation.

Input : Initial PMF of $T_0 \sim \text{geom}(p_{\text{gen}})$ as an array
Output: Waiting time PMF for a repeater chain with 2^n segments
for *1* to n **do**
    PMF $\leftarrow$ wait_for_entanglement(PMF) `// (Using Alg. 4)`
    PMF $\leftarrow$ swap(PMF, p_{swap}) `// (Using Alg. 6)`
end

Algorithm 3: Numerically calculate the distribution of waiting time T_n for the basic model (Section 3.3.1) with nested distillation.

Input : Initial PMF of $T_0 \sim \text{geom}(p_{\text{gen}})$ as an array
Output: Waiting time PMF for a repeater chain with 2^n segments
for *1* to n **do**
    for *1* to m **do**
        PMF $\leftarrow$ wait_for_entanglement(PMF) `// (Using Alg. 4)`
        PMF $\leftarrow$ distillation(PMF, p_{dist}) `// (Using Alg. 6)`
    end
    PMF $\leftarrow$ wait_for_entanglement(PMF) `// (Using Alg. 4)`
    PMF $\leftarrow$ swap(PMF, p_{swap}) `// (Using Alg. 6)`
end

Waiting for parallel processes Before a swap or distillation attempt can be made, it is necessary to wait for the required entanglement first. In our model we require exactly two links to do either, and we assume that these links can be generated in parallel in both cases. When both the required links have the same waiting time distribution, the CDF of the resulting distribution can be found by squaring the CDF of the input distribution, as seen in Equation (5.2). It is possible to translate between the PMF and CDF of a random variable using Corollary 2.2.1. Algorithm 4 gives a pseudocode overview of computing the new probability mass function.

Algorithm 4: Computing the probability mass function for two parallel i.i.d. random processes

Input : PMF (as array) of completion time of single process
Output: PMF (as array) of maximum waiting time to complete 2 such independent processes in parallel

```
CDF ← pmf_to_cdf(PMF)
for t = 0 to t_trunc do
    CDF[t] ← CDF[t]^2
end
PMF ← cdf_to_pmf(CDF)
```

Probabilistically succeeding operations Recall from Def. 3.3.1 that we can describe the effect of a probabilistically succeeding swap or distillation attempt on the waiting time as a geometric sum of i.i.d. random variables.

$$T_{n+1} = (T_n)_1 + (T_n)_2 + \cdots + (T_n)_S \qquad \text{(from Def. 3.3.1)}$$

where in the case of entanglement swapping $S \sim \text{geom}(p_{\text{swap}})$. The sums of two random variables can be computed through the convolution of their PMFs (Lem. 2.7). Algorithm 5 shows how this can be done for our numerical, truncated distributions.

Algorithm 5: Computing the probability mass function of $Z = X + Y$

Input : PMF_X, PMF_Y as arrays
Output: PMF_Z as array

```
PMF_Z ← t_trunc zeros
for x = 0 to t_trunc do
    for y = 0 to t_trunc − x do
        PMF_Z[x + y] ← PMF_Z[x + y] + PMF_X[x]·PMF_Y[y]
    end
end
```

As a side note, the convolution of two discrete functions can also be computed using fast Fourier transforms (FFTs) [BB88]. For two functions f_1 and f_2, the Fourier transform of their convolution equals the product of their Fourier transforms.

$$\text{FFT}(f_1 * f_2) = \text{FFT}(f_1) \cdot \text{FFT}(f_2) \tag{5.8}$$

This method is computationally cheaper than computing the convolutions directly. The details of this complexity improvement are given in Section 5.3.2.

Being able to numerically compute distributions of the sum of two random variables, we now need to solve the problem for the case where the length of the sum is also a random variable. The expression we obtained in Equation (5.3) is the following:

$$\Pr(T_{n+1} = t) = \sum_{s=1}^{\infty} \Pr(S = s) \Pr\left(\sum_{k=1}^{s} T'_n = t\right) \tag{5.3 restated}$$

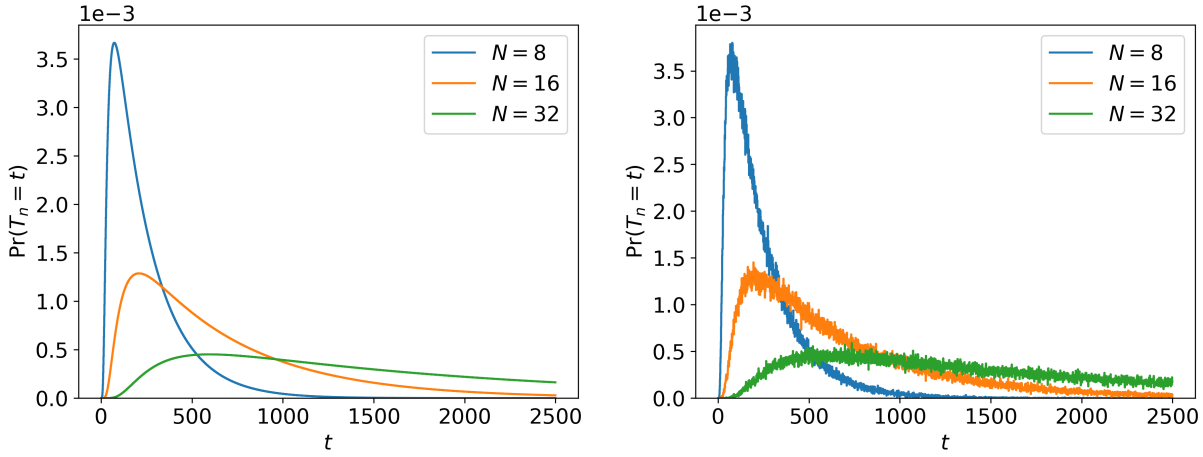
Something which might seem a problem at first is that, since S is geometrically distributed, $\Pr(S = s)$ will never become 0, and in order to obtain an accurate result we would have to evaluate this sum for an infinite number of terms. However, if we look at Lemma 2.9 we see that in order to compute $\Pr(T_n = t)$ this sum only needs to run from $S = 1$ to $S = t$, as for any other values of S the terms in this sum become 0. The intuition here is that in the process which distributes entanglement between the end nodes in t time steps can never involve $s > t$ number of swaps. This leaves

$$\Pr(T_{n+1} = t) = \sum_{s=1}^{t} \Pr(S = s) \Pr\left(\sum_{k=1}^{s} T'_n = t\right). \tag{5.9}$$

Algorithm 6 shows how this can be implemented. Because we are not calculating each $\Pr(T_n = t)$ individually, but rather the entire PMF in one go (so that we can leverage efficient library implementations of subroutines) it also suffices to evaluate this geometric sum from $S = 1$ to $S = t_{\text{trunc}}$.

Algorithm 6: Computing the PMF of $Z = X_1 + ... + X_S$. Note that in the algorithm below, $*$ indicates the convolution of two PMFs, and $+$ indicates the linear combination of two arrays.

Input : PMF_X as array, $0 < p_{\text{swap}} \leq 1$ such that $S \sim \text{geom}(p_{\text{swap}})$
Output: PMF_Z as array
$\text{PMF}_Z \leftarrow 0$
$\text{PMF\_temp} \leftarrow \text{PMF}_X$
for *s = 1* to t_{trunc} **do**
  $\text{PMF}_Z \leftarrow \text{PMF}_Z + \Pr(S = s) \cdot \text{PMF\_temp}$
  $\text{PMF\_temp} \leftarrow \text{PMF\_temp} * \text{PMF}_X$ `// (Using Alg. 5)`
end



(a) *Numerically calculated waiting time PMF with Algorithm 2. (runtime ≈ 10 sec. for $N = 32$)*

(b) *Empirical waiting time PMF, obtained from 270,000 samples generated using Algorithm 1. (runtime ≈ 500 sec. for $N = 32$)*

Figure 5.3: *We can verify the exact numerical results from Algorithm 2 against the empirical distribution obtained from the Monte Carlo simulation. The repeater chain parameters here are $p_{\text{gen}} = 0.1$, $p_{\text{swap}} = 0.5$, and no distillation, for varying number of repeater segments N. The given runtimes are for single-threaded computations on commodity hardware.*

5.3.2 Computational complexity

The computational complexity of this method is given in Lemma 5.7. This complexity depends not only on the number of repeater segments N, but also on the value of t_{trunc}. As Lemma 5.10 shows, in order to capture a minimum amount of probability mass c so that $\Pr(T_n < t_{\text{trunc}}) \geq c$, t_{trunc} grows polynomially with N. Lemma 5.10 also suggests that the value of t_{trunc} would be heavily dependent on p_{swap}, which is confirmed by the comparison of numerical results for different p_{swap} in Figure 5.4.

Lemma 5.7. *The computational complexity of Algorithm 2 is given by*

$$O\left(\log N \cdot t_{\text{trunc}}^2 \log t_{\text{trunc}}\right)$$

where t_{trunc} can be set according to Lemma 5.10.

Proof. The loop in Algorithm 2 has $n = \log N$ iterations over Algorithms 4 and 6. This gives a time complexity of

$$O\left(\log N \cdot f(t_{\text{trunc}})\right)$$

where $f(t_{\text{trunc}})$ is the time complexity of running both Algorithms 4 and 6. Algorithm 4 has a time complexity of $O(t_{\text{trunc}})$ (Lem. 5.8), and Algorithm 6 has a time complexity of $O(t_{\text{trunc}}^2 \log t_{\text{trunc}})$ (Lem. 5.9). This brings the overall time complexity to

$$O\left(\log N \cdot t_{\text{trunc}}^2 \log t_{\text{trunc}}\right)$$

□

Lemma 5.8. *The computational complexity of Algorithm 4 is given by*

$$O\left(t_{\text{trunc}}\right).$$

Proof. Translating the PMF to a CDF, squaring all the elements in the CDF, and translating the resulting CDF back to a PMF can be done in time linear to the support of the distribution, so for our truncated distributions the entire algorithm takes $O(t_{\text{trunc}})$ time. □

Lemma 5.9. *The computational complexity of Algorithm 6 is given by*

$$O\left(t_{\text{trunc}}^2 \log t_{\text{trunc}}\right).$$

Proof. For Algorithm 6, t_{trunc} convolutions need to be done. Each of these convolutions can be done in $O(t_{\text{trunc}}^2)$ time using Algorithm 5, or in $O(t_{\text{trunc}} \log t_{\text{trunc}})$ time using fast Fourier transforms [BB88], bringing the total time complexity of Algorithm 6 to $O(t_{\text{trunc}}^2 \log t_{\text{trunc}})$. □

Lemma 5.10. *In order to capture a minimum amount of probability mass* $\Pr(T_n < t_{\text{trunc}}) \geq c$ *of the distribution of* T_n*, for some constant* $0 \leq c \leq 1$*, the value* t_{trunc} *needs to increase polynomially with the number of repeater segments* N*.*

$$t_{\text{trunc}} = O\left(\frac{1}{1-c}\left(\frac{2}{p_{\text{swap}}}\right)^{\log N}\frac{1}{p_{\text{gen}}}\right).$$

Proof. Markov's inequality [ES10] for non-negative random variables states

$$\Pr(T_n \geq t_{\text{trunc}}) \leq \frac{\mathrm{E}\left[T_n\right]}{t_{\text{trunc}}}.$$

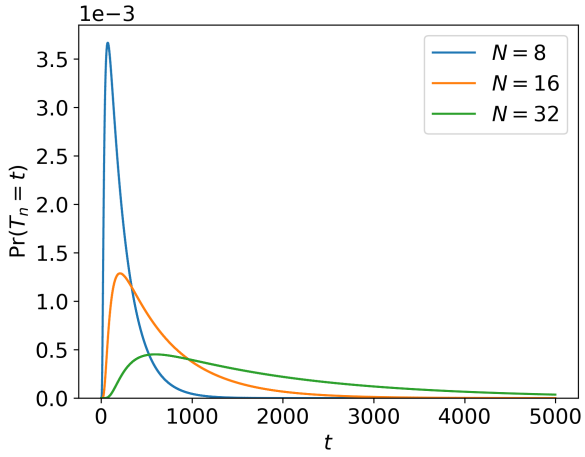
Using this bound we find that, if we want to capture at least $\Pr(T_n < t_{\text{trunc}}) \geq c$ probability mass, we can achieve this by setting

$$t_{\text{trunc}} = \frac{\mathrm{E}\left[T_n\right]}{1-c}.$$

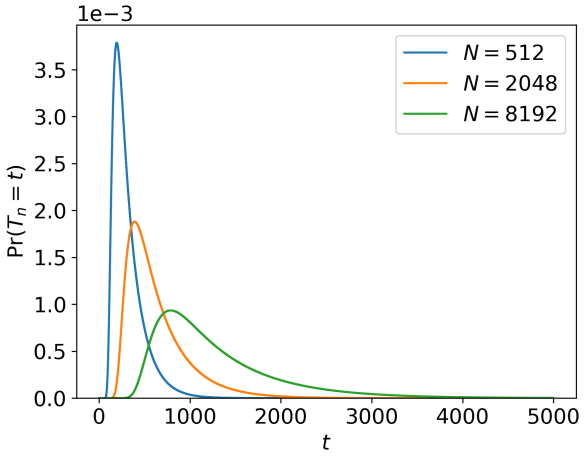
Using the upper bound on $\mathrm{E}\left[T_n\right]$ from Lemma 5.2 we get

$$t_{\text{trunc}} \leq \frac{1}{1-c}\left(\frac{2}{p_{\text{swap}}}\right)^{\log N}\frac{1}{p_{\text{gen}}}.$$

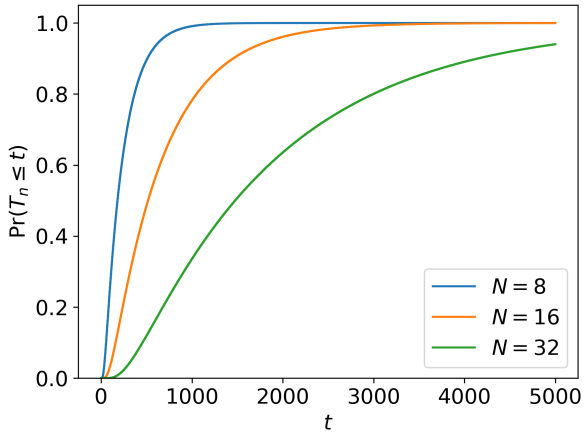
□



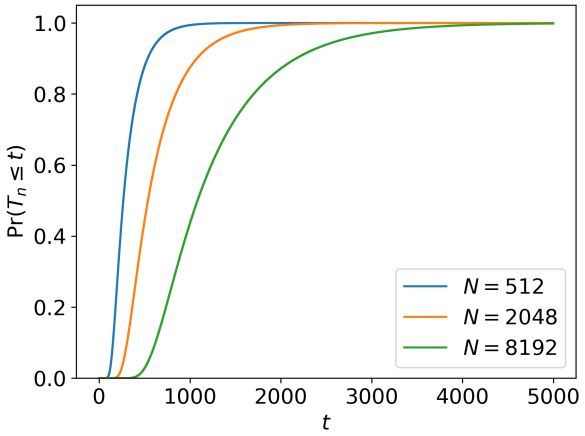
(a) *Waiting time PMF for $p_{\text{swap}} = 0.5$ (runtime $\approx$ 75 sec. for N = 32)*



(b) *Waiting time PMF for $p_{\text{swap}} = 0.9$ (runtime $\approx$ 150 sec. for N = 8,192)*



(c) *Waiting time CDF for $p_{\text{swap}} = 0.5$, $\Pr(T_5 \leq 5000) = 0.94$ (runtime $\approx$ 75 sec. for N = 32)*



(d) *Waiting time CDF for $p_{\text{swap}} = 0.9$, $\Pr(T_{13} \leq 5000) = 0.9985$ (runtime $\approx$ 150 sec. for N = 8,192)*

Figure 5.4: *As can also be see in Lemma 5.10, the value to set t_{trunc} to in order to capture close to 100% of the probability mass is heavily dependent on p_{swap}. Here we set $t_{\text{trunc}} = 5{,}000$ and compare the captured probability mass $\Pr(T_n \leq 5000)$ for different numbers of repeater segments $N = 2^n$ and different swap success probabilities. The probability $p_{\text{swap}} = 0.5$ is the highest theoretically achievable success probability for a single entanglement swap attempt via linear optics in a setup like [DLCZ01], although different setups for near-deterministic swapping have been proposed [BRP+10]. The given runtimes are for single-threaded computations on commodity hardware.*

Multiprocessing It is possible to parallelize the for-loop of length t_{trunc} in Algorithm 6 to some degree. At each iteration of this loop two things happen: a linear (weighted) sum of two PMFs taking $O(t_{\text{trunc}})$ time, and a convolution taking at best $O(t_{\text{trunc}} \log t_{\text{trunc}})$ time. Because the convolutions are the most computationally expensive of these two, we will focus on parallelizing these.

Let f_1 be the PMF of X, which is the input distribution of Algorithm 6, and f_k the convolution of k of these functions.

$$f_k = \underbrace{f_1 * f_1 * \ldots * f_1}_{k}$$

The for-loop in Algorithm 6 iteratively computes all f_k up to $f_{t_{\text{trunc}}}$ using

$$f_k = f_{k-1} * f_1.$$

However, from the associativity of convolutions it follows that f_k can also be computed in the following way. For any $j, l \geq 1$ such that $j + l = k$ we have that

$$f_k = f_j * f_l.$$

Using this we can divide the functions f_1 through $f_{t_{\text{trunc}}}$ which need to be computed into a number of groups, such that any function f_k in a given group can be computed from the convolution of f_j and f_l in a previous group. Table 5.1 gives an overview of the resulting groups of functions. For example f_7 in group 4 can be obtained using $f_3 * f_4$ which have been computed earlier in group 3, while computing f_9 in group 5 is impossible without at least one f_k from group 4.

All the functions in any given group can be computed in parallel on different threads as they do not depend on each other. Because the number of functions which can be computed from functions in previous groups grows exponentially with each group, the total number of these groups will be $O(\log t_{\text{trunc}})$. This means that in the limit of having an infinite number of independent threads, the total time complexity would drop from $O(t_{\text{trunc}}^2 \log t_{\text{trunc}})$ to $O(t_{\text{trunc}} \log^2 t_{\text{trunc}})$. When the number of available threads is much smaller than t_{trunc}, the speedup of Alg. 6 gains from parallelization will behave like a constant factor in the number of threads.

group	f_k only dependent on previous groups
1	f_1
2	f_2
3	f_3, f_4
4	f_5, f_6, f_7, f_8
5	$f_9, \ldots, f_{16}$
6	$f_{17}, \ldots, f_{32}$
	$\vdots$
$\lceil \log_2 t_{\text{trunc}} \rceil + 1$	$f_{\frac{t_{\text{trunc}}}{2}+1}, \ldots, f_{t_{\text{trunc}}}$

Table 5.1: *Functions f_1 through $f_{t_{\text{trunc}}}$ divided into groups, such that all f_k in a given group can be computed using f_j and f_l from previous groups.*

5.3.3 Improving bounds on the mean waiting time

The bound on the mean derived in Section 5.2.2 can be improved upon using the numerical PMFs obtained from the approach described in Section 5.3.1.

Numerical lower bound Our algorithm calculates the PMF of T_n up until some truncation point t_{trunc}. By truncating this distribution we are discarding all elements in the support of T_n greater than t_{trunc}. If we renormalize this truncated distribution we obtain a new distribution of a random variable which we will call T_n^{trunc} (Def. 5.11). The mean of this new random variable forms a lower bound on the mean of T_n, as shown in Lemma 5.12.

Definition 5.11. *We define the random variable T_n^{trunc} as T_n given $T_n < t_{\text{trunc}}$. The probability mass function of T_n^{trunc} is given by*

$$\Pr\big(T_n^{\text{trunc}} = t\big) = \begin{cases} \Pr(T_n = t) / \Pr(T_n < t_{\text{trunc}}), & \text{if } t < t_{\text{trunc}}. \\ 0, & \text{if } t \geq t_{\text{trunc}}. \end{cases}$$

Lemma 5.12. *The mean of T_n is lower bounded by the mean of T_n^{trunc} (Def. 5.11).*

$$\mathrm{E}\left[T_n^{\text{trunc}}\right] \leq \mathrm{E}\left[T_n\right].$$

Proof. Using Definition 5.11 we write the CDF of T_n^{trunc} as

$$\Pr\big(T_n^{\text{trunc}} \leq t\big) = \begin{cases} \sum_{k=1}^{t} \Pr(T_n = k) / \Pr(T_n < t_{\text{trunc}}), & \text{if } t < t_{\text{trunc}}. \\ 1, & \text{if } t \geq t_{\text{trunc}}. \end{cases}$$

Comparing this we the CDF of T_n

$$\Pr(T_n \leq t) = \sum_{k=1}^{t} \Pr(T_n = k)$$

we can see that

$$\forall t, \Pr\big(T_n^{\text{trunc}} \leq t\big) \geq \Pr(T_n \leq t)$$

which we can also write as

$$\forall t, \Pr\big(T_n^{\text{trunc}} > t\big) \leq \Pr(T_n > t).$$

Using Lemma 2.5, we get

$$\mathrm{E}\left[T_n^{\text{trunc}}\right] \leq \mathrm{E}\left[T_n\right].$$

□

Numerical upper bound An upper bound on $\mathrm{E}[T_n]$ is given in Lemma 5.13. This upper bound allows us to combine information from our numerically calculated, truncated distribution of T_n, with the analytical upper bound on the mean found in Lemma 5.2.

Lemma 5.13. *The mean of T_n can be upper bounded by*

$$\mathrm{E}\left[T_n\right] \leq \sum_{k=1}^{t_{\text{trunc}}-1} \Pr(T_n \geq k) + \sum_{k=t_{\text{trunc}}}^{\infty} \Pr(Y_n \geq k)$$

where Y_n is given in Definition 5.3.

Proof. Using Lemma 2.5, the expectation of T_n is given by

$$\begin{aligned} \mathrm{E}\left[T_n\right] &= \sum_{t=1}^{\infty} \Pr(T_n \geq t) \\ &= \sum_{t=1}^{t_{\text{trunc}}} \Pr(T_n \geq t) + \sum_{t=t_{\text{trunc}}}^{\infty} \Pr(T_n \geq t). \end{aligned}$$

Using

$$\forall t, \Pr(T_n \geq t) \leq \Pr(Y_n \geq t),$$

from Lemma 5.5, we can replace the second summation with one over $\Pr(Y_n \geq t)$ and obtain the following inequality:

$$\mathrm{E}[T_n] \leq \sum_{k=1}^{t_{\text{trunc}}-1} \Pr(T_n \geq k) + \sum_{k=t_{\text{trunc}}}^{\infty} \Pr(Y_n \geq k).$$

□

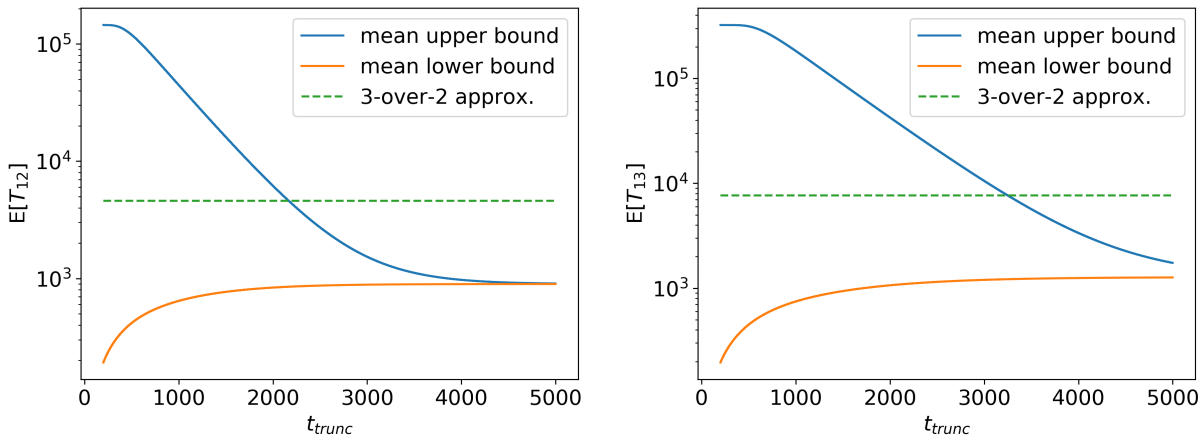
The terms in the first summation of the upper bound given in Lemma 5.13 can be obtained from our truncated, numerical distribution of T_n. Obtaining the value of the second summation is a bit more involved. When trying to bound tail of Y_n we notice that, aside from its mean, we have no closed form expression for the behavior of Y_n. However, using the same tools for computing the distribution of T_n numerically up to some t_{trunc}, we can also compute the distribution of Y_n up to that same t_{trunc}. The mean of Y_n can then be written as

$$\mathrm{E}[Y_n] = \sum_{t=1}^{t_{\text{trunc}}-1} \Pr(Y_n \geq t) + \sum_{t=t_{\text{trunc}}}^{\infty} \Pr(Y_n \geq t).$$

Because we can obtain the values in the first of these summations from our numerical calculation of Y_n, and we know the mean of Y_n analytically (Lem. 5.2), we can compute the value of the second summation as follows:

$$\begin{aligned}\sum_{t=t_{\text{trunc}}}^{\infty} \Pr(Y_n \geq t) &= \mathrm{E}[Y_n] - \sum_{t=1}^{t_{\text{trunc}}-1} \Pr(Y_n \geq t) \\ &= \left(\frac{2}{p_{\text{swap}}}\right)^n \frac{1}{p_{\text{gen}}} - \sum_{t=1}^{t_{\text{trunc}}-1} \Pr(Y_n \geq t).\end{aligned}$$

The upper bound in Lemma 5.13 can be made arbitrarily tight by increasing the value of t_{trunc}.



(a) *Bounds on* $\mathrm{E}[T_{12}]$*, i.e.* $N = 4{,}096$ *repeater segments. (runtime* $\approx$ *140 sec.)*

(b) *Bounds on* $\mathrm{E}[T_{13}]$*, i.e.* $N = 8{,}192$ *repeater segments. (runtime* $\approx$ *150 sec.)*

Figure 5.5: *Using Lemmas 5.12 and 5.13 we can compute increasingly better bounds on the average waiting time by increasing* t_{trunc}*. From Definition 5.11 it follows that these bounds will always converge on the actual mean for sufficiently large* t_{trunc}*. The repeater chain parameters here are* $p_{\text{gen}} = 0.1$*,* $p_{\text{swap}} = 0.9$*, and no distillation. The given runtimes are for single-threaded computations on commodity hardware.*

Chapter 6

Results: fidelity

Using the model for fidelity as given in Section 3.4, and having the previously described methods for calculating the waiting time probability distribution, our next goal is to find the average fidelity as a function of the generation time. In Section 6.1 we describe how this can be done for the Monte Carlo method, and in Section 6.2 we describe how this can be done for the numerical approach.

6.1 Monte Carlo simulation

Because our Monte Carlo simulation draws individual waiting time samples, it is straightforward to calculate the fidelity alongside each sample, given the functions for swapping, distillation, and decay described in Section 3.4.

Entanglement swapping and decay Algorithm 7 shows which additions to the Monte Carlo simulation for waiting time (Alg. 1) can be made to produce fidelity samples alongside the waiting time samples when we only consider entanglement swapping and decay over time, but no distillation.

Entanglement distillation Because we assume distillation to follow the same nested structure as entanglement swapping, we can use a similar recursive function as Algorithm 7 to compute fidelities after m layers of distillation, using the function for distillation from Section 3.4. For this the line

$$\text{F\_swap} \leftarrow \text{swap(F\_left, F\_right)}$$

would be replaced by

$$\text{F\_swap}\,, p_{\text{dist}} \leftarrow \text{distill(F\_left, F\_right)}$$

and the random sample r would then be checked against (this now fidelity dependent) p_{dist}.

Algorithm 7: Function $F(n)$, generates waiting time sample from T_n and corresponding fidelity. Differences with Algorithm 1 have been highlighted.

Input : Params of Algorithm 1, fidelity of links between neighbors F_0
Output: Single sample from T_n and corresponding fidelity
if $n = 0$ **then**
  return (sample from T_0, F_0)
else if $n \geq 1$ **then**
  (left, F_left) $\leftarrow F(n-1)$
  (right, F_right) $\leftarrow F(n-1)$
  diff $\leftarrow$ |left − right|
  if left $<$ right **then**
    F_left $\leftarrow$ decay(F_left, diff) `// left link earlier`
  else if right $<$ left **then**
    F_right $\leftarrow$ decay(F_right, diff) `// right link earlier`
  end
  F_swap $\leftarrow$ swap(F_left, F_right)
  $r \leftarrow$ uniform random sample from $[0, 1]$
  if $r \leq p_{\text{swap}}$ **then**
    return max(left, right), F_swap `// swap was successful`
  else
    return max(left, right) + $F(n)$, F_swap `// swap failed, retry`
  end
end

6.2 Numerical approach

Computing fidelities in our exact numerical approach is more involved than for the Monte Carlo simulation. Here we present a method for computing output fidelities as a function of time, given we only consider decay over time, and deterministic entanglement swapping (i.e. $p_{\text{swap}} = 1$). We can verify the results of this calculation against fidelities produced by the Monte Carlo simulation, which can be seen in Figure 6.1.

Fidelity as function of time We start by defining the function $F_n(t)$ as the average fidelity of entangled pairs outputted by a repeater chain with 2^n segments at time t. For $F_0(t)$ we assume this to be some constant value: the fidelity of links generated between neighboring nodes (a repeater chain with 2^0 segments) does not depend on time, as the links do not spend any time in memory and thus do not decay. The next thing we will do is show how the function $F_{n+1}(t)$ can be computed, given we have $F_n(t)$.

Decay of earlier generated link We first look at the fidelities of both input links for an entanglement swap, given the swap happens at time t. We will refer to these links as "earlier" and "later", referring to the order in which they are generated. The later link of the two does not decay between the time it is generated and the time the swap is performed because our model assumes the swap is performed

directly after the second link is ready. For the fidelity of the later link we can thus write

$$F_n^{\text{later}}(t) = F_n(t).$$

The earlier generated link however does decay. When the later link is generated at time t, there are a number of combinations in which this could have happened. In order for the later link to be generated at time t, either the "left" link or the "right" link needs to be generated at time t (or both). Table 6.1 givens an overview of this.

left link	1	2	3	$\cdots$	t−1	t	t	t	$\cdots$	t	t
right link	t	t	t	$\cdots$	t	1	2	3	$\cdots$	t−1	t

Table 6.1: *All possible combinations of generation times for two links such that the maximum generation time (the generation time of the later link) is equal to t.*

Each of these tuples of left and right has a probability associated with it. These probabilities can be obtained from our numerically calculated probability distribution of T_n. Using these probabilities we we can extract the following expression for the average decay the earlier link experiences while waiting for the later link, given this later link is created at time t.

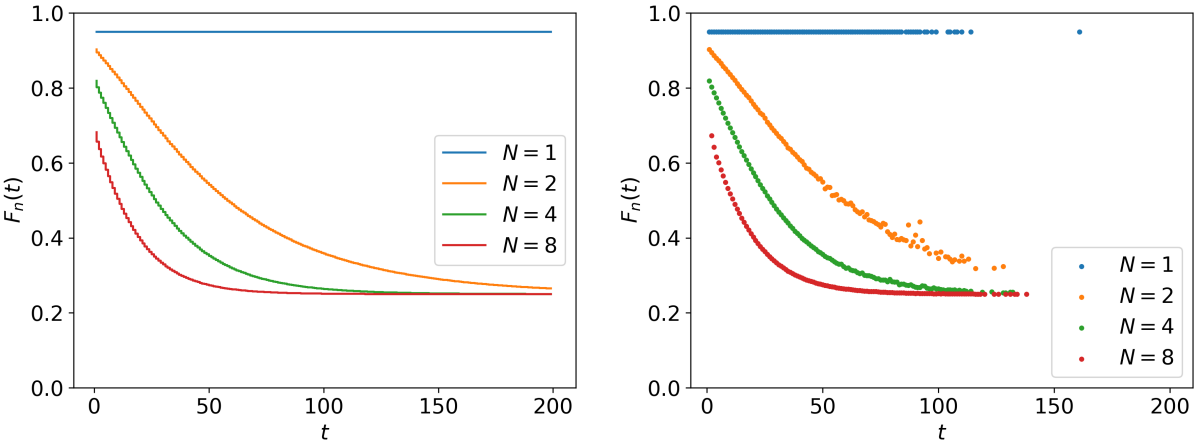
$$F_n^{\text{earlier}}(t) = \Bigg[\sum_{k=1}^{t-1} \text{decay}\left(F_n(t), t-k\right) \cdot 2 \Pr(T_n = k) \Pr(T_n = t) + F_n(t) \cdot \Pr(T_n = t)^2 \Bigg] / \Pr(\max\{(T_n)_l, (T_n)_r\} = t),$$

where $\text{decay}(F, \Delta t)$ is the function given in Section 3.4 and in more detail in Lemma A.6. Intuitively, what this function does is take the weighted average over all the possible amounts of decay the earlier link can experience, given the later link is generated at time t.

Deterministic entanglement swapping From the average fidelities of the earlier link and later link at any given time, computing the fidelity at that time after the swap (given the swap succeeds) is straightforward. For the case of deterministic entanglement swapping, having generated two links at layer n of the protocol allows for the deterministic delivery of entanglement on layer $n+1$. Because swapping is assumed to take no time, the corresponding fidelity is then given by

$$F_{n+1}(t) = \text{swap}\left(F_n^{\text{earlier}}(t), F_n^{\text{later}}(t)\right),$$

where the swap function is the function described in Section 3.4, and given in detail in Lemma A.4. Extending this to probabilistic swapping is currently still an open problem.



(a) *Numerically calculated average fidelities using the method in Section 6.2. (runtime ≈ 3 sec. for $N = 8$)*

(b) *Empirical average fidelities, obtained from 270,000 samples generated by Algorithm 7. (runtime ≈ 10 sec. for $N = 8$)*

Figure 6.1: *$F_n(t)$ gives the average fidelity of entanglement delivered at time t by repeater chains with varying number of segments $N = 2^n$, deterministic swapping ($p_{\text{swap}} = 1$), no distillation, and $p_{\text{gen}} = 0.1$. The mean memory lifetime, relevant for the amount of decay waiting links experience, is set here to 50 time steps.*

Chapter 7

Discussion

7.1 Summary

Goal and motivation Our goal was to analyze the waiting time for distributing entanglement in quantum repeater chains. Quantum repeaters form an important building block of quantum networks as they help overcome exponential losses in optical fiber which would otherwise make long distance quantum communication through fibers impossible. Having more detailed information about the waiting time probability distributions for distributing entanglement using repeater chains is useful for example in the investigation of what kind of parameter regimes would be needed to achieve certain distances with these repeater chains. Having full information about this probability distribution can also aid in making design decisions for a repeater protocol, for example when setting entanglement timeout duration.

Model We define the waiting time as the amount of time it takes for a repeater chain which starts with no entanglement to generate entanglement between its end nodes. Our repeater chain model uses the BDCZ protocol for entanglement swapping, and assumes entanglement generation between neighbors, as well as entanglement swapping, are heralded probabilistic operations which succeed with a fixed probability. A heralded distillation scheme following the same nested structure as the BDCZ protocol can also be included in the model when the success probabilities are assumed to be fixed. The communication time for entanglement swapping or distillation are neglected. This model captures most of the core components of a heralded repeaters scheme, although additions can be made to extend or improve this model. These additions are discussed in detail in Section 7.2.

Results We have presented two methods of obtaining information about the probability distribution of this waiting time.

The first approach uses a Monte Carlo simulation to draw waiting time samples which are distributed according to our repeater chain model. Many samples are then aggregated to obtain a numerical approximation of the waiting time probability distribution. Drawing a single sample has an average time complexity which is polynomial in the number of repeater segments. With this approach we are also able to estimate the fidelity of the generated entanglement, taking swap and distillation operations into account, as well as state decay over time.

The second approach is an algorithm for numerically calculating the waiting time distribution up to some truncation value, as well as obtaining tight bounds on the mean. This algorithm produces exact results up to the precision of the numerical operations on the specific machine. The results of this algorithm are verified against the results from the Monte Carlo method. This method has a time complexity which is polynomial in the number of repeater nodes. This improves on previous methods which numerically obtained exact information about the waiting time distribution in exponential time. With this improvement in time complexity we are now able to compute the waiting time distributions for repeater chains with thousands of segments with computation times in the order of minutes on consumer hardware, while previously these distributions have only been obtained for repeater chains with 16 segments. The calculation of fidelity conditioned on time can be integrated in this method when taking the effects of decay over time into account, in combination with deterministic entanglement swapping.

For the characterization of the waiting time and output fidelity the Monte Carlo approach is able to produce results for more detailed models, e.g. the inclusion of fidelity dependent probabilistic entanglement distillation. An advantage the numerical approach still holds over the Monte Carlo simulation is that, even though both of them run in polynomial time, with their current implementations the numerical approach is much faster in practice.

7.2 Future work

In this section we discuss a number of potential improvements to our model and methods, as well as other future steps.

Validation While we have verified our numerical calculations against a simulation of the same model, the question of how well this model compares to reality, i.e. the validity of the model, is still open. To this end one might consider comparing results from our methods against those of a more physically realistic simulation [QuT].

Communication and local operation time In our current model we assume entanglement swapping takes no time at all, while in reality this is not true. Extending our current model and methods to include the communication time between nodes, as well as some fixed amount of time for local operations would narrow the gap between the model and reality. Exploratory investigation into this seem to indicate that including the communication and local operation time to our algorithm is not too difficult to do.

Iterative distillation So far we have assumed that entanglement distillation follows the same nested structure as entanglement swapping. For distillation to actually follow this structure a large amount of parallelism on the quantum channel between any two nodes is required, as well as the need for the nodes to be able to store large amounts of qubits. An iterative distillation protocol which reduces the required parallel resources, sometimes called *entanglement pumping*, has also been proposed [BDCZ98, DBCZ99]. Because this protocol requires significantly fewer parallel quantum resources it might be more practical for some physical implementations of quantum repeaters, and hence it could be worth extending our model and methods with this distillation protocol.

Entanglement rate and idling repeater nodes In our current model, repeater chain nodes which have delivered their share of the total required entanglement will idle until it turns out that a following swap has failed and the entanglement in question needs to be regenerated. A higher overall entanglement rate could be achieved when idling repeater nodes would instead continue to generate entanglement, on the assumption that a swap or distillation attempt can fail in the future. Because of this it is also not the case that the rate at which entanglement can be distributed in a repeater chain is necessarily equal to the inverse of the average waiting time $\mathrm{E}\left[T_n\right]^{-1}$, as that would mean the entanglement generation for a second link would only start when the repeater chain has delivered the first link. In practice however this means that before finishing distributing the first link, and starting to generate entanglement for the second link, some components of the repeater chain are idling, thus having a lower overall rate than would be possible. Future work could potentially focus on analyzing the entanglement rate for a repeater chain model in which the individual components are constantly attempting to generate and distribute entanglement rather than idling when their job temporarily done.

Optimizing entanglement timeout duration Having full information about the probability distribution of generating entanglement could be useful for optimizing the duration of entanglement timeouts. These timeouts enforce a maximum storage time of generated links, and discard these links if this time limit is reached. By limiting the amount of time links are stored the amount of decay they experience is also limited. However, regenerating entanglement after discarding it takes time, and so these entanglement timeouts create a trade-off between fidelity and waiting time. Ultimately what needs to be optimized is not either fidelity or waiting time individually, but rather a function of both (a common metric used for this the secret key rate). Knowing exactly what the probability is of generating the next link within the next t timesteps could help in the decision of whether to discard a currently decaying link. Knowing how the expected fidelity changes as a function of the time it takes to generate a link is also useful for optimizing these cut-offs.

Fidelities, state decay, and entanglement distillation Combining state decay over time and entanglement distillation into a method for computing both the waiting times as well as corresponding fidelities has proven difficult. The main challenge here is the two-way dependence between fidelity and waiting time. The fidelity affects the waiting time because the success probability of distillation depends on the fidelity (for common distillation schemes like [BBP$^+$96] or [DEJ$^+$96]). At the same time the waiting time affects fidelity, because when already generated links spend time waiting for other links to generate, the already generated links decay over time, worsening their fidelity. While for simulations this two-way dependence between waiting time and fidelity does not pose a fundamental problem, incorporating this into our numerical methods remains an open problem.

Bibliography

[ALW+17] F Kimiaee Asadi, N Lauk, S Wein, N Sinclair, C O'Brien, and C Simon. Quantum repeaters with individual rare-earth ions at telecommunication wavelengths. *arXiv preprint arXiv:1712.05356*, 2017.

[AW92] Joseph Abate and Ward Whitt. Numerical inversion of probability generating functions. *Operations Research Letters*, 12(4):245–251, 1992.

[BB84] Charles H Bennett and Gilles Brassard. Quantum cryptography: public key distribution and coin tossing int. In *Conf. on Computers, Systems and Signal Processing (Bangalore, India, Dec. 1984)*, pages 175–9, 1984.

[BB88] E Oran Brigham and E Oran Brigham. *The fast Fourier transform and its applications*. prentice Hall Englewood Cliffs, NJ, 1988.

[BBC+93] 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. *Physical review letters*, 70(13):1895, 1993.

[BBP+96] Charles H Bennett, Gilles Brassard, Sandu Popescu, Benjamin Schumacher, John A Smolin, and William K Wootters. Purification of noisy entanglement and faithful teleportation via noisy channels. *Physical review letters*, 76(5):722, 1996.

[BDCZ98] 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.

[BFK09] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 517–526. IEEE, 2009.

[BPvL11] Nadja K Bernardes, Ludmiła Praxmeyer, and Peter van Loock. Rate analysis for a hybrid quantum repeater. *Physical Review A*, 83(1):012323, 2011.

[BRP+10] Jonatan B Brask, Ioannes Rigas, Eugene S Polzik, Ulrik L Andersen, and Anders S Sørensen. Hybrid long-distance entanglement distribution protocol. *Physical review letters*, 105(16):160501, 2010.

[BS08] Jonatan Bohr Brask and Anders Søndberg Sørensen. Memory imperfections in atomic-ensemble-based quantum repeaters. *Physical Review A*, 78(1):012350, 2008.

[CEHM99] JI Cirac, AK Ekert, SF Huelga, and Chiara Macchiavello. Distributed quantum computation over noisy channels. *Physical Review A*, 59(6):4249, 1999.

[DB07] Wolfgang Dür and Hans J Briegel. Entanglement purification and quantum error correction. *Reports on Progress in Physics*, 70(8):1381, 2007.

[DBCZ99] W Dür, H-J Briegel, JI Cirac, and P Zoller. Quantum repeaters based on entanglement purification. *Physical Review A*, 59(1):169, 1999.

[DEJ+96] David Deutsch, Artur Ekert, Richard Jozsa, Chiara Macchiavello, Sandu Popescu, and Anna Sanpera. Quantum privacy amplification and the security of quantum cryptography over noisy channels. *Physical review letters*, 77(13):2818, 1996.

[Dev86] Luc Devroye. *Non-uniform random variate generation*. Springer-Verlag, 1986.

[DLCZ01] L-M Duan, MD Lukin, J Ignacio Cirac, and Peter Zoller. Long-distance quantum communication with atomic ensembles and linear optics. *Nature*, 414(6862):413, 2001.

[Eis08] Bennett Eisenberg. On the expectation of the maximum of iid geometric random variables. *Statistics & Probability Letters*, 78(2):135–143, 2008.

[Eke91] Artur K Ekert. Quantum cryptography based on Bell's theorem. *Physical review letters*, 67(6):661, 1991.

[ES10] Brian S Everitt and Anders Skrondal. *The Cambridge dictionary of statistics*. Cambridge University Press, 4th edition, 2010.

[GKP94] Ronald L Graham, Donald E Knuth, and Oren Patashnik. *Concrete mathematics: A foundation for computer science*. Addison-Wesley, 2nd edition, 1994.

[Gri02] Gleb Gribakin. Lecture notes probability and distribution theory, chapter 3: probability generating functions. `http://www.am.qub.ac.uk/users/g.gribakin/sor/Chap3.pdf`, Queen's University Belfast, 2002.

[JTL07] L Jiang, JM Taylor, and MD Lukin. Fast and robust approach to long-distance quantum communication with atomic ensembles. *Physical Review A*, 76(1):012301, 2007.

[KKB+14] 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, 2014.

[KMSD19] Sumeet Khatri, Corey T Matyas, Aliza U Siddiqui, and Jonathan P Dowling. Practical figures of merit and thresholds for entanglement distribution in large-scale quantum repeater networks. *arXiv preprint arXiv:1905.06881*, 2019.

[MATN15] William J Munro, Koji Azuma, Kiyoshi Tamaki, and Kae Nemoto. Inside quantum repeaters. *IEEE Journal of Selected Topics in Quantum Electronics*, 21(3):78–90, 2015.

[NC11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011.

[QuT] QuTech. NetSquid. A simulation platform for quantum networks. `https://netsquid.org/`. (In development).

[RGR+18] F Rozpedek, K Goodenough, J Ribeiro, N Kalb, V Caprara Vivoli, Andreas Reiserer, R Hanson, S Wehner, and D Elkouss. Parameter regimes for a single sequential quantum repeater. *Quantum Science and Technology*, 3(3):034002, 2018.

[SDRA+07] Christoph Simon, Hugues De Riedmatten, Mikael Afzelius, Nicolas Sangouard, Hugo Zbinden, and Nicolas Gisin. Quantum repeaters with photon pair sources and multimode memories. *Physical review letters*, 98(19):190503, 2007.

[SDS09] Nicolas Sangouard, Romain Dubessy, and Christoph Simon. Quantum repeaters based on single trapped ions. *Physical Review A*, 79(4):042340, 2009.

[SJM18] S Santra, L Jiang, and V Malinovsky. Quantum repeater architecture with hierarchically optimized access times. *arXiv preprint arXiv:1811.01080*, 2018.

[SSDRG11] Nicolas Sangouard, Christoph Simon, Hugues De Riedmatten, and Nicolas Gisin. Quantum repeaters based on atomic ensembles and linear optics. *Reviews of Modern Physics*, 83(1):33, 2011.

[SSM+07] Nicolas Sangouard, Christoph Simon, Jiří Minář, Hugo Zbinden, Hugues De Riedmatten, and Nicolas Gisin. Long-distance entanglement distribution with single-photon sources. *Physical Review A*, 76(5):050301, 2007.

[SSvL17] E Shchukin, F Schmidt, and P van Loock. On the waiting time in quantum repeaters with probabilistic entanglement swapping. *arXiv preprint arXiv:1710.06214v4*, 2017.

[SSZ+08] N Sangouard, C Simon, B Zhao, Y Chen, H De Riedmatten, J Pan, and N Gisin. Robust and efficient quantum repeaters with atomic ensembles and linear optics. *Physical Review A*, 77(6):062301, 2008.

[TKM05] Seiichiro Tani, Hirotada Kobayashi, and Keiji Matsumoto. Exact quantum algorithms for the leader election problem. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 581–592. Springer, 2005.

[VK17] Scott E Vinay and Pieter Kok. Practical repeaters for ultralong-distance quantum communication. *Physical Review A*, 95(5):052336, 2017.

[VK19] Scott E. Vinay and Pieter Kok. Statistical analysis of quantum-entangled-network generation. *Phys. Rev. A*, 99:042313, Apr 2019.

[Wal45] Abraham Wald. Sequential tests of statistical hypotheses. *The annals of mathematical statistics*, 16(2):117–186, 1945.

[Was06] Larry Wasserman. *All of nonparametric statistics*. Springer Science & Business Media, 2006.

Appendix A

Fidelity tracking

In order to estimate or compute fidelity of entanglement generated by a repeater chain as a function of time, we need to make a number of assumptions. This appendix goes over all assumptions we make with regard of quantum states, operations (distillation and swaps), and state decay over time.

In short we assume Werner states as input for all operations, distillation is done using the BBPSSW protocol [BBP$^+$96], swapping is done using the standard teleportation scheme [BBC$^+$93], and for state decay over time we assume exclusively depolarizing noise.

Fidelity The fidelity F of a quantum state ρ is a measure of how close your actual state is to some desired (target) state. In Definition A.1 the fidelity is given when this target state is a pure state. While fidelity is a metric for the quality of a state and not strictly for the amount of entanglement, when the target state is a maximally entangled two-qubit state (often a Bell state), we can state that we have discernible entanglement when $F > 0.5$.

Definition A.1. *For some quantum state ρ, and some desired pure state $|\psi\rangle$, the fidelity F of ρ is determined by the distance between these states, given by*

$$F = \langle\psi| \rho |\psi\rangle .$$

Werner states Our goal is to compute fidelities of two-qubit entangled states. For this purpose we make the assumption that all our states are Werner states (Def. A.2). These Werner states have two important properties:

- A Werner state is completely defined by its fidelity. This means that when we keep track of fidelities, we are also keeping track of the entire quantum state. Having the complete quantum state helps when determining what effect certain operations have on the fidelity.
- Any two-qubit state can be "twirled" to a Werner state with the same fidelity [BBP$^+$96, DB07]. This twirling involves applying random gates to the state, effectively discarding information about the state until all that remains is a Werner state.

Definition A.2. *A Werner state is a two qubit state ρ which can be written entirely as a function of its fidelity F and a given target Bell state (here Φ^+):*

$$\rho_W(F) = F\Phi^+ + \left(\frac{1-F}{3}\right)\left(\Phi^- + \Psi^+ + \Psi^-\right), \tag{A.1}$$

where $\Phi^{\pm} = |\Phi^{\pm}\rangle\langle\Phi^{\pm}|$ *and* $\Psi^{\pm} = |\Psi^{\pm}\rangle\langle\Psi^{\pm}|$*. This state can also be written as*

$$\rho_W(F) = \alpha\Phi^+ + (1-\alpha)\frac{\mathbb{I}}{4}. \tag{A.2}$$

where

$$\alpha = \frac{4F-1}{3}.$$

Distillation Entanglement distillation, also referred to as entanglement purification, is a procedure which takes two (or more) imperfect entangled pairs of qubits and (probabilistically) "distills" a single pair with higher fidelity from them. Here we assume all entanglement distillation follows the BBPSSW protocol [BBP$^+$96], which is a distillation protocol which takes two Werner states as input. Lemma A.3 gives both the output fidelity as well as the probability of success as a function of the input fidelities for this protocol.

Lemma A.3. *For two Werner states with fidelity* F_1 *and* F_2*, the fidelity after a successful distillation attempt using the BBPSSW protocol [BBP*$^+$*96] is given by the expression below [DB07]. The corresponding success probability of the distillation attempt is given by the denominator.*

$$F' = \frac{F_1F_2 + \left(\frac{1-F_1}{3}\right)\left(\frac{1-F_2}{3}\right)}{F_1F_2 + F_1\left(\frac{1-F_2}{3}\right) + \left(\frac{1-F_1}{3}\right)F_2 + 5\left(\frac{1-F_1}{3}\right)\left(\frac{1-F_2}{3}\right)}. \tag{A.3}$$

Entanglement swapping Because we know that the input states of any swap operation in our model are two Werner states with fidelities F_1 and F_2, we can exactly compute what the fidelity of the output state will be. We can do this by looking at all possible pairs of pure input states, and which output state would be produced when applying the quantum teleportation scheme [BBC$^+$93] on them. Table A.1 gives an overview of these input and corresponding output states, as well as the probability of having these states as input given out state descriptions are $\rho_W(F_1)$ and $\rho_W(F_2)$. Using Table A.1, we can find the output fidelity as a function of the input fidelities, which is given in Lemma A.4. An equivalent expression for the case where $F_1 = F_2$ is also given in [BDCZ98, DBCZ99].

Lemma A.4. *Given two Werner states* $\rho_W(F_1)$ *and* $\rho_W(F_2)$*, performing an entanglement swap operation on these states (without any gate or measurement noise) using the quantum teleportation scheme [BBC*$^+$*93], produces an output state with fidelity*

$$F' = F_1F_2 + 3\left(\frac{1-F_1}{3}\right)\left(\frac{1-F_2}{3}\right).$$

Proof. Table A.1 gives an overview of all the possible input states, corresponding probabilities and corresponding output states. We can find to total output fidelity as

$$F' = \sum_{\psi,\phi} p_{\text{in}}(\psi,\phi)F_{\text{out}}(\psi,\phi)$$

for $\psi, \phi \in \{\Phi^+, \Phi^-, \Psi^+, \Psi^-\}$. Filling in the non-zero values from Table A.1 gives

$$F' = F_1F_2 + 3\left(\frac{1-F_1}{3}\right)\left(\frac{1-F_2}{3}\right).$$

□

State in	p_{in}	State out	F_{out}
$\Phi^+ \otimes \Phi^+$	$F_1 F_2$	Φ^+	1
$\Phi^+ \otimes \Phi^-$		Φ^-	0
$\Phi^+ \otimes \Psi^+$		Ψ^+	0
$\Phi^+ \otimes \Psi^-$		Ψ^-	0
$\Phi^- \otimes \Phi^+$		Φ^-	0
$\Phi^- \otimes \Phi^-$	$\left(\frac{1-F_1}{3}\right)\left(\frac{1-F_1}{3}\right)$	Φ^+	1
$\Phi^- \otimes \Psi^+$		Ψ^-	0
$\Phi^- \otimes \Psi^-$		Ψ^+	0
$\Psi^+ \otimes \Phi^+$		Ψ^+	0
$\Psi^+ \otimes \Phi^-$		Ψ^-	0
$\Psi^+ \otimes \Psi^+$	$\left(\frac{1-F_1}{3}\right)\left(\frac{1-F_1}{3}\right)$	Φ^+	1
$\Psi^+ \otimes \Psi^-$		Φ^-	0
$\Psi^- \otimes \Phi^+$		Ψ^-	0
$\Psi^- \otimes \Phi^-$		Ψ^+	0
$\Psi^- \otimes \Psi^+$		Φ^-	0
$\Psi^- \otimes \Psi^-$	$\left(\frac{1-F_1}{3}\right)\left(\frac{1-F_1}{3}\right)$	Φ^+	1

Table A.1: *All possible combinations of input Bell states and their corresponding output state after entanglement swapping. The column F_{out} gives the fidelity of output state in relation to the target state Φ^+. The column p_{in} gives the probability of this particular combination of input states, given the two input states are described by $\rho_W(F_1)$ and $\rho_W(F_2)$.*

Decay over time For the decay, or noise, the quantum states experience when stored in memory we assume exclusively depolarizing noise (Def. A.5). While different physical systems can suffer from different kinds of noise, depolarization can be seen as a "worst case" scenario, as it affects all possible measurement bases the qubits could be measured in. When applying a depolarizing channel with probability p to a Werner state with some parameter α (as in Eq. (A.2)), this produces a new Werner state with $\alpha' = \alpha \cdot (1 - p)$ (see Lem. A.6). For a link which has spend Δt time in memory, we apply a depolarizing channel with parameter $p = e^{-\lambda \Delta t}$, where λ^{-1} is the characteristic time, or the mean lifetime of the state in the memory.

Definition A.5. *The depolarizing channel $\mathcal{E}_p$ acting on a state ρ is given by*

$$\mathcal{E}_p(\rho) = (1 - p)\rho + p\frac{\mathbb{I}}{d}$$

where d is the dimensionality of ρ, and $0 \leq p \leq 1$ can be seen as the amount of depolarization or the probability with which ρ completely depolarizes (becomes the maximally mixed state $\mathbb{I}/d$) [NC11].

Lemma A.6. *Applying a depolarizing channel $\mathcal{E}_p$ to a Werner state $\rho_W(F)$ produces a another Werner state $\rho_W(F')$, where parameter α (as in the form of Eq.* (A.2)*) becomes $\alpha' = \alpha \cdot (1-p)$.*

Proof. Writing $\rho_W(F)$ in the form of Eq. (A.2) and applying $\mathcal{E}_p$ gives

$$\begin{aligned}\mathcal{E}_p\left(\rho_W(F)\right) &= (1-p)\left(\alpha\Phi^+ + (1-\alpha)\frac{\mathbb{I}}{4}\right) + p\frac{\mathbb{I}}{4} \\ &= (1-p)\alpha\Phi^+ + (1-p)(1-\alpha)\frac{\mathbb{I}}{4} + p\frac{\mathbb{I}}{4} \\ &= (1-p)\alpha\Phi^+ + (1-(1-p)\alpha)\frac{\mathbb{I}}{4} \\ &= \alpha'\Phi^+ + (1-\alpha')\frac{\mathbb{I}}{4},\end{aligned}$$

which is a Werner state with fidelity

$$F' = \frac{3\alpha' + 1}{4}$$

and

$$\alpha' = \alpha \cdot (1-p).$$

□

Appendix B

Random sums using PGFs

The following is an overview of a different method for handling the geometric sum in Equation (5.3). This approach did not yield any improvements in comparison with the results presented in Sections 5.2 and 5.3, however for future reference we summarize this approach here.

Motivation In Sections 5.2 and 5.3 the geometric sum of i.i.d. random variables is expressed as and computed through repeated convolutions of their probability mass functions (Lemma 2.9). This method however presents an obstacle in Section 5.2 because it gives an expression which we cannot work with analytically. And while in Section 5.3 this difficulty is overcome by expressing the probability distributions numerically, it is also found that these repeated convolutions are now the computational bottleneck of the algorithm. Because this geometric sum causes difficulties for both our analytical as well as numerical efforts, it might be worth investigating if this sum can be computed via different means.

Probability generating functions As it turns out, this geometric sum can be expressed in terms of probability generating functions (PGFs) in a more elegant way than Equation (5.3). Definition B.1 gives the definition of the probability generating function of a random variable, and shows its relation to the corresponding probability mass function. Lemma B.2 shows how we can express the geometric sum in terms of PGFs.

Definition B.1. *For a count random variable X with probability mass function $\Pr(X = x)$, the probability generating function (PGF) is defined as*

$$G_X(z) = \mathrm{E}\left[z^X\right] = \sum_{x=0}^{\infty} \Pr(X = x) \cdot z^x.$$

Lemma B.2. *For the random variable $W_N = X_1 + X_2 + \cdots + X_N$, with all X_k i.i.d., and X and N independent count random variables, the PGF of W_N is given by the following expression [Gri02].*

$$G_{W_N}(z) = G_N(G_X(z)).$$

Results for analytical approach While we have the expression from Lemma B.2 for the geometric sum, for the maximum of random variables (the other operation we need to do, see Definition 3.3.1) there exists no expression for probability generating functions.

Instead of evaluating the maximum of random variables in terms of their probability generating functions, we could also translate the PGFs back to the corresponding PMF or CDF. However, even though a probability generating function is unique to its corresponding probability distribution [Gri02], there exists no method for finding a closed form expression of the PMF from an arbitrary PGF. This means that also this route fails to produce a completely analytical solution.

Results for numerical approach Being able to unable to use these PGFs to obtain a completely analytical solution, we might still get use out of it by integrating it in the numerical approach (Section 5.3). The expression in Lemma B.2 can be used to compute the geometric sum via PGFs, rather than via (computationally expensive) convolutions of the PMF. But since we can only do the max operation via the CDF we need to numerically translate the PGFs back to PMFs or CDFs first. A method for doing this is presented in [AW92], which we have implemented and tested. The time complexity of this inversion method is similar that of doing the geometric sum via convolutions, but because the convolutions can easily leverage efficient library implementations they are practically much faster.