Simulating Quantum Scheduler Performance for an Entanglement Generation Switch

QuTech - Wehner Group

Marit Talsma

Delft University of Technology

TU Delft

Simulating Quantum Scheduler Performance for an Entanglement Generation Switch

QuTech - Wehner Group

by

Marit Talsma

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday February 13, 2024 at 14:00H.

Student number:	5420083	
Project duration:	May 1, 2023 – February 13, 2024	
Daily supervisor:	Scarlett Gauthier, MSc.	
Thesis committee:	Prof. Dr. S. Wehner,	TU Delft & QuTech, supervisor
	Dr. G. Vardoyan,	TU Delft & QuTech
	Dr. P. Pawelczak	TU Delft

Cover:	AQT in collaboration with the Quantum Internet Alliance (QIA)
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft | QuTech

Abstract

The quantum internet improves upon the classical internet with several new possibilities. However, to create such a quantum internet, metropolitan hubs (for this research we will use an Entanglement Generation Switch - EGS - as hub) are needed to avoid a scaling problem when connecting all end nodes individually with one another. These EGSs share resources for multiple end nodes. Hence, resource sharing protocols are needed to manage these resources. Unfortunately due to the probabilistic nature of end-to-end entanglement generation, classical resource sharing protocols will not suffice for implementation in a quantum network. Therefore the need to design quantum resource sharing protocols arises. These protocols, where we specifically investigate quantum schedulers, should be simple and predictable to provide a comprehensible addition to current research by incorporating more parameters into the current EGS model while still attempting optimal resource efficiency. In this research we have extended quantum network simulation software to be used in this research and following projects. We have implemented a classical Max Weight scheduler with a constant time window size to simulate the inefficiency of such a classical protocol in a quantum environment and improve upon this protocol by implementing a scheduler with a dynamic time window size which scales with the link length and the probability of a photon arriving at the EGS. We investigate the entanglement successes, idle time and entanglement rates when using these schedulers and compared these with each other and with a third scheduler without a time window size but using a cutoff timer instead, the Cutoff scheduler. These results show an optimality of entanglement successes, idle time and entanglement rates for the Cutoff scheduler over the Dynamic Time Window scheduler and for the Dynamic Time Window scheduler over the Max Weight scheduler with a constant time window size. Also a decrease in entanglement rate for all schedulers is shown when increasing the link length.

Acknowledgments

This thesis is a show of the research I did for the past nine months, what I have learned in this period and ultimately it is the conclusion of my master at TU Delft. Achieving all of this truly would not have been possible without the community around me and for that I want to express my gratitude.

Firstly I would like to thank my supervisor, Prof. Stephanie Wehner for giving me the opportunity to work on such an interesting subject and for all her advice during this work. I would also like to thank Dr. Przemyslaw Pawelczak and Dr. Gayane Vardoyan for being on my thesis committee.

I especially want to thank my daily supervisor Scarlett Gauthier for her advice and support during this thesis. Finding the right direction to take with my research given the possibilities at the time sometimes required a bit of puzzling which I am glad to say we did together happily. I do not only want to thank you for your help with my research, but also for teaching me about so much more than the subject of my thesis and always keeping our conversations fun. I would also like to thank Michal van Hooft for his help in understanding the software setup. Thanks to my friends and family who have supported me and have always pushed me to be my best. Working on my master and on my thesis has been with incredible ups but also some downs and I want to express my gratitude to everyone who has supported me through this entire experience. I also want to specifically mention all members of the Wehner and Vardoyan group and all people of the master office for creating such a welcoming and dynamic environment at QuTech. You all made it incredibly fun going to work every day.

Nomenclature

Abbreviations

Abbreviation	Definition
AET	Average Entanglement Time
DTW	Dynamic Time Window
EDF	Earliest Deadline First
EGS	Entanglement Generation Switch
FIFO	First-In-First-Out
MW	Max Weight
QIA	Quantum Internet Alliance
RCP	Rate Control Protocol
RR	Round-Robin
TW	Time Window
TWP	Time Window Prefix

Symbols

Symbol	Definition	Unit
p_{emit}	Probability of travelling photon generation	
p_{init}	Initiation probability of a qubit pair in an end node	
$P_{success}$	Percentage of successfully entangled qubits in one time window	
$p_{travel,tot}$	Arrival probability of photons from both end node arriving at the EGS	
r_{ent}	Entanglement generation rate	
t_{bsm}	Bell state measurement time	[s]
t_{init}	Initiation time of a qubit pair in an end node	[s]
t_{emit}	Travelling photon generation time	[s]
$t_{ent,avg}$	Average entanglement time of one qubit pair	[s]
t_{switch}	Optical switch time	[s]
t_{travel}	Two way travel time from an end node to the EGS	[s]
w	Time window size	[s]

Contents

1 Introduction

Classical networks like the internet have become crucial for connecting people all over the world. However the classical computer networks currently used have various limitations [11, 31, 33]. Quantum networking, and with this the quantum internet, has the possibility of solving some of these problems and allows for new functionalities, some of which are secure communication [3, 10] and quantum cloud computing [4, 1]. However, many elements needed for the use of a quantum internet are not ready yet for real world use.

Quantum hardware is thus far limited to small, error-prone quantum systems [16, 22] and to date network demonstrations have only connected few users via dedicated links that privately provided all of the resources needed for entanglement generation. Such a setup with dedicated links grows as $\binom{N}{2}$ (where N is the number of nodes in the network) when connecting all end nodes with each other. To solve this scaling problem, quantum networks may include hubs with shared resources to generate end-to-end entanglement which instead scales as R (with R being the number of resource nodes of the hub). However, not enough research has been done on the possible improvements upon classical networking protocols needed or possibly even the need for completely new protocols to be able to implement such hubs in a working quantum internet. In Ref. [12] the authors develop an architecture for such a hub - an EGS - and propose a resource sharing protocol for managing the entanglement rates demanded of an EGS by the end nodes, namely a Rate Control Protocol (RPC). This protocol should then be used in combination with a scheduler at the EGS, where in Ref. [12] a Max Weight scheduler with constant time window sizes was used. This setup has been tested using numerical evaluation. In this work however the authors make several assumptions, one of which is the assumption of constant entanglement success probability for all end nodes using the EGS. To contribute to this research we create a more realistic setup which takes varying link lengths for the different end nodes to the EGS into account thus simulating different entanglement success probabilities for these different end nodes. With this setup we look into the effect of link length on the end-to-end entanglement rate when using an EGS with various types of schedulers. This research is intended to further the development of a quantum network which uses an EGS by creating a more realistic model and by contributing to a software framework for simulating such a network.

In this thesis we investigate the following research questions:

1. What is the efficacy, in terms of well-defined performance metrics, of scheduler implementations for an EGS without fixed time window sizes in comparison to a scheduler with fixed time window sizes?
2. Is there an optimal time window size for quantum schedulers when attempting end-to-end entanglement?
3. What is the influence of link length on the entanglement rate?

In this thesis we investigate various Max Weight schedulers and their optimal time window sizes for generating end-to-end entanglement between end nodes using an intermediate quantum network hub. We weigh their implementation difficulty to their optimality in terms of throughput and look at the implications of the entanglement rates of these schedulers for rate control protocols. The contributions of this thesis are as follows:

- We quantify the entanglement rates for various link distances generated when using a Max Weight scheduler with various constant time windows. This will give us a time window size for which entanglement rates are optimal in terms of throughput given our setup. These entanglement rates are important to measure the need for protocol adjustment (for the RPC and the scheduling protocol) when removing the assumption that all distances between end nodes and the EGS, and therefore all entanglement success for various sessions, are identical as assumed in Ref. [12].
- We introduce two new schedulers, a Dynamic Time Window scheduler and a Cutoff scheduler which could improve upon the entanglement rates possible with the MW scheduler when extending EGS research to a more realistic setup. The relevance of investigating schedulers other than the MW scheduler with constant time window size comes from the assumption that this MW scheduler will not be optimal for use in a realistic quantum network model with non-homogeneous link lengths. Therefore other schedulers should be considered that will take into account the decreased probability of successful entanglement when increasing link lengths. The allowed execution time of one request using either one of the two proposed schedulers scales with the link length and therefore could provide a solution to this optimization problem.
- We provide a software setup integrated with SquidASM that includes the MW and DTW schedulers and a new optical fibre link type implementation. These software contributions provide researchers with more tools to investigate various quantum network protocols and with a software setup that behaves more like a real-world network than had been implemented thus far. This software is intended to be used specifically for research on the Quantum Internet Alliance's (QIA) first full-stack quantum internet prototype network [23]. QIA is a research collaboration with partners across Europe sharing the goal of building a prototype quantum network by 2029.

In this thesis we will first explain some of the background concepts relevant to this research in Chapter 2. Specifically, we describe some classical networking concepts in Sec. 2.1, and a quantum internet network architecture and a detailed description of EGSs in Sec. 2.2 and 2.3, respectively. In this background chapter we also take a look at the Heralded Single Click scheme for bi-partite entanglement generation used in this research (Sec. 2.4), the constraints on entanglement rates in our setup (Sec. 2.5) and the quantum simulation software packages used (Sec. 2.7). Our evaluation metrics are explained in Sec. 2.6. Related work is described in Chapter 3 and the model used for the simulations in Chapter 4, including the definition of the time window sizes (Sec. 4.2). Sec. 4.3 shows the expected results. Chapter 5 then describes the contributions made to the SquidASM simulation software and the software implementation for our research. Results addressing our research questions are shown and discussed in Chapter 6. We conclude in Chapter 7 by looking back at the implications of these results and possible directions for further research.

2 Background

The future quantum internet will be a classical-quantum hybrid network. Such a network uses a classical control plane and a quantum data plane. In other words, such a network uses classical control messages to facilitate end-to-end entanglement between end nodes. Research has already been done on the implementation of such a quantum network [6], but hardware is not yet able to facilitate its use on a large scale [5, 22]. Investigations on necessary network capabilities of a quantum internet and what it could look like are described in Ref. [18, 24, 32]. In these works the authors provide overviews of expectations of quantum network requirements for various stages of development of the quantum internet and the corresponding design choices to be made.

2.1. Classical Networking

Because of decades of research on classical networking there are nowadays extremely many and various networking protocols. Simple protocols which formed the backbone of the first generations of the classical internet are (often) replaced by optimized versions of these protocols, if not completely new ones. Many of these protocols are dedicated to managing resource sharing where users can request resources from resource sharing stations. This is needed as connecting users directly scales as $\binom{N}{2}$ with N being the number of nodes in the network. Schedulers are one such type of protocol, managing this resource sharing locally for one resource sharing station. These schedulers are responsible for assigning these shared resources to users.

Looking at classical schedulers there are now many types – First-In-First-Out (FIFO), Max Weight (MW), Earliest Deadline First (EDF), Round Robin (RR) to name a few [25] – all with their own advantages and disadvantages. The schedulers should assign the shared resources in such a way that all requirements are met, depending on what scheduling policy is used. This could for some schedulers mean that the number of deadlines met is optimized (EDF) while for others it ensures that no users will starve (RR).

Max Weight Scheduler

Fig. 2.1 shows a representation of a classical resource sharing station using a Max Weight (MW) scheduler with four users. Here the grey squares represent the users which are connected to the station. An example of such a station is a server shared by users which request that applications be executed. The four bins represent the request queues of the MW scheduler, one for each user which submits requests. These requests specify the requirements given by the user. This can include various details including for instance the periodicity and priority required. The queued requests are shown in this figure as the green and orange coloured squares. The MW scheduler at the station is responsible for assigning its resources to requests and does this based on which request queue is the largest. In Fig. 2.1 this would be the queue belonging to user number 1, since this queue has the largest number of queued requests, namely two.

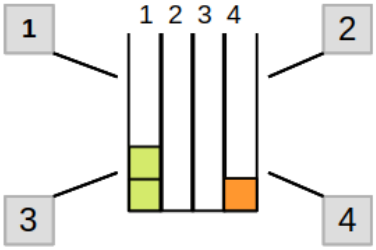


Figure 2.1: Representation of a classical resource sharing station using a Max Weight scheduler (center) with four users (squares).

2.1.1. Taxonomy of Schedulers

Not only are there various types of schedulers, these schedulers can also vary in implementation. For instance one can choose for preemtive scheduling where one allows a job to be cut off when a higher priority comes along or one can choose to obey a soft rather than a firm deadline of requests which allows for a schedule acknowledging the different impacts of small versus large consequences of missing the deadline, respectively. Another source of variation is static versus dynamic scheduling, where for static scheduling the schedule is known prior to starting the scheduler and for dynamic scheduling the schedule is calculated in real-time. The difference in implementation relevant to us is the use of constant time windows versus dynamic time windows. Constant time windows have a constant, predefined duration whereas the dynamic time windows do not have a constant duration and can vary per request or user. The schedulers used in this research and additional design choices are described in Sec. 4.1.

2.2. Network Architecture

A quantum network generally consists of end nodes, quantum repeaters, network hubs and elementary links. One such network topology can be seen in Fig. 2.2 and an experimental realization of a simple quantum network can be found in Ref. [21]. End nodes are responsible for the preparation and measurement of qubits. These nodes consist of at minimum a communication qubit and possibly a quantum memory or additional communication qubits. Quantum repeaters are responsible for entangling travelling photons to generate entanglement at longer distances. Quantum network hubs connect multiple end nodes through shared resources. For this research we have focused on a specific type of quantum network hub, namely an EGS. More on the workings of this EGS can be found in Sec. 2.3. Gluing this entire system together by connecting these elements with each other are the elementary links, which we will refer to simply as links. These links can be of various types, including optical fibre links, free space links and satellite links. Here we consider optical fibre links since we aim to contribute to QIA, which aims to build a prototype network where links connecting end-nodes to hubs will be over optical fibre.

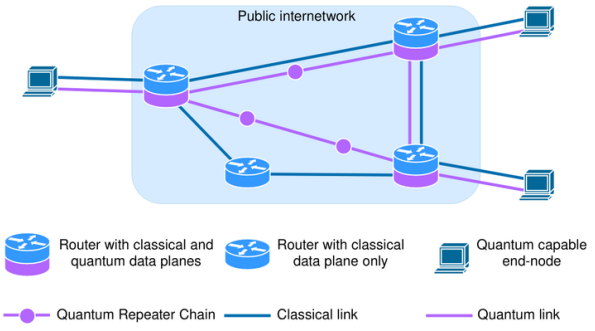


Figure 2.2: Example topology of a quantum network. Appears originally in [15].

2.3. Entanglement Generation Switch

An EGS is a type of quantum network hub. As defined in Ref. [12], an EGS allocates resources to subsets of nodes which may directly attempt entanglement generation using the allocated resources. A schematic example of an EGS is shown in Fig. 2.3. This figure shows nine end nodes with a classical (yellow part) and quantum mechanical (green) component connected using optical fibre links to the classical processor (or controller) and quantum mechanical part of the EGS, respectively. The controller of the EGS is responsible for handling the control plane protocols of the EGS. The other elements of the EGS - its quantum mechanical part - are responsible for handling the data plane. This part consists of an optical switch (shown in Fig. 2.3 as 'Switch') and a limited number of resource nodes (purple). Given that the EGS will generally be connected to more end nodes than it has resource nodes and therefore will not have a one-to-one connection of end nodes to resource nodes, the EGS controller will be responsible for some form of scheduling of the incoming entanglement requests. Because of this the EGS also needs a system to allocate these resource nodes to the correct end node pairs. The actual connection process is facilitated by the optical switch, which is connected to all end nodes and to all resource nodes and relays the connection to the required resource node to the correct end nodes. In these resource nodes Bell state measurements can then be executed after which the controller sends back a heralding pulse to the relevant end nodes (Ref. [13]). The heralded protocol we use in this research is explained in Sec. 2.4. Also note that this EGS does not have memory qubits and is therefore not able to store qubits.

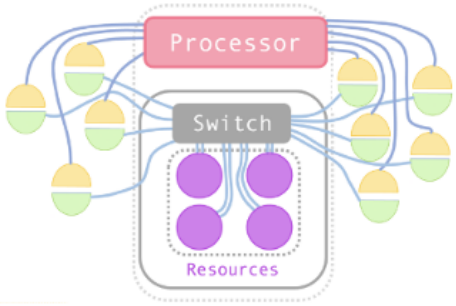


Figure 2.3: EGS with four resources connected to nine end nodes. Appears originally in [12].

In [12] pairs of nodes are called communication sessions. Each unique pair is indexed by a label s. Communication sessions demand some rate of entanglement generation λ_s per time window. Sec. 2.5 describes bounds to which this entanglement generation rate has to adhere. An EGS has a certain capacity region $\mathcal{C}$ describing the constraints on the ability of the EGS to deliver a requested set of rates. In [12] the authors prove that $\mathcal{C}$ states that if the specified constraints are met, then there exists a scheduler for which the rate at which jobs are removed from the queue is greater than the rate at which they are added. This capacity region is defined as:

$$\mathcal{C} = \left\{ \boldsymbol{\lambda} : \boldsymbol{\lambda} \geq 0, \quad \sum_s \lambda_s \leq \lambda_{EGS} \quad \lambda_s \leq \lambda_{gen,s}^{max} \; \forall \; s \in S \right\} \tag{2.1}$$

Here $\boldsymbol{\lambda}$ is the rate vector with components λ_s from all sessions s. λ_{EGS} is then defined as the maximal achievable entanglement rate of the EGS, $\lambda_{EGS} = R \cdot p_{gen}$ with R being the number of resource nodes. p_{gen} is the entanglement generation success probability of a session and is assumed to be equal for all sessions in Ref. [12]. $\lambda_{gen,s}^{max} = x_s^* \cdot p_{gen}$ describes the maximal entanglement request rate per session where x_s^* is the maximum number of resource nodes that can be allocated per session per time slot. This capacity region is independent of the number of end nodes requesting entanglement and depends only on their requested entanglement rates. When a control algorithm ensures that a system operates within its capacity region, the algorithm is said to stabilize the system.

To addess the resource scheduling problem for an EGS, Ref. [12] proves the optimality of a MW scheduler in terms of throughput and develops the RCP to stabilize EGS operation under MW scheduling. Max Weight Schedulers for classical networks [26] and entanglement distribution switches in quantum networks [17] have previously been examined and proven to exhibit optimal performance in terms of throughput (see Ref. [30]). Sec. 2.1.1 explains some of the additional design choices of schedulers.

2.4. Heralded Single Click Protocol

The Heralded Single-Click protocol lets the to-be-entangled end nodes each create a qubit pair of which one stays stored in the state of the communication qubit of the end node and the other qubit is sent over a link to an entanglement swapping station, in our case an EGS resource, in the form of a photon. When the EGS then receives the photons from both end nodes it performs a Bell state measurement which will create entanglement if successful. A heralding pulse is then sent to the nodes informing them of the result of the successful, or possibly unsuccessful, entanglement swap. If the attempt for end-to-end entanglement creation was not successful the protocol can be repeated.

2.5. Entanglement Rate Constraints

In a setting where pairs of nodes demand rates of entanglement generation, the end nodes supply certain entanglement rate constraints which have to be taken into account by the EGS. A minimal entanglement rate λ_{min} has to be adhered to in order to ensure usability of the target application at the end node. An upper limit λ_{max} of the attempt rate is due to technical limitations in the end nodes, for instance at qubit initiation. When scheduling it is useful to have an expected entanglement rate in order to guarantee with decent certainty to the end nodes that their entanglement rate will fall in this rate region and therefore their target application can be executed as intended. Knowing the average end-to-end entanglement rate and the entanglement rate limitations can be relevant for decision making at the EGS. This can be done by for instance by only accepting a request if multiple resource nodes are available to up the entanglement rate so that the entanglement rate falls within its constraints.

2.6. Evaluation Metrics

The evaluation metric used for this research is throughput. In our case of quantum networking this is quantified by the end-to-end entanglement rate. Entanglement rate r_{ent} is determined by the percentage of successful entanglements generated $P_{success}$ divided by the time window size w:

$$r_{ent} = P_{success}/w \tag{2.2}$$

The time windows per type of scheduler are described in Sec. 4.1. Since for the Cutoff scheduler the time window sizes change depending on how fast entanglement is generated (see Sec. 4.1), we used the average time window size w_{avg}.

One other possible evaluation metric for quantum networks is fidelity of generated entanglement [19]. The fidelity is a measure for the indistinguishability of two quantum states. In a quantum network this fidelity then is a measure of how likely it is that two entangled states will collapse to the same definite state when measured. However, although it is noteworthy to mention that fidelity influences the performance of quantum network applications, we do not investigate this evaluation metric in this work and leave it for further research.

2.7. NetSquid and SquidASM

NetSquid [7] is a discrete event simulator used for simulating quantum networks. This software package, in combination with Python, can be used to model an accurate representation of a quantum network taking into account effects like decoherence. Built on top of NetSquid is SquidASM [14], another software package, providing a more elaborate framework for quantum network simulation. In SquidASM various models can be directly used, including models for end nodes and link types. SquidASM also contains functionality for automating the setup of certain network topologies.

3

Related Work

This work is an extension of a theoretical paper (Ref. [12]) proposing a RPC for entanglement generation between nodes using an EGS. In this paper the authors prove that MW schedulers provide throughput optimal service for such an EGS. For this MW scheduler, Ref. [12] used a constant time window size to model a synchronous network and enable studying a synchronous protocol for resource allocation and rate adjustment. Doing this however, a number of assumptions were made in order to enable working in the slotted time model. One example is that the entanglement generation probability during one time window (p_{gen}) was assumed to be equal for all sessions, meaning that end nodes were assumed to be identical and the distance of each link was assumed to be equal. Using our research we come one step closer to an asynchronous real-world model where the link lengths are taken into account when calculating p_{gen}.

Ref. [8] identifies the problem that not only throughput is important for the implementation of a quantum switch (in this case with memory), but also the stability of the switch while using scheduling policies. With this in mind the authors look into some design choices for scheduling protocols. Like in Ref. [12], Ref. [8] uses constant time windows as well. Research has also been done on the capacity of other quantum network switches (see Ref. [29, 20]) which also use constant time windows for their schedulers. We anticipate that our research into schedulers for asynchronous models will not only be relevant for the scheduling protocols of the EGS, but also for more widespread research into quantum switch protocols like the ones mentioned above. In this thesis we focus on the efficiency of various schedulers in terms of throughput and with this intend to provide a comprehensive baseline for investigating more realistic, general quantum network switch models. This might include combining the schedulers developed in this research with an upgraded version of the RCP in order to achieve stability.

SquidASM [14] is used as base code for our simulation implementation. It provides us with several blueprint classes for schedulers, link protocols and end node protocols and it contains some complete implementations of a quantum network including construction classes. We will use some blueprint classes and setup functionalities of SquidASM for our implelemtation, of which the detailed description can be seen in Sec. 5.3.

4

Model

The model used for these experiments is that of a single EGS connected to two Nitrogen-Vacancy end nodes with symmetric links. The basic architecture of the EGS is described in Sec. 2.3. For our experiments we used an EGS which consists of a controller, an optical switch and one single resource node. Fig. 4.1 shows such a model, where the grey squares represent the end nodes, the solid and dotted black lines represent the quantum and classical links, respectively, and in the center of the purple hub of this figure one can see the controller and the connection of the quantum links to this one resource node via an optical switch. The end nodes are connected with links for classical communication to the EGS controller and with links for quantum communication to the resource node. These are simple optical fibre links.

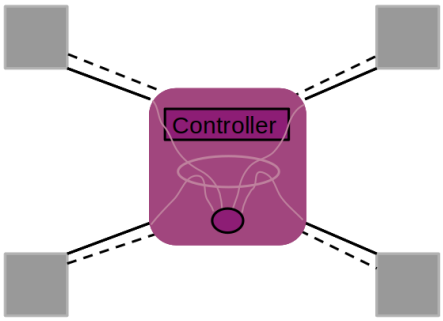


Figure 4.1: Four end nodes (grey) classically (dotted black lines) and quantum mechanically (solid black lines) connected to an EGS (purple) with a controller, optical switch and one resource node.

Whenever two end nodes want to create entanglement using the EGS resources, these end nodes first communicate this classically among each other. This is assumed to be done before a request is sent to the EGS and these requests will only be sent from one of these two end nodes in our model. Requests are for one single entangled pair. To generate this one single end-to-end entangled pair batched attempts are made and the time windows are set large enough to facilitate these batches. The requests are placed in the queue of the scheduler, where the order of execution of these requests is defined by the type of scheduler used. We use dynamic Max Weight schedulers for which Sec. 2.1 describes its workings. Also this scheduling, which is the responsibility of the EGS, is done in a separate thread from the actual end-to-end entanglement generation attempts. Therefore the schedule of the next time window is assumed to be known before this time window initiates without interfering with the execution of the scheduled requests. We use batched attempts as mentioned above and make use of a Heralded Single Click scheme (see Sec. 2.4). This is a practical model to implement in software that suffices for the scope of this research. Only when the heralding pulse of a successful entanglement has arrived at both end nodes are these attempts ceased.

For this model we have chosen symmetric link lengths to investigate the influence of the total link lengths on the schedule. We only look at photon loss and will not consider other error models over

the links, such as the possible decrease of successful entanglement attempts due to asymmetrical link lengths. Thus for our scheduling only the maximal distance between the end nodes and the EGS is relevant. When implementing asymmetric links, one could account for the effects of this asymmetry by creating and sending the photon of the end node with a shorter link to the hub at a later time to in this way ensure that the photons arrive at the EGS at the same time.

4.1. Schedulers

The three schedulers used in this thesis are a Max Weight (MW) scheduler with a constant time window, a Dynamic Time Window (DTW) scheduler and a Cutoff scheduler. For all of these schedulers we have chosen a simple model without preemptive scheduling. The exact implementation of these schedulers in our code can be found in Sec. 5.1.

Max Weight Scheduler

Sec. 2.1 describes the workings of a classical MW scheduler. For implementation of this MW scheduler in quantum networks, the queue bins are assigned per session s of two end nodes requesting entanglement instead of per user as done in the classical model. The requests contain a request identifier, the number of entangled pairs required by the sessions and an identifier of the end nodes involved in a session s. These requests are then to be executed within one time window. To ensure that collectively all requests stay within the rate region (see Sec. 2.5 for the rate region) when allowing for dynamic time windows to be used, these requests can be modified to also include this rate region.

We implemented this scheduler for our research is using a constant time window size. This means that when a resource node is assigned to a session given a request, they have a predefined, finite time to generate entanglement. If entanglement was able to be generated in this time the requests is popped from the queue and if this is not the case and the end nodes were not able to generate entanglement within this time window, the scheduler does not pop the request from the queue but instead chooses a new request to be executed in the same MW fashion. This can, but is not guaranteed to be the same request.

Dynamic Time Window Scheduler

The DTW scheduler uses the same queuing system as the MW scheduler and also uses time windows but instead of using a constant time window size as our MW scheduler does it adjusts the time window size dependent on the link length of the end nodes requesting entanglement. How the time window sizes are calculated is explained in Sec. 4.2. Then adding onto this time window size we allow for an option to implement a prefix value to the time window size scaling its time up or down depending on this prefix value.

Cutoff Scheduler

The last type of scheduler we use for this research is a Cutoff scheduler. This scheduler also uses the MW queuing method, however instead of using time windows it keeps the connection of a session open until entanglement is successfully generated within the bounds of a maximum attempt time defined by a cutoff timer. Thus, a new request is only able to be executed if a resource node is freed up because the previous request being executed is finished or if the maximum attempt time specified by the cutoff timer runs out.

4.2. Service Duration

For the implementation of the Dynamic Time Window scheduler (see Sec. 5.1 for the implementation description) one should know what is the expected entanglement generation time as to optimize the size of the time windows for maximal entanglement rate and minimal idle time. The three elements that weight in on the time it takes to generate an entangled pair are the end nodes, the links and the EGS. In this section we look at the influence of all these three elements on the entanglement time separately and later combine these to get to a final time window size. However, for our experiments we have used a simplified model which omits certain parts from the simulation and therefore also from the final equation for the time window size as is explained below. App. A.1 shows an overview of the parameters mentioned in this section and their values.

End Node

The end nodes are responsible for the first steps of generating end-to-end entanglement, namely both end nodes of a session have to generate a Bell pair of which they keep one qubit stored in the state of the communication qubit and send the other one in the form of a photon over a quantum link. This initiation consists of an initiation time t_{init} and an initiation probability p_{init}. The travelling photon generation also has a time (t_{emit}) and probability (p_{emit}). This gives us an average initiation time of:

$$t_{init,avg} = \frac{t_{init} + t_{emit}}{p_{init} \cdot p_{emit}}. \tag{4.1}$$

However both these processes are probabilistic processes with small success probabilities. Moreover, given that the probability of the Bell pair generation is related to a variable value α introducing more complexity, we decided to work in a simplified model, omitting this initiation time. As for the probability: In order to isolate our investigation to the scheduler performance, independent of the quantum node performance, we omit the probability of correct initiation, p_{init} and successful photon emission p_{emit} (thus assuming $p_{init} = 1$ and $p_{emit} = 1$).

Once the end-to-end entanglement is successfully generated and the relevant end nodes have received this heralding pulse, the end nodes can do their calculations and measurements. These calculations and measurements come with their own probabilities of success and their own time frames, however this as well is outside the scope of this research given that we focus on only the generation of end-to-end entanglement.

Link

Links carry the photons from the end nodes to the EGS. For our model we have chosen to use optical fibre links as explained in Sec. 2.2. During its travel the photons are subject to attenuation losses. Fortunately these losses are known, namely the probability that a photon travelling through L kilometer of optical fibre, as is the case in our model, arrives at its destination is given by the following equation (see Ref. [2]):

$$p_{travel} = e^{-\frac{L}{L_{att}}} \tag{4.2}$$

Here L_{att} is the attenuation length[1] for single-mode optical fibre cables, where $L_{att} \approx 30.5km$ corresponds to attenuation losses of $0.2dB/km$ of these fibre cables as found in Ref. [27]. To calculate the total travel probability of the system we multiply the photon arrival probabilities p_{travel} from both end nodes A and B:

$$p_{travel,tot} = p_{travel,A} \cdot p_{travel,B} \tag{4.3}$$

The travel time (t_{travel}) of the photon from an end node to the EGS through the fibre link also depends on the elementary link lengths L of the current session. For our experiments we assume $L_A = L_B$ for the lengths of the two end nodes to the EGS of a session between A and B. Since we are using a Heralded Single Click scheme, this travel time should be doubled when calculating the travel time using using this single click scheme because we need to take into account the travel time of the heralding pulse as well. The travel time is thus defined as (see Ref. [2]):

$$t_{travel} = 2 \cdot \left(\frac{2}{3} \cdot c\right)^{-1} \tag{4.4}$$

c is the speed of light in fibre defined as $c = 3e8 \quad km/s$.

EGS

For the calculation of the time window size we do not need to take into account the calculation time of the EGS controller since we assume the scheduling and other control protocols are executed on a separate thread from the entanglement generation attempts, meaning the schedule for the upcoming time window should be calculated before the beginning of the current time window.

Before the EGS receives two photons from to-be-entangled end nodes it needs to make sure the optical switch is set correctly (t_{switch}), connecting the active session links to the assigned resource node. Then when the photons arrive at the resource it performs a Bell state measurement (t_{bsm}). Given that

[1] Where L_{att} is based on a received power equal to $\frac{1}{4}$ of the input power.

the optical switching can be done simultaneously with the Bell pair generation at the end nodes we get the preparation time (t_{prep}) by taking the maximum of these two:

$$t_{prep} = max((t_{init} + t_{emit}), t_{switch}) \tag{4.5}$$

Since the optical switching time is as can be seen in Eq. 4.5 related to the initiation time of the end nodes, we have chosen to omit this preparation time from the simulations to not unnecessarily complicate our setup.

Combining the aforementioned equations we get an equation for the average time it takes to generate one entangled pair:

$$t_{ent,avg} = \frac{1}{p_{init} \cdot p_{emit} \cdot p_{travel,tot}} \Big(t_{prep} + t_{travel} + t_{bsm} \Big) \tag{4.6}$$

This would then be a good baseline equation to use for a time window size equation. However this, as mentioned before is a complex model which contains many probabilistic elements and although the time and probability of Bell pair generation and travelling photon generation at the end nodes is relevant for making scheduling decisions and should be included in the calculation for a final time window size, for our experiments we have used a simplified model where it is assumed that the Bell pair generation and the travelling photon generation are deterministic processes with no initiation time. This simplifies Eq. 4.6 to the following:

$$t_{ent,avg} = \frac{1}{p_{travel,tot}} \Big(t_{travel} + t_{bsm} \Big) \tag{4.7}$$

The time window size w for the DTW scheduler should then be defined by this average end-to-end entanglement time ($w = t_{ent,avg}$). Thus we implemented the time window size to be as long as the time stated in Eq. 4.7. However this of course only represents the average entanglement time, which means that there is a large chance that there will be entanglement generation attempts which take longer than this average time. Therefore we have introduced a prefix value into our simulations which scales this w up to a larger value and with this possibly increasing the entanglement rate. The expected results of the introduction of this prefix value are described in Sec. 4.3.

When one wants to use varying link lengths one could modify Eq. 4.6 to use the maximal value of the two link lengths A and B. The time window size with varying link lengths would then be defined as:

$$t_{ent,avg} = \frac{1}{p_{init} \cdot p_{emit} \cdot p_{travel,tot}} \Big(t_{prep} + \max(t_{travel,A}, t_{travel,B}) + t_{bsm} \Big) \tag{4.8}$$

Or using our simplified Eq. 4.7:

$$t_{ent,avg} = \frac{1}{p_{travel,tot}} \Big(\max(t_{travel,A}, t_{travel,B}) + t_{bsm} \Big) \tag{4.9}$$

4.3. Expected Results

The end-to-end entanglement time is expected to grow exponentially with the link length. Using our MW scheduler, and therefore constant time window sizes, we expect both the number of successful entanglements and hence the entanglement rate to be stable when the expected entanglement time is below the time window size. When this average entanglement time nears the time window, the probability that entanglement is successfully established within this time window (and with that the entanglement rate) decreases and the number of successful entanglements will eventually fall to zero when this time window is smaller than at least one attempt.

The DTW scheduler is expected to have a stable value for the number of entanglement successes but given that the time window sizes increase with the link length a drop in entanglement rate proportional to the increase in time window size is expected. The DTW scheduler with a prefix value of 1 is expected to have a stable value for the percentage of entanglement successes. This is expected because for this prefix value the average entanglement time is equal to the time window size. Assuming a Gaussian distribution of the entanglement time per link length with the mean of the curve

being the average entanglement time, for all distances roughly the same percentage of entanglement attempts should prove successful. Namely, all entanglement successes that take the same amount or less time than this average entanglement time (and thus the time window size) which is slightly over half of the entanglement successes with this assumption of a Gaussian distribution. Given that the time window sizes increase with the link length a drop in entanglement rate proportional to the increase in time window size is expected. A prefix value larger than 1 is expected to have a higher, stable number of entanglement successes than when using a prefix value of 1. Therefore this larger prefix value is expected to have a higher entanglement rate.

For the Cutoff scheduler its performance (meaning its entanglement rate and therefore its throughput) will highly depend on the value set for the cutoff timer. If the timer of this scheduler is set for a relatively long time, the average entanglement time using this Cutoff scheduler is expected to be slightly smaller than the average entanglement time without a cutoff timer. The reason for this is that this timer will cut off the attempts that take too long according to the timer. If the cutoff timer is set to a smaller value, therefore further limiting the time for attempting entanglement we assume a convergence of this curve towards the value set for the time window size. Would this scheduler not have a cutoff timer there is no way to know up front how long a given request will take and due to these high outliers and is therefore expected to have a low entanglement rate. The percentage of successful entanglements is expected to depend on the choice for the cutoff timer value as for the other two schedulers. Overall however, depending on the value set for the cutoff timer, we expect this scheduler to have a higher entanglement rate than the DTW and MW schedulers given the ommitance of the idle time.

5 Implementation

In this chapter the code used for the experiments of this thesis will be explained. This code is written in Python using NetSquid and SquidASM. All code created for this project can be found at `https://github.com/Marit003/EGS`. There are two main contributions in terms of delivered code, namely schedulers and a network link protocol, as explained in Sec. 5.1 and Sec. 5.2 respectively. Furthermore the implementation of the end node is described in Sec. 5.3. App. A.1 shows the configuration parameters used.

5.1. Schedulers

SquidASM at this moment offers two types of schedulers, namely a FIFO scheduler and a static scheduler. However for various reasons these schedulers are not useful for our research. The available static scheduler is similar to a Round Robin scheduler, where every node combination is given a constant time window. This method however has as a disadvantage that even though not every session might request entanglement, all sessions are given a time slot of equal time and therefore many cycles of this scheduler will not attempt entanglement and are therefore redundant. The FIFO scheduler provided on the other hand does take into account the entanglement requests from the end node pairs when generating a schedule. However this scheduler does not offer a constant time window and instead executes a new request, possibly of a new session, as soon as the previous entanglement is established. Thus it does not have a maximal time window size as needed for quantum networking. Moreover, we are motivated to implement a scheduler with strong performance in terms of throughput–our target performance metric. As discussed in Chapter 3, Max Weight scheduling has been show to be throughput optimal for intended use on an EGS. In contrast, classical FIFO schedulers are well known to exhibit sub-optimal performance in terms of throughput, and we expect the same weakness for implementations in quantum networks. These schedulers therefore fail to meet our requirements. We therefore propose three schedulers - a Max Weight, Dynamic Time Window and Cutoff scheduler - and later describe why they meet our requirements. These three schedulers are all child classes of the blueprint classes (`IScheduleProtocol`, `IScheduleConfig` and `IScheduleBuilder`) mentioned in Chapter 3.

Max Weight Scheduler

Our Max Weight scheduler makes use of a queue when registering requests from the end nodes and when there is a free time window assigns a resource node to the session with the largest number of requests. A `QueItem` consists of a `node_id` describing the end node requesting entanglement, a request `req` containing session information and an identifier `create_id`. A session can per queue item request a single entangled qubit pair or multiple entangled qubit pairs, giving users the opportunity of batch requests. In our experiments we focus on generating single qubit pairs per request as explained in Sec. 4.3. This scheduler also allows for the use of multiple resource nodes within the EGS for different sessions.

When executing this scheduler, a request is only popped from the queue if this request is fulfilled and end-to-end entanglement is generated. If a request was not able to entangle all qubit pairs of a certain request, it is deemed an unsuccessful execution and the execution of the full request will be

tried again in a following time window where this session has the largest number of requests as per the queuing rules of a Max Weight scheduler.

A user of this MW scheduler software implementation can feed the scheduler some arguments at initiation time through `MaxWeightScheduleConfig`. With this the user can alter the values for the time window size (`time_window`), the switch time (`switch_time`) and the maximal number of resource nodes (`max_multiplexing`). Note that although we have set the switch time to zero seconds ($t_{switch} = 0$) in our experiments, the code provides the possibility to set this switch time to a different value using this configuration file.

Dynamic Time Window Scheduler

The DTW scheduler works similarly to the MW scheduler, except the time window size is not defined by the user in the configuration class (`DTWScheduleConfig`) anymore, but rather is calculated using Eq. 4.8. The reason Eq. 4.8 - the equation with the more complex entanglement time calculation - is chosen over Eq. 4.7 is to include the possibility of using this scheduler when adding asymmetric link lengths and end node initiation time and probabilities. However in our experiments we set $p_{init} \cdot p_{emit} = 1$ and $\max((t_{init} + t_{emit}), t_{switch}) = 0$ and implement the links with equal lengths to reduce Eq. 4.8 to Eq. 4.7. `DTWScheduleConfig` consists of variables for the time window prefix (`time_window_prefix`), the switch time (`switch_time`), maximal number of resource nodes (`max_multiplexing`), initiation probability as defined by $p_{init} \cdot p_{emit}$ (`prob_init`) and the additional static delay encapsulating $\max((t_{init} + t_{emit}), t_{switch})$ (`static_delay`), for implementation in SquidASM all to be specified by the user. Currently the DTW scheduler is ready for use with a single resource. However, the DTW scheduler is not yet ready for use with multiple resource nodes and therefore can not be used with `max_multiplexing` $\neq 1$.

Cutoff Scheduler

The Cutoff scheduler is not implemented in SquidASM as it is described in Sec. 4.1. Instead, for proof of principle use in our research, we have used the implementation of the MW scheduler and adjusted this to imitate a Cutoff scheduler. This is done by omitting the idle time generated when using the MW scheduler from our data to imitate the direct closing of the connection after successful entanglement generation as a Cutoff scheduler would. The time window size of the MW scheduler then functions as a cutoff timer which cuts off all connections and therefore entanglement attempts that would otherwise exceed this time.

5.2. Link Implementation

The manner in which the links and hubs are implemented in SquidASM is using the NetSquid package `netsquid_magic`. Given the large time consumption due to the low probability of entanglement of a real-world quantum network model and therefore the same large time consumption it takes to simulate such a model using quantum simulation software, `netsquid_magic` does not implement a quantum network using a real-world model but rather implements an analytical scheme calculating the resulting times and probabilities generated by this model. Fig. 5.1 shows an implementation of a network with four end nodes interconnected by links using `netsquid_magic` represented by the purple boxes on the links. It is assumed that within these links a hub is centered and although these links seem separate from each other, it is assumed that the hubs centered in these links are one and the same and the scheduler implemented takes this into account by still scheduling sessions as if there was only one hub in this system and therefore only scheduling sessions when there are free resource nodes in this hub.

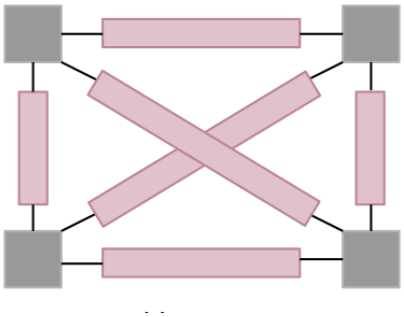


Figure 5.1: NetSquid network implementation example with four end nodes

SquidASM provides the user with several possibilities for link implementations through a `HubBuilder` construction class for our implementation of a model with an EGS. However the link implementations provided by SquidASM were not usable for implementing our EGS network model. Because of this a new type of link had to be created – the `Fibre` link – as described later in this section.

The four link types provided by SquidASM are a `Depolarise` link, a `Perfect` link, a `HeraldedSingleClickLink` and a `HeraldedDoubleClickLink`. For the purpose of our research, namely researching the entanglement rate depending on link lengths, the `Perfect` link is not usable. Given that it simulates an instant connection (meaning no travel time) it removes the possibility of studying such entanglement times. The `Depolarise` link does use travel time, however it was not suitable for our research since it did not use input variables representing initiation values to calculate output values representing resulting values of the entanglement generation attempt of the system, for instance fidelity and probability. It instead took as parameters these output values therefore needing to know these in advance rather than have these be simulated by the simulator. For this reason also the `Depolarise` link was not of use for our implementation.

The `HeraldedSingleClickLink` and `HeraldedDoubleClickLink` implementations did account for far more parameters and calculations making them better possibilities for use in our research. However, after investigation into the `HeraldedSingleClickLink` implementation it was shown that this implementation still depended on too much unfinished code. One such unfinished element is the problem of unaligned parameters where chosen values for parameters do not feed into later uses of those parameters and problems in the code occur because of hidden implementations of these parameters. Another example is the need for the user to still implement Bell state corrections when using this link type. Therefore this code was deemed not ready for use in our research. For the `HeraldedDoubleClickLink` implementation, aside from the fact that it was not the type of implementation we needed for our research given that we used a Heralded Single Click scheme, the expectation was that this software was in a similar, unfinished state as the `HeraldedSingleClickLink` implementation and therefore would also not be useful for our research.

Fibre Link

Since the link setups provided by SquidASM were not useful for our research, we have created our own `Fibre` link implementation. This link takes as input a fidelity and a link length from both end nodes A and B involved in the session (`fidelity_A` and `fidelity_B` and `length_A` and `length_B` as defined in the configuration file `FibreLinkConfig`). It then proceeds to calculate the final probability using Eq. 4.3.

Although for our experiments the focus is on generating end-to-end entanglement, meaning that the results are not dependent on the resulting fidelity, we have decided to still implement a calculation of the final fidelity in our link model. This is done for completeness of the model. This final fidelity is calculated as $F = F_A \cdot F_B$ and is then used to calculate the probability of the qubits being in the maximally mixed state p_{mms}. For this calculation of this probability we use the calculation of the `Depolarise` link, namely using:

$$p_{mms} = \frac{4(1-F)}{3} \tag{5.1}$$

Eq. 5.1 corresponds to the final fidelity when using Werner states as done in Ref. [28]. Thus when

using this link model one should therefore make sure the state preparations in the simulations adhere to the use of these Werner states.

5.3. End Node Implementation

For the end node implementation we have used SquidASM's Nitrogen-Vacancy (`NV`) end nodes. SquidASM provided two types of end nodes, namely `NV` end nodes and `generic` end nodes. The `NV` end nodes consist of more parameters than the `generic` end nodes, therefore allowing for a more elaborate model. However to correctly implement these end nodes in the whole of the simulation, end node protocols still have to be written and the `NV` functionalities called. Only an example setup of these protocols was provided by SquidASM and therefore we needed to improve these protocols for our use-case. The protocols created work for our intended use, but are not robust for addition to the SquidASM framework and should therefore solely be used for this research and should be improved upon before using them for further research.

6 Results

Using the software setup as described in Sec. 5.3 we ran several simulations of which the results are shown in this section. All of these simulations are based on 10 000 runs. This means that for every scheduler used (with various simulation parameters as can be seen in App. A.2) 10 000 requests for one entangled pair are queued for which the EGS attempts to generate successful end-to-end entanglement. This queuing is done one by one per request per session, thus filing the next request of a session when the previous one is finished.Our model was also tested with all to-be-executed requests in an input file, however in an attempt for filing requests dynamically this method was left aside. Nonetheless, in both these queuing models dynamic scheduling is used, assigning only the next request to be executed to a resource node as opposed to creating a complete schedule before the initiation of the scheduler as static schedulers do. Time windows are then assigned per request. This means that when a session files multiple requests, as done in our case, these are all assigned different time slots. If one were to specify the need for multiple entangled pairs in one request this will be attempted in the same time window. More on this option is explained in Sec. 5.1. We set the maximal number of time windows to 10 000, stopping the simulation if this maximal number is reached. Note that we do batched entanglement requests meaning that per entanglement request assigned to a resource node, the end nodes keep attempting to generate entanglement until successful or until the time window is over.

The link lengths of each individual link are chosen to be up to $50km$. The values for the time window and prefix sizes and for the cutoff timer shown in this section are chosen to show the clear effect of the different schedulers and different values and are therefore per definition not all optimal values. App. B contains figures showing these experiments for more values of the time window size, prefix value and cutoff times. As for the fidelity of these results, the measurement results of the successfully entangled qubits in the end nodes are not taken into account when counting the number of successful entanglements. Meaning even if because of low fidelity the entangled qubits collapse to different states after measurement in the end nodes, these are still counted as successful entanglements. However, one could choose to extend this model to only deem an entanglement successful if a minimum fidelity is guaranteed.

In this section we will show four main types of results. First we qualify the average time it takes to generate end-to-end entanglement between two end nodes (Sec. 6.1). We show the dynamic time window sizes as defined in Sec. 5.1 and compare them to this average entanglement time (Sec. 6.2). We also compare the average entanglement time to the average entanglement time when using the Cutoff scheduler (Sec. 6.3). Secondly, in Sec. 6.4 we look into the percentage of successful entanglements generated per type of scheduler and per time window or cutoff size. Thirdly the idle time when using these scheduler implementations is shown in Sec. 6.5. Lastly the entanglement rates reached using these schedulers is shown in Sec. 6.6. As a proof of principle we also show the effect of using two resource nodes per session on the entanglement rate of this session in Sec. 6.7.

6.1. End-to-End Entanglement Time

In this section we look into the average entanglement time (AET) to later compare the time window sizes of the used schedulers to this average entanglement time. Figure 6.1 shows the average time it takes to

generate one entangled pair between two end nodes for various link lengths using our software setup. Here we can see that the average entanglement time scales exponentially and the standard deviation (represented by the error bars) increases with the link length. This is within the expectations given the decreased probability of success (see Sec. 4.2) at larger link lengths. Given probabilistic nature of this model the average end-to-end entanglement time does not have a maximal duration. For this plot we therefore used a very large cutoff time of $1000s$.

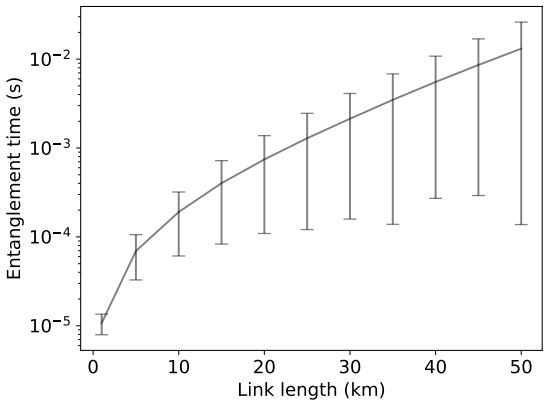


Figure 6.1: Average entanglement time for successfully generating one entangled pair per link length. Error bars show one standard deviation.

6.2. Dynamic Time Window Size

In Figure 6.2 we plot various time window sizes used for the DTW scheduler over the average end-to-end entanglement of Sec. 6.1, where TWP represents the time window prefix value. AET represents the average entanglement time as also shown in Fig. 6.1. Here we can see that when we use a prefix of value 1, the time window size coincides with the average entanglement time. This is because the time window size with $TWP = 1$ is taken directly from the average entanglement time as described in Sec. 4.2. This figure also shows that for larger prefix values the time window size increases as expected. App. B.1 shows that for prefix values smaller than ($TWP \leq 1$) entanglement was not achieved for shorter link lengths. The question now remains what the optimal prefix value is. We measure this by comparing entanglement rates which is done in Fig. 6.7b.

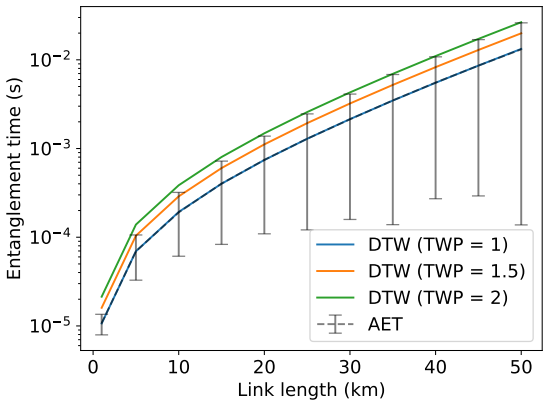


Figure 6.2: Dynamic time window sizes using prefix values (TWP) compared to the average entanglement time (AET) for successfully generating one entangled pair per link length. Error bars show one standard standard deviation.

6.3. Cutoff Scheduler Entanglement Time

The Cutoff scheduler should have a cutoff time that stops all requests that take too long to execute as to avoid the possibility of infinite execution time of a request. The definition of 'too long' and therefore the optimality of this scheduler when used in combination with an EGS should be related to the entanglement rate generated (see Fig. 6.7c) and to other design choices made to optimize the use of the EGS like a maximal cutoff time to avoid starvation. Investigating these other design choices however goes beyond the scope of our research. Fig. 6.3 shows the average entanglement times using the Cutoff scheduler with various cutoff times with respect to the average entanglement time. The error bars represent the standard deviation of the entanglement time when using this Cutoff scheduler. Here we can see that when using shorter cutoff times the average entanglement time stabilizes at larger link lengths because the entanglement times reach the maximal time per request as enforced by the cutoff timer. For cutoff time with value $TWP = 0.01s$ we see the average entanglement times when using the Cutoff scheduler coincide with the average entanglement time. Surprisingly, the impact of the cutoff timer on the average entanglement time using this scheduler compared to not using this scheduler is negligible for larger cutoff times. We expected these curves where the Cutoff scheduler is used to stay under the AET curve given that when using the Cutoff timer the outliers with very long entanglement times are cut off. In App. B.1 the average entanglement times using the Cutoff scheduler with other cutoff values are shown. In Fig. 6.3 a comparison is made between the most optimal cutoff values. Why these cutoff values are optimal is explained in Sec. 6.6. The fluctuations in Fig. 6.3 can be attributed to fluctuations in the raw data. This is in line with fluctuations shown in figures later in this Chapter.

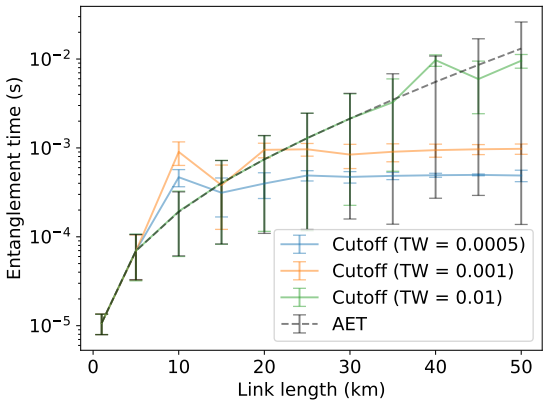


Figure 6.3: Average entanglement times for successfully generating one entangled pair using the Cutoff scheduler with various cutoff times compared to the average entanglement time (AET) for successfully generating one entangled pair per link length. Error bars show one standard standard deviation.

6.4. Entanglement Successes

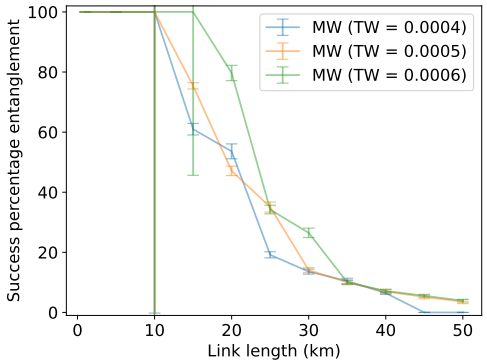
The percentage of requests for which one qubit pair was successfully entangled using the different schedulers per link length is shown in Fig. 6.4. Figures 6.4a, 6.4b and 6.4c show the success percentage for the three schedulers used with various time window sizes, prefix values and cutoff timers, respectively. Fig. 6.4d then combines these success percentage graphs for various schedulers in one figure to highlight its differences. In these figures we used a total of 10 000 runs as explained in the preface of Chapter 6. However we divided this total number of runs in 10 folds of 1000 runs to gain information about the standard deviation, which is then represented in these figures by the error bars.

Fig. 6.4a shows the percentage of entanglement successes when using the MW scheduler. App. B.2 shows this percentage for more time window sizes. From these figures it shows that for the smaller time window sizes the drop from 100% successes to 0% successes happens quite fast, however for larger time window sizes this drop becomes a bit more gradual. A success percentage of 100% means that all entanglement attempts within the time window succeed. A success percentage of 0% means that no entanglement attempts within the time window succeed and therefore most likely that the time window is too small for even one entanglement attempt to be successful. The range of success percentages in between these values can be explained by the probabilistic nature of the end-to-end en-

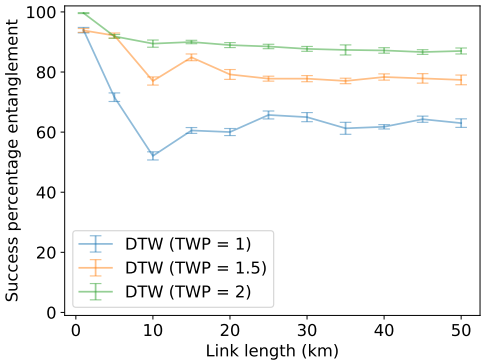
tanglement generation attempts where, when increasing the link length the probability of successful end-to-end decreases and therefore an increasingly smaller percentage of entanglement attempts result in a successful end-to-end entanglement. We can see some fluctuations in the data given the large error bars and therefore standard deviation in this figure. Since the Cutoff scheduler adheres to the same limitations in terms of maximal time allocated (for the Cutoff scheduler in terms of a cutoff timer whereas for the MW in terms of time window sizes), its success percentage curves are comparable given the same time window sizes are chosen. The success percentages for the Cutoff timer can be seen in Fig. 6.4c. Fluctuations in the success percentages between the MW scheduler and the Cutoff scheduler for the same time window size compared to the cutoff timer are because of the variations in data sets between these two schedulers and can therefore be labeled sampling fluctuations.

The DTW scheduler has a much more stable success percentage when increasing the link lengths than the MW and DTW schedulers as shown in Fig. 6.4b. The success percentage is also significantly higher than these other two schedulers but this can mostly be attributed to the choice of the time window sizes for these two schedulers. The DTW scheduler with prefix value 1, which has a time window size of the average entanglement time (see Sec. 5.1), converges towards around 60% successes which is in line with our expectations as explained in Sec. 4.3. For all prefix values the DTW scheduler starts out with a peak in entanglement successes for small link lengths and depending on the smaller or larger prefix value this percentage of successes drops faster or slower, respectively. This again can most likely be explained by the larger entanglement probability for shorter link lengths and therefore more stable values for the expected entanglement time (as shown by the smaller error bars in Fig. 6.2) for shorter link lengths compared to longer link lengths. There are some fluctuations in the data most noticeable with dips in the trend at link length $10km$ for the DTW scheduler with prefix values 1 and 2. Although we ran the simulations with 10 000 runs, these are still fluctuations in the raw data and therefore might even out with more runs and more folds of the data.

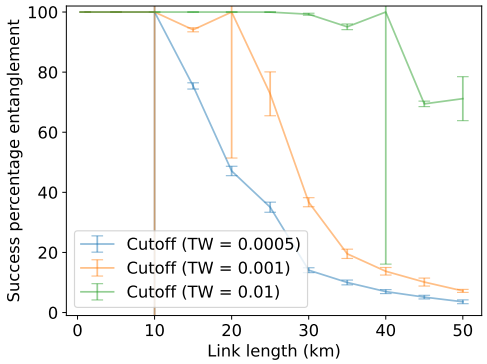
The fluctuations shown by the large standard deviation for some link lengths in Fig. 6.4a and Fig. 6.4c but also the fluctuations shown in Fig. 6.3 and in the mean in Fig. 6.7b show the volatile nature of a quantum network due to it being a probabilistic system given that one would assume 10 000 runs would stabilize the results more than shown in these figures.



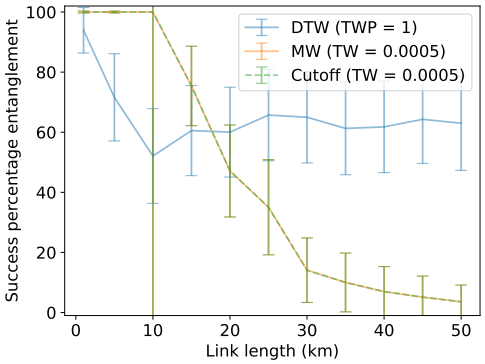
(a) Percentage of entanglement successes using the MW scheduler with various time windows (TW).



(b) Percentage of entanglement successes using the DTW scheduler with various prefix values (TWP).



(c) Percentage of entanglement successes using the Cutoff scheduler with various cutoff times (TW).



(d) Percentage of entanglement successes comparing all schedulers.

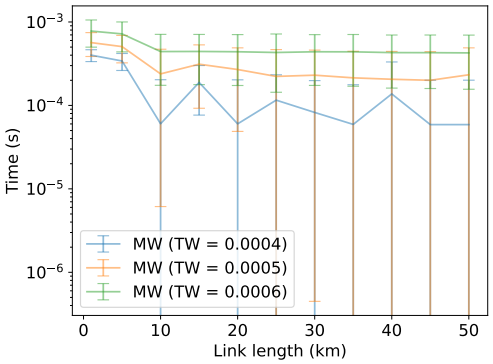
Figure 6.4: Comparison of the percentage of entanglement successes when using different schedulers, within one time window per link length. Error bars show one standard deviation.

6.5. EGS Idle Time

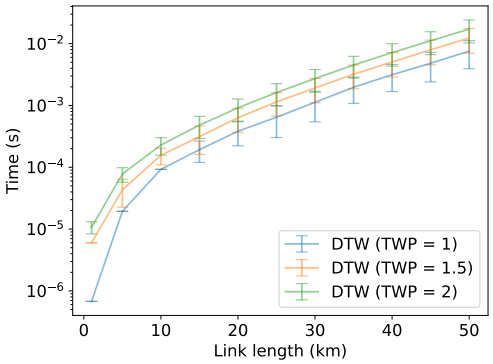
We calculated the average idle time experienced by the EGS during one time window (in which a request is executed) per link length. For the calculation of this idle time we excluded the runs where successful entanglement could not be created and thus for which there was no idle time. Fig. 6.5 shows the idle times using the MW and DTW schedulers per link length (Fig. 6.5a and 6.5b, respectively) and Fig. 6.5c combines these in one figure for a more clear comparison. The Cutoff scheduler inherently does not have idle time since it schedules new requests as soon as the previous request is fulfilled or the cutoff timer is activated.

For the MW scheduler the idle times are shown in Fig. 6.5a. Here we see relatively stable trends, however with a peak for the shortest link lengths for all time window sizes. This is in line with the expectations given that the average entanglement time and its variation are smallest at shortest link lengths. This results in the largest idle time for these shorter distances when using a constant time window for all link lengths. This also shows the inefficiency of the MW schedulers used in previous research as mentioned in Sec. 3. The fluctuations in this figure can again be explained by sampling fluctuations as can be seen by the large standard deviation represented by the error bars.

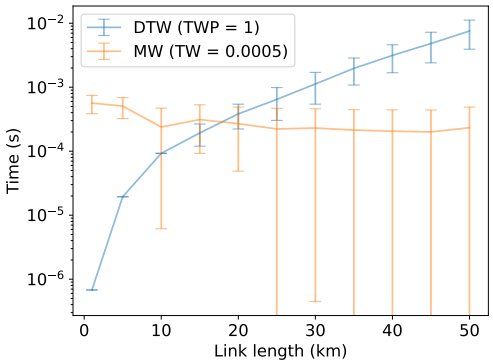
Fig. 6.5b shows the idle times for the DTW scheduler. These idle times, even for the prefix value of 1 are significant and larger than expected. In 6.5c one can see that the idle times of the DTW scheduler are better for smaller link lengths, but when increasing these link lengths the idle times quickly grow to larger values than the MW scheduler idle time values. Whether this increasing idle time of the DTW scheduler still allows for an improved performance when using this scheduler is shown in Sec. 6.6.



(a) EGS idle time during one time window when using the MW scheduler.



(b) EGS idle time during one time window when using the DTW scheduler.



(c) EGS idle time during one time window comparing the MW and DTW schedulers.

Figure 6.5: Comparison of the EGS idle time during one time window when using different schedulers, per link length. Error bars show one standard standard deviation.

6.6. Entanglement Rate

To measure the throughput of our schedulers we look at the entanglement rate (r_{ent}) which is calculated using Eq. 2.2. Fig. 6.6 shows the maximal entanglement rate per link length. This would be the rate reached when all entanglement generation attempts succeed on the first try and when the time window is equal to the entanglement time of one (successful) entanglement attempt. This figure shows an exponential decrease in entanglement rate as function of the link length.

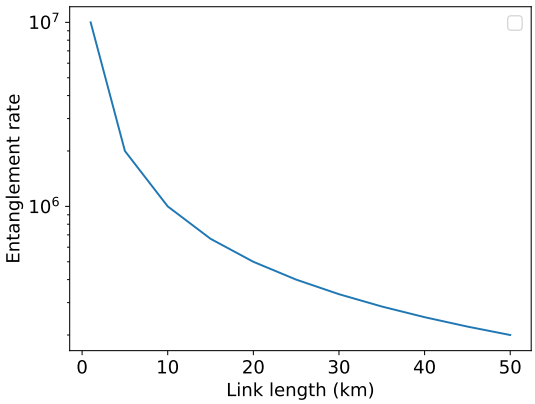
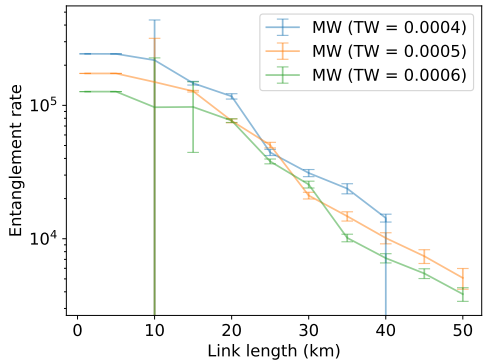


Figure 6.6: Maximal entanglement rate for successfully generating one entangled pair per link length.

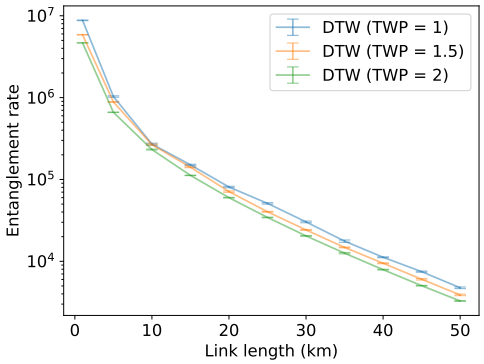
Fig. 6.7 shows the entanglement rates reached for successfully entangling one qubit within one time window when using the MW, DTW and Cutoff schedulers for some values of these time windows. App. B.4 shows the entanglement rates reached for more time window sizes. In all these figures the error bars represent the standard deviation obtained by using the same folds in the simulations as in Sec. 6.4.

We can see the impact of the high idle time of the MW scheduler on its entanglement rate for shorter link lengths in Fig. 6.7a. Especially for larger time window sizes, the entanglement rate is fairly low at these shorter entanglement distances. This can be attributed to the peak in idle time as shown in Fig. 6.5a. We can also see the entanglement rates converging towards the curve of the maximal entanglement rate as shown in Fig. 6.6, however with a steep drop in entanglement rate where the time window size caps the success percentage for shorter time window sizes (see Fig. 6.4a). The DTW scheduler shows more steady entanglement rates in line with the maximal entanglement rate of Fig. 6.6, however with a steeper decrease in rate when growing the link length. Since for all prefix values the entanglement rate curve is similar, we can address this steeper entanglement rate decrease compared to the maximal entanglement rate mostly to the idle time overhead as shown in Fig. 6.5b. Noteworthy however is that the DTW scheduler with prefix value 1 shows the best entanglement rate compared to this scheduler with larger prefix values. So even though this prefix value is responsible for a lower success percentage (Fig. 6.4b), this is made up for by its lower idle time compared to the DTW scheduler with larger prefix values. For calculating the entanglement rate of the Cutoff scheduler we have used the average entanglement time per link length as its time window size while using Eq. 2.2. This average entanglement time for the corresponding cutoff value is shown in Fig. 6.3. Here and in App. B.4 we see that for cutoff times of $5e-04s$ and higher the entanglements rates converge. This is surprising since this value is much lower than the average entanglement time without using a maximal entanglement time as can be seen in Fig. 6.3 and its success percentage is not stable for all link lengths (see Fig. 6.4c). However later in Sec. 6.6.1 we will look at the reliability of these values.

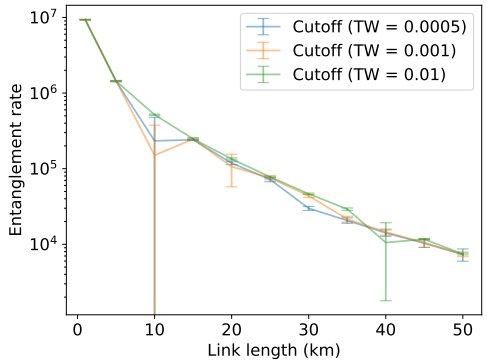
This can be attributed to the zero idle time of the Cutoff scheduler. Fig. 6.7d compares the rates of the schedulers mentioned above with optimal values for time window size, prefix value and cutoff time when looking at the entanglement rate. The optimality of these time window values is backed by the figures shown in App. B.4. Fig. 6.7d now clearly shows the non-optimality of the MW scheduler compared to the DTW and Cutoff scheduler for shorter link distances. For larger link distances the Cutoff scheduler seems optimal, however given the small error bars this conclusion can not be drawn solely from this figure. We look further into this in Sec. 6.6.1.



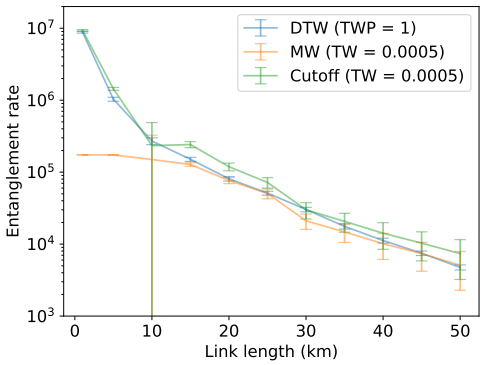
(a) Average entanglement rate for successfully generating one entangled pair using the MW scheduler with various time windows (TW).



(b) Average entanglement rate for successfully generating one entangled pair using the DTW scheduler with various prefix values (TWP).



(c) Average entanglement rate for successfully generating one entangled pair using the Cutoff scheduler with various cutoff times (TW).



(d) Average entanglement rate for successfully generating one entangled pair comparing all schedulers.

Figure 6.7: Comparison of the average entanglement rate for successfully generating one entangled pair when using different schedulers, within one time window per link length. Error bars show one standard standard deviation.

6.6.1. Reliability of the Entanglement Rate

As mentioned above, in Fig. 6.7d we can not clearly see the reliability of the schedulers given the small standard deviation and therefore the small error bars. We therefore changed our setup to 100 folds of 100 runs each as opposed to the previously used 10 folds of 1000 runs for which the results are shown in Fig. 6.9. This should give us a less biased but more reliable plot than Fig. 6.7d which uses only 10 folds.

As mentioned above, when using the Cutoff scheduler the entanglement rates using cutoff values of $5e - 4s$ and up seem to converge. In Fig. 6.8 we compare these results using 100 folds of the data instead of 10. This shows that the standard deviation decreases and therefore the reliability increases when increasing the cutoff values.

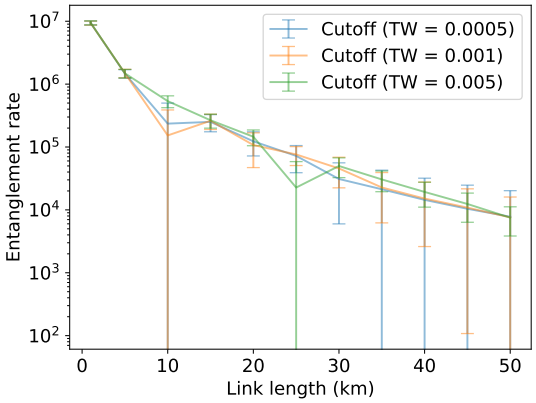


Figure 6.8: Average entanglement rate for successfully generating one entangled pair using the Cutoff scheduler within one time window per link length. Error bars show one standard standard deviation using 100 folds of the data.

Fig. 6.9 again compares the entanglement rates of the three schedulers used and shows a high reliability for the DTW scheduler given the still small standard deviation when increasing the link distances. However for the Cutoff scheduler but especially the MW scheduler, the standard deviation is relatively higher and therefore one can state that these schedulers are less reliable in terms of guaranteed entanglement rates than the DTW scheduler.

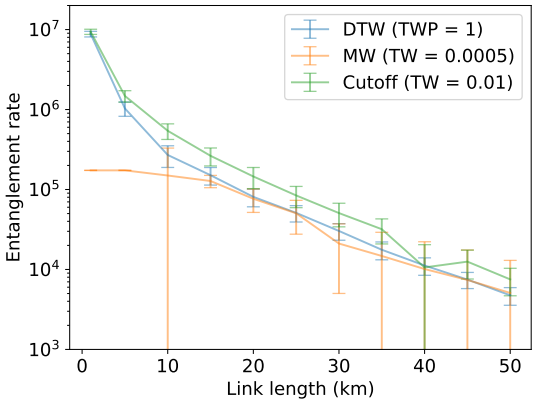


Figure 6.9: Average entanglement rate for successfully generating one entangled pair comparing all schedulers within one time window per link length. Error bars show one standard standard deviation using 100 folds of the data.

6.7. Multiple Resource Nodes Entanglement Rates

As proof of principle, in Fig. 6.10 we show the entanglement rates achieved using the MW scheduler for an EGS with two resource nodes connected to four end nodes using 100 folds of 100 runs. Given that this EGS contains two resource nodes, one time window can now execute requests from two sessions simultaneously. This results in a substantial increase in entanglement rate as shown in Fig. 6.10.

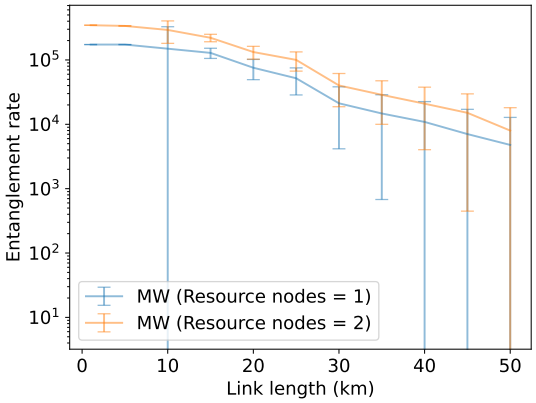


Figure 6.10: Average entanglement rate for successfully generating one entangled pair using a MW scheduler with various number of resource nodes available. Error bars show one standard standard deviation using 100 folds of the data.

7

Conclusion

In this research we found that the Cutoff scheduler shows optimal average entanglement rates and therefore optimal throughput – our main performance metric – compared to the MW and DTW scheduler across all link distances in a non-homogeneous network as is our setup. However we also found that the DTW scheduler has a higher reliability in terms of entanglement rate than the Cutoff scheduler as well as the MW scheduler for larger link lengths. Which scheduler therefore best meets the needs of a near-term real-world quantum internet depends on how important this reliability is in the implementation of this network. This research delivers SquidASM software implementations in the form of schedulers and a quantum link and gives an impression of the scaling of the entanglement rates for various schedulers with increasing link lengths which can be used for further research on quantum network switches.

We found that when increasing the link length, the maximal entanglement rate decreases exponentially. For all schedulers shown in this research the entanglement rates when using these schedules grow towards the curve of this maximal entanglement rate. However, for all schedulers the distance between the maximal entanglement rate and the achieved entanglement rates grows with the link length given the probabilistic properties of the system. For use on an EGS, which will be used in the $50km$ maximal link length range, we found that the optimal constant time window size when using a MW scheduler is $5e-04$ seconds. This time window size is the smallest time window for which the entanglement rate does not drop to zero with link lengths up to $50km$. The results show however a limited entanglement rate for shorter link length when using this MW scheduler. Thus however functional for simplified systems, a MW scheduler with constant time window size will not be optimal for a non-homogeneous network.

The DTW and Cutoff scheduler solve this problem and this is shown in the results where one can see that for these two schedulers the entanglement rates at shorter distances are significantly higher than for the MW scheduler. For the DTW we found an optimal prefix value of 1. For the Cutoff scheduler we found that when using a cutoff timer of $5e-04s$ or higher the entanglement rates converge, however that these rates are more reliable using higher values for the cutoff timer. The Cutoff scheduler has an overall higher throughput for all link lengths so one would tend to choose the Cutoff scheduler as the optimal scheduler, however as this is an active field of research ease of implementation and reliability should also be taken into account for which a DTW scheduler may prove optimal for implementation in early generations of the quantum internet. Given the limitations on the maximal time window size (for the Cutoff scheduler in the form of the cutoff timer) all schedulers should preserve the stability of the EGS.

7.1. Limitations

The research done for this thesis was limited by the unfinished implementation of the SquidASM software package for the intended use on this thesis. Because of this and with it the amount of time it took to improve SquidASM for our intended use, during our project we had to limit the research we initially intended to do to smaller, simpler research questions resulting in the research shown in this thesis. Therefore we recommend further research in this field to be within the current capabilities of SquidASM or to use separate software whenever the required software is not yet implemented in SquidASM.

7.2. Further Research

The software contributions in this work are still far from a real-life model. Several assumptions were made for this research and further steps would be to add more parameters to this simulation software like end node initiation times, initiation probabilities, and photon emission probabilities as mentioned in Sec. 5.3 but also parameters like dark counts. This research however still provides valuable insight into the performance characteristics of different scheduler implementations and can be combined with future research on control of EGS systems and general quantum network hubs. The software contributions of this thesis, specifically the possibility of requests asking for multiple entangled pairs per session can be used by research into batched entanglement as described in Ref. [9].

Previous work [12] showed that a MW scheduler combined with a control algorithm for modifying the rates requested by pairs of nodes can achieve stability of an EGS. A direction for further research could be to develop rate control protocols compatible with the DTW and Cutoff scheduler that would also result in stabilizing the EGS. One possibility for such a control protocol would be a protocol that changes the number of resource nodes assigned to a session in order to optimize the number of sessions being able to use the EGS simultaneously while ensuring all sessions meet their rate requirements. When requesting entanglement from the EGS, the end nodes will state a request rate which should fall between a minimal and maximal value as described in Sec. 2.5. The entanglement rates shown in Sec. 6.6 define the entanglement rates achievable by using one resource node in combination with the three different schedulers. Sec. 6.7 shows the increase of the entanglement rates when using two resource nodes and assumed is that for larger numbers of resource nodes the entanglement rate grows with the number of resource nodes available at a given time, restricted by Eq. 2.1. Thus when, because of the distance of an end node to the EGS, the required minimal entanglement rate of a session can not be achieved, if possible it can request more resources to be used within that time window. When resources are not available, the EGS could then decide to purge resource nodes from other sessions which would be able to achieve their minimal entanglement rate with less resource nodes than originally made available for that session. Using the entanglement rates shown, the scheduler could predict whether a session would need more resources and with this shift resource node utilization before the next time window so that the entanglement rates of all sessions within a time window fall between its minimal and maximal required values.

References

[1] Pablo Arrighi and Louis Salvail. “Blind quantum computation”. In: *International Journal of Quantum Information* 4.05 (2006), pp. 883–898.

[2] Guus Avis et al. *Asymmetric node placement in fiber-based quantum networks*. 2024. arXiv: 2305.09635 [quant-ph].

[3] Charles H. Bennett and Gilles Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: *Theoretical Computer Science* 560 (Dec. 2014), pp. 7–11. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2014.05.025. URL: http://dx.doi.org/10.1016/j.tcs.2014.05.025.

[4] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. “Universal blind quantum computation”. In: *2009 50th annual IEEE symposium on foundations of computer science*. IEEE. 2009, pp. 517–526.

[5] Angela Sara Cacciapuoti et al. “Quantum Internet: Networking Challenges in Distributed Quantum Computing”. In: *IEEE Network* 34.1 (2020), pp. 137–143. DOI: 10.1109/MNET.001.1900092.

[6] Yu-Ao Chen et al. “An integrated space-to-ground quantum communication network over 4,600 kilometres”. In: *Nature* 589 (Jan. 2021). DOI: 10.1038/s41586-020-03093-8.

[7] Tim Coopmans et al. “NetSquid, a NETwork Simulator for QUantum Information using Discrete events”. In: *Communications Physics* 4.1 (July 2021). DOI: 10.1038/s42005-021-00647-8. URL: https://doi.org/10.1038%2Fs42005-021-00647-8.

[8] Wenhan Dai, Anthony Rinaldi, and Don Towsley. *Entanglement Swapping in Quantum Switches: Protocol Design and Stability Analysis*. 2023. arXiv: 2110.04116 [quant-ph].

[9] Bethany Davies et al. *Tools for the analysis of quantum protocols requiring state generation within a time window*. 2023. arXiv: 2304.12673 [quant-ph].

[10] Artur K. Ekert. “Quantum cryptography based on Bell’s theorem”. In: *Phys. Rev. Lett.* 67 (6 Aug. 1991), pp. 661–663. DOI: 10.1103/PhysRevLett.67.661. URL: https://link.aps.org/doi/10.1103/PhysRevLett.67.661.

[11] Nuno Garcia. *The Internet Protocol – Past, some current limitations and a glimpse of a possible future*. Apr. 2021.

[12] Scarlett Gauthier, Gayane Vardoyan, and Stephanie Wehner. “A Control Architecture for Entanglement Generation Switches in Quantum Networks”. In: *IEEE Transactions on Quantum Engineering* (2023), pp. 1–17. DOI: 10.1109/TQE.2023.3320047.

[13] Roger A. Grimes. “Quantum Networking”. In: *Cryptography Apocalypse: Preparing for the Day When Quantum Computing Breaks Today’s Crypto*. 2020, pp. 189–205. DOI: 10.1002/9781119618232.ch8.

[14] Michal van Hooft. *SquidASM*. 2023. URL: https://squidasm.readthedocs.io/en/stable/#.

[15] Wojciech Kozlowski, Axel Dahlberg, and Stephanie Wehner. “Designing a quantum network protocol”. In: *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*. CoNEXT ’20. ACM, Nov. 2020. DOI: 10.1145/3386367.3431293. URL: http://dx.doi.org/10.1145/3386367.3431293.

[16] Nathalie P. de Leon et al. “Materials challenges and opportunities for quantum computing hardware”. In: *Science* 372.6539 (2021), eabb2823. DOI: 10.1126/science.abb2823. eprint: https://www.science.org/doi/pdf/10.1126/science.abb2823. URL: https://www.science.org/doi/abs/10.1126/science.abb2823.

[17] N. McKeown et al. “Achieving 100% throughput in an input-queued switch”. In: *IEEE Transactions on Communications* 47.8 (1999), pp. 1260–1267. DOI: 10.1109/26.780463.

[18] Sreraman Muralidharan et al. “Optimal architectures for long distance quantum communication”. In: *Nature* (June 2016).

[19] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[20] Nitish K. Panigrahy et al. *On the Capacity Region of a Quantum Switch with Entanglement Purification*. 2022. arXiv: 2212.01463 [quant-ph].

[21] M. Pompili et al. “Realization of a multinode quantum network of remote solid-state qubits”. In: *Science* 372.6539 (2021), pp. 259–264. DOI: 10.1126/science.abg1919. eprint: https://www.science.org/doi/pdf/10.1126/science.abg1919. URL: https://www.science.org/doi/abs/10.1126/science.abg1919.

[22] John Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: http://dx.doi.org/10.22331/q-2018-08-06-79.

[23] *Quantum Internet Alliance*. https://quantuminternetalliance.org/.

[24] Christoph Simon. “Towards a global quantum network”. In: *Nature Photonics* 11.11 (Oct. 2017), pp. 678–680. ISSN: 1749-4893. DOI: 10.1038/s41566-017-0032-0. URL: http://dx.doi.org/10.1038/s41566-017-0032-0.

[25] R. Srikant and Lei Ying. *Communication Networks: An Optimization, Control, and Stochastic Nerworks Perspective*. Cambridge, UK: Cambridge University Press, 2014. ISBN: 978-1-107-03605-5.

[26] L. Tassiulas and A. Ephremides. “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks”. In: *IEEE Transactions on Automatic Control* 37.12 (1992), pp. 1936–1948. DOI: 10.1109/9.182479.

[27] Inc. The Fiber Optic Association. *Optical Fiber Testing - Loss and Attenuation Coefficient*. 2019. URL: https://www.thefoa.org/tech/ref/testing/test/loss.html.

[28] Gayane Vardoyan and Stephanie Wehner. *Quantum Network Utility Maximization*. 2022. arXiv: 2210.08135 [quant-ph].

[29] Gayane Vardoyan et al. “On the Stochastic Analysis of a Quantum Entanglement Distribution Switch”. In: *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–16. ISSN: 2689-1808. DOI: 10.1109/tqe.2021.3058058. URL: http://dx.doi.org/10.1109/tqe.2021.3058058.

[30] Thirupathaiah Vasantam and Don Towsley. “A throughput optimal scheduling policy for a quantum switch”. In: *Quantum Computing, Communication, and Simulation II*. Ed. by Philip R. Hemmer and Alan L. Migdall. SPIE, Mar. 2022. DOI: 10.1117/12.2616950. URL: http://dx.doi.org/10.1117/12.2616950.

[31] Maudlyn Victor-ikoh and Ledisi Kabari. “Internet Architecture: Current Limitations Leading Towards Future Internet Architecture”. In: *International Journal of Computer Science and Mobile Computing* 10 (May 2021), pp. 102–112. DOI: 10.47760/ijcsmc.2021.v10i05.011.

[32] Stephanie Wehner, David Elkouss, and Ronald Hanson. “Quantum internet: A vision for the road ahead”. In: *Science* 362.6412 (2018), eaam9288. DOI: 10.1126/science.aam9288. eprint: https://www.science.org/doi/pdf/10.1126/science.aam9288. URL: https://www.science.org/doi/abs/10.1126/science.aam9288.

[33] Dilovan Zebari and Renas Asaad. “A Cyber Security Threats, Vulnerability, Challenges and Proposed Solution”. In: *Applied computing Journal* (Dec. 2022), pp. 227–244. DOI: 10.52098/acj.202260.

A

Parameters

A.1. Configuration Parameters

End node parameters	
Travelling photon generation probability (p_{emit})	1
Initiation probability (p_{init})	1
Measurement error zero state ($p_{meas,0}$)	0.06
Measurement error one state ($p_{meas,1}$)	0.005
Travelling photon generation time (t_{emit})	0s
Initiation time (t_{init})	0s
Measurement time (t_{meas})	0s
Link parameters	
Fidelity (F)	0.809
Speed of light (c)	3e8 km/s
Attenuation length (L_{att})	30.5 km
Switch time (t_{switch})	0s

Table A.1: Parameters used in the software model

A.2. Simulation Parameters

Simulation parameters	
MW time window sizes	2e-04 s, 3e-04 s, 4e-04 s, 5e-04 s, 6e-04 s, 8e-04 s, 1e-03 s, 5e-3 s, 1e-2 s
Cutoff times	2e-04 s, 3e-04 s, 5e-04 s, 8e-04 s, 1e-03 s, 5e-3 s, 1e-2 s
DTW prefix values	0.2, 0.5, 1, 1.5, 2, 4, 6
Link lengths	1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50

Table A.2: Simulation parameters used in the experiments

B

Data

B.1. End-to-End Entanglement Time

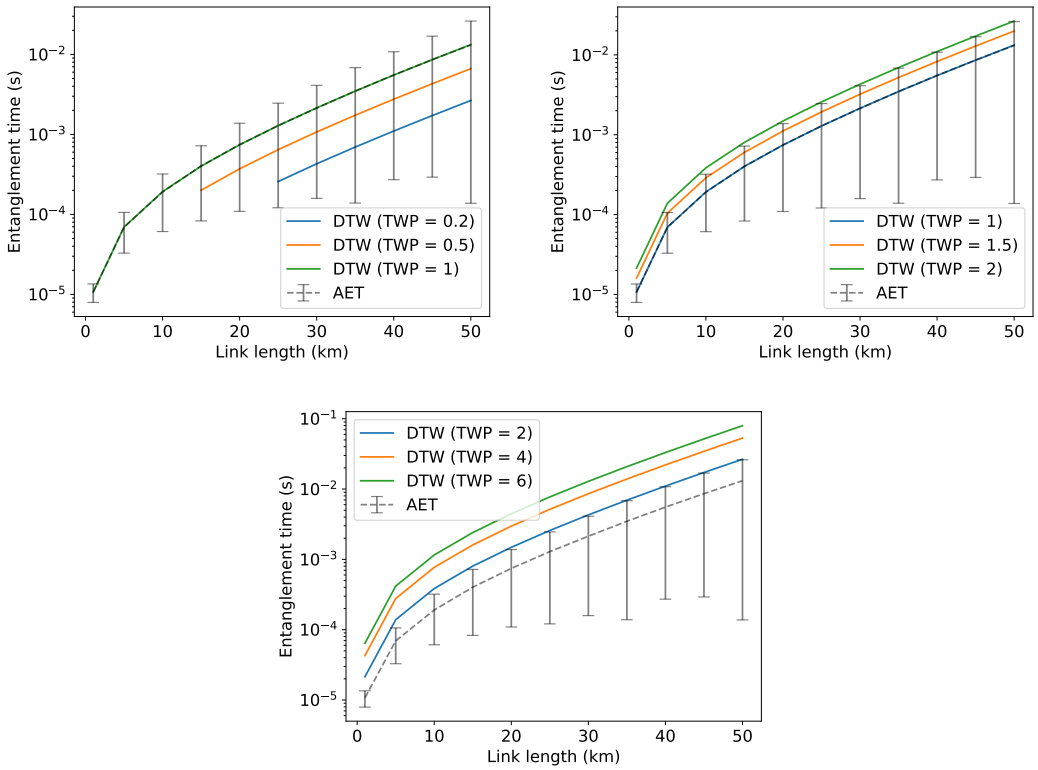


Figure B.1: Dynamic time window sizes using prefix values (TWP) compared to the average entanglement time (AET) for successfully generating one entangled pair per link length. Error bars show one standard standard deviation.

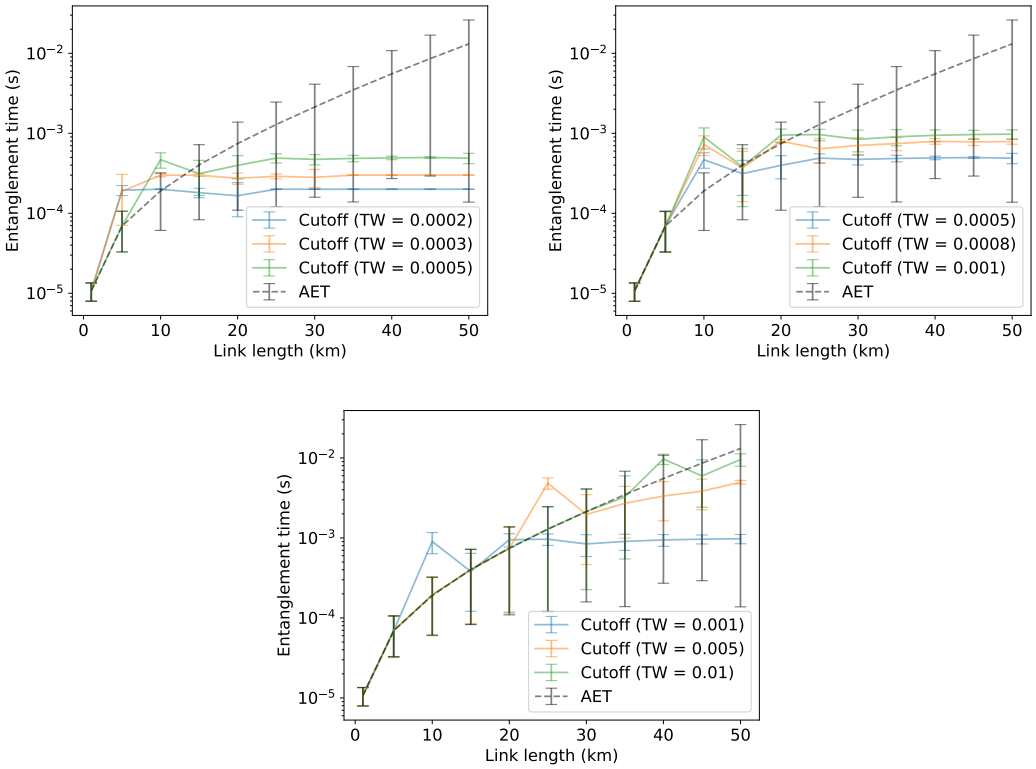


Figure B.2: Average entanglement times for successfully generating one entangled pair using the Cutoff scheduler with various cutoff times (TW) compared to the average entanglement time (AET) for successfully generating one entangled pair per link length. Error bars show one standard standard deviation.

B.2. Entanglement Successes

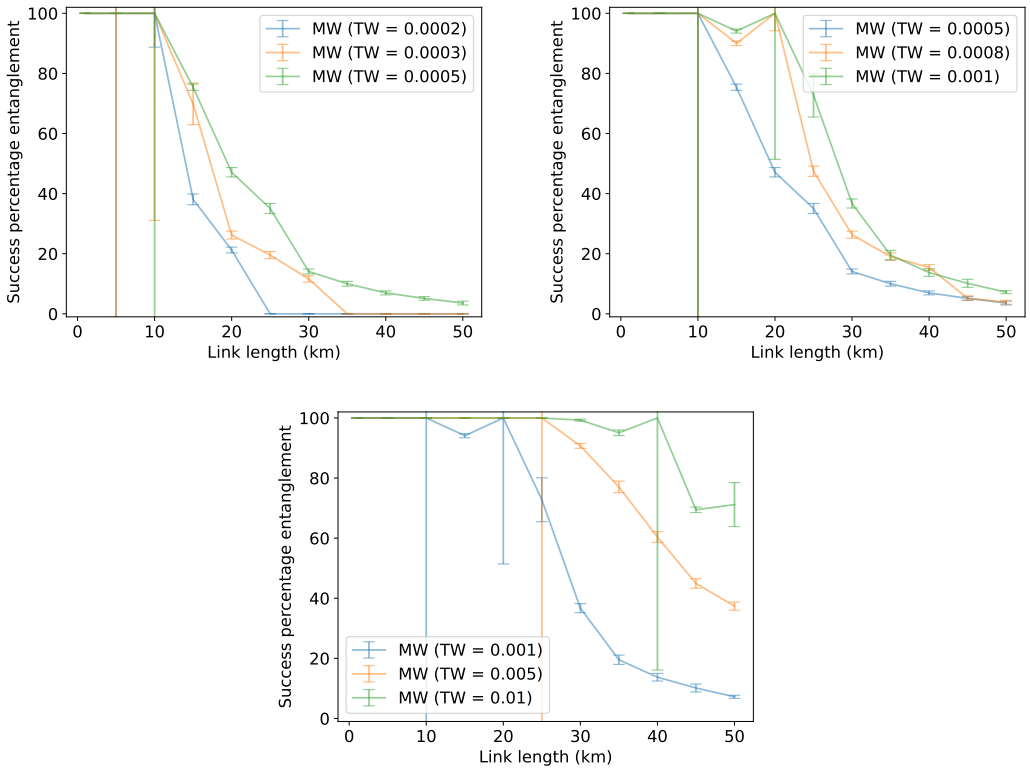


Figure B.3: Percentage of entanglement successes within one time window (TW) using the MW scheduler per link length. Error bars show one standard standard deviation.

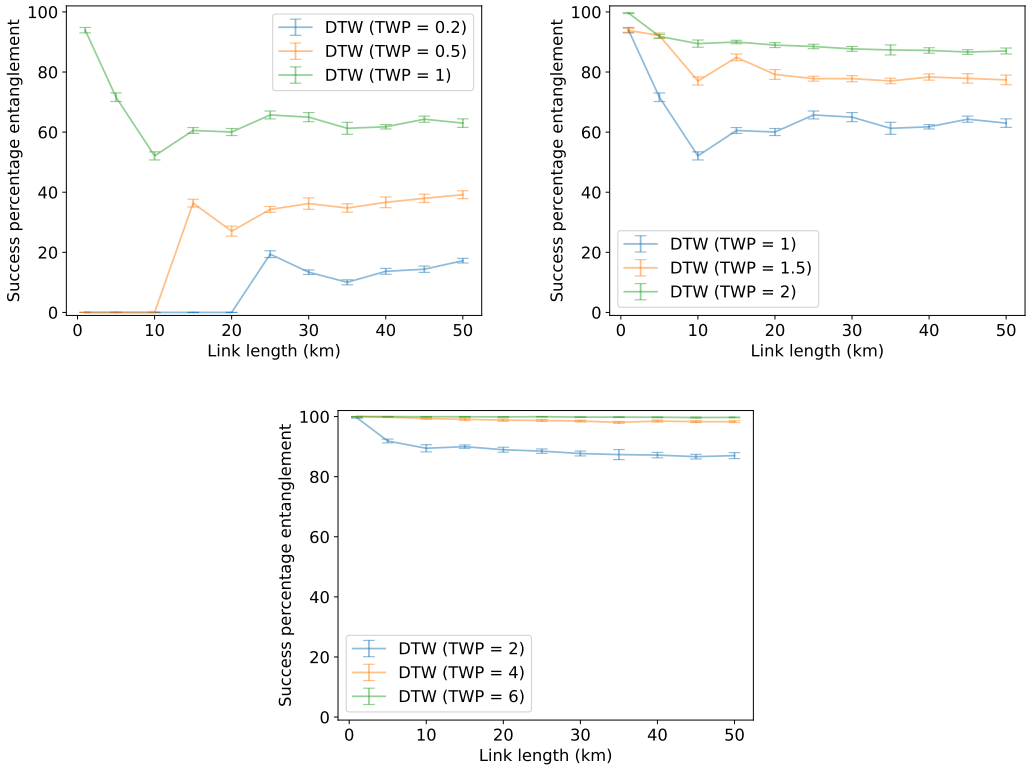


Figure B.4: Percentage of entanglement successes within one time window using the DTW scheduler with prefix values (TWP) per link length. Error bars show one standard standard deviation.

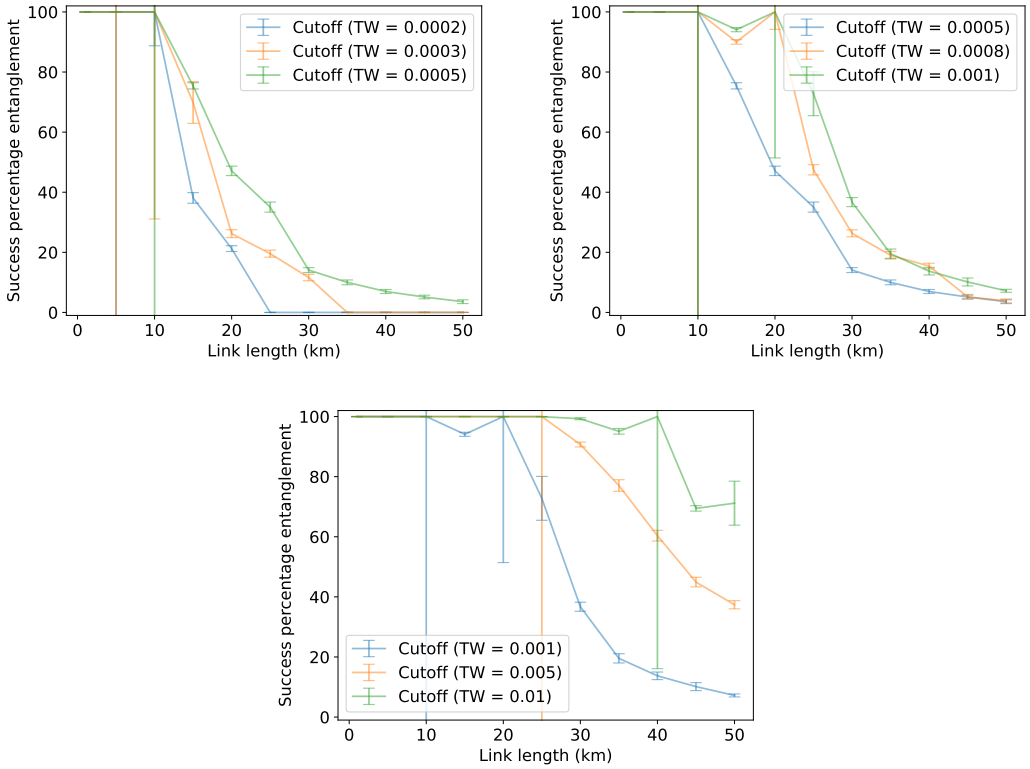


Figure B.5: Percentage of entanglement successes within one time window (TW) using the Cutoff scheduler per link length. Error bars show one standard standard deviation.

B.3. EGS Idle Time

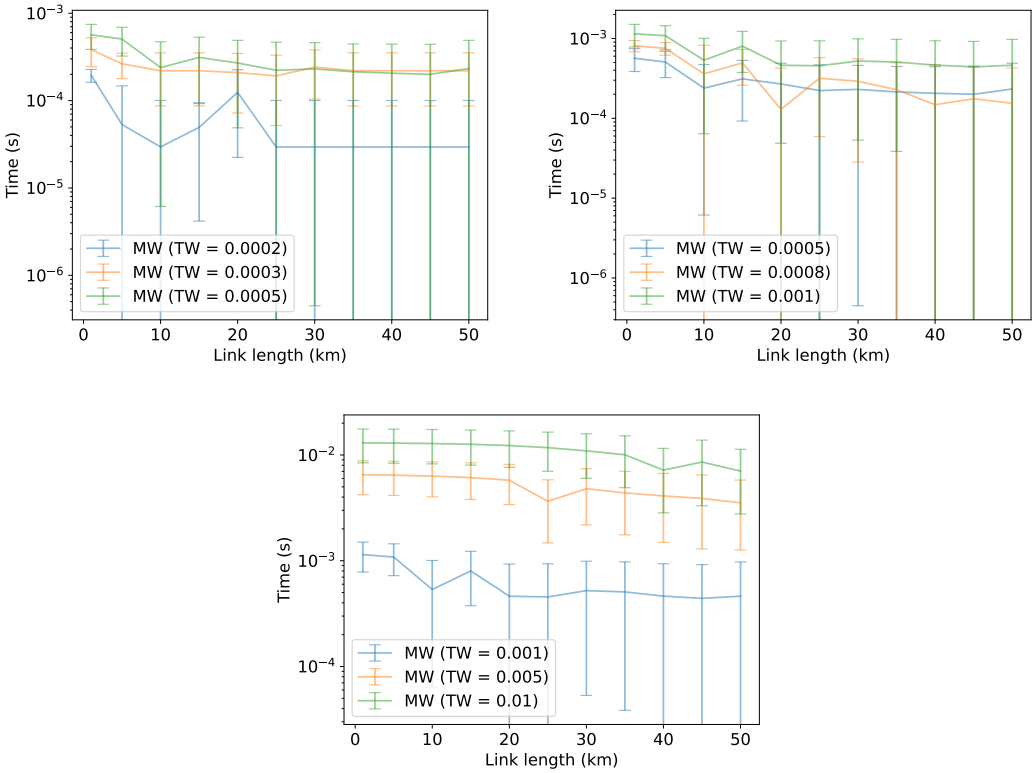


Figure B.6: EGS idle time during one time window (TW) using the MW scheduler per link length. Error bars show one standard standard deviation.

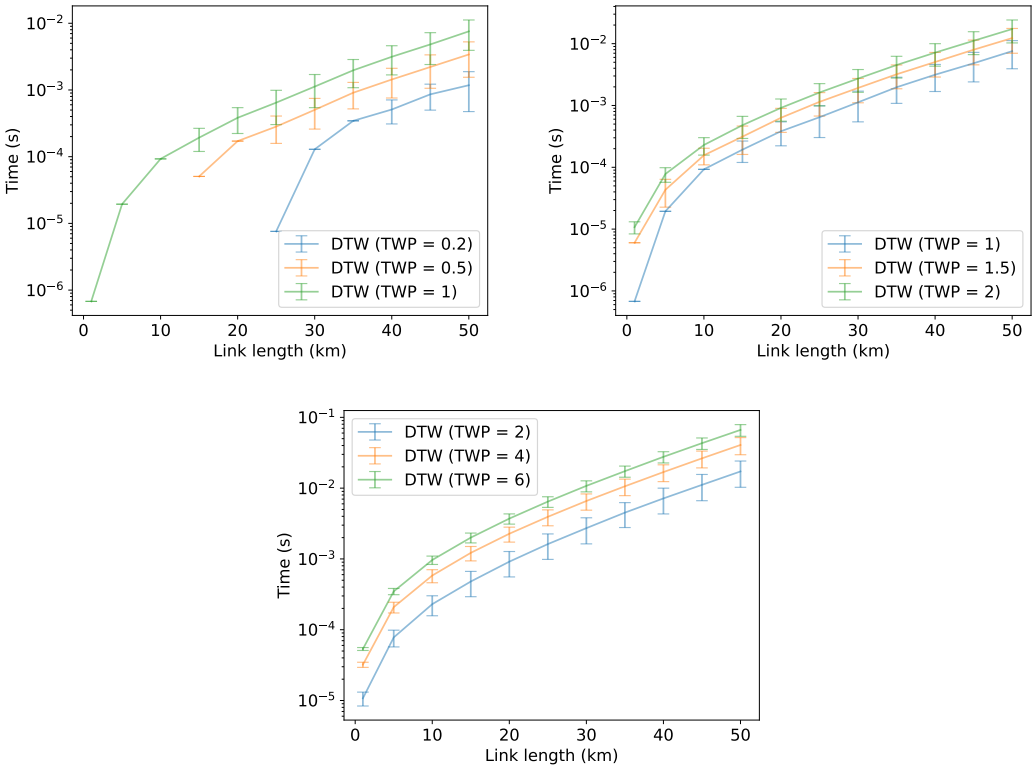


Figure B.7: EGS idle time during one time window using the DTW scheduler with prefix values (TWP) per link length. Error bars show one standard standard deviation.

B.4. Entanglement Rate

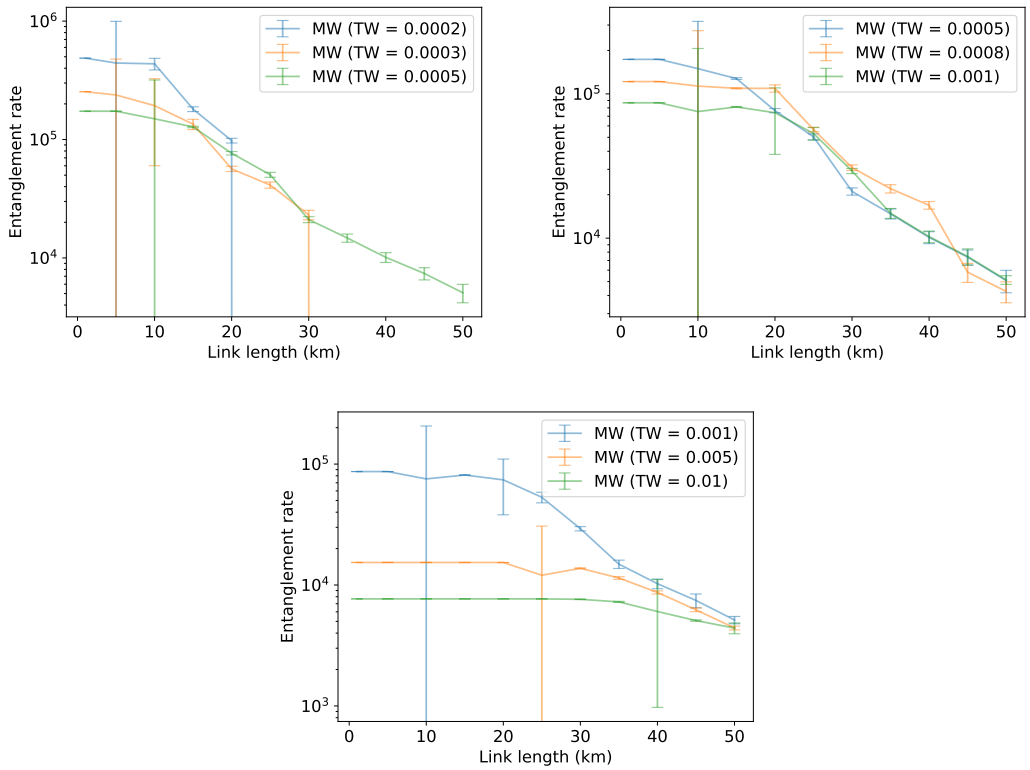


Figure B.8: Average entanglement rate for successfully generating one entangled pair using the MW scheduler with time windows TW per link length. Error bars show one standard standard deviation.

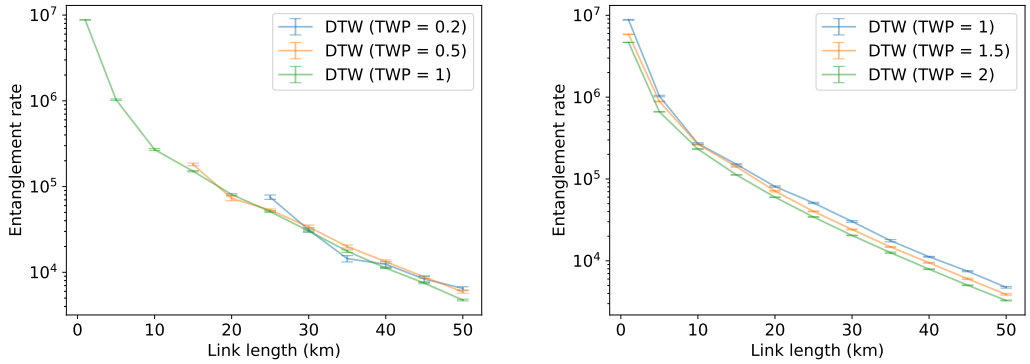


Figure B.9: Average entanglement rate for successfully generating one entangled pair using the DTW scheduler with prefix values (TWP) per link length. Error bars show one standard standard deviation.

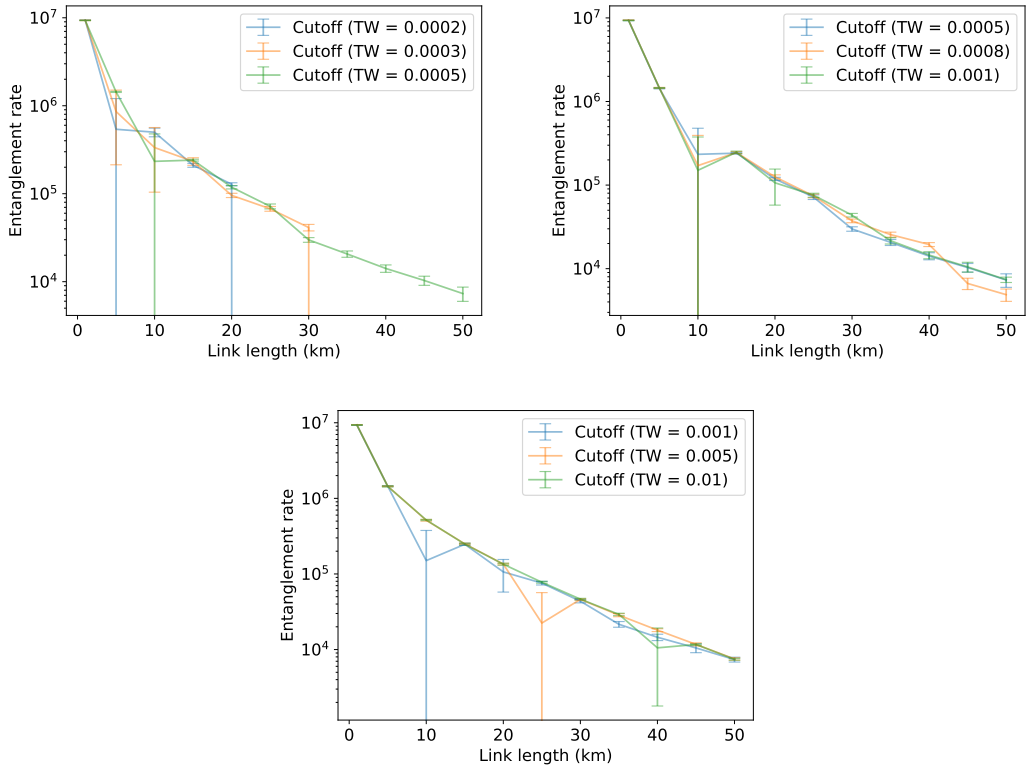


Figure B.10: Average entanglement rate for successfully generating one entangled pair using the Cutoff scheduler with cutoff times TW per link length. Error bars show one standard standard deviation.