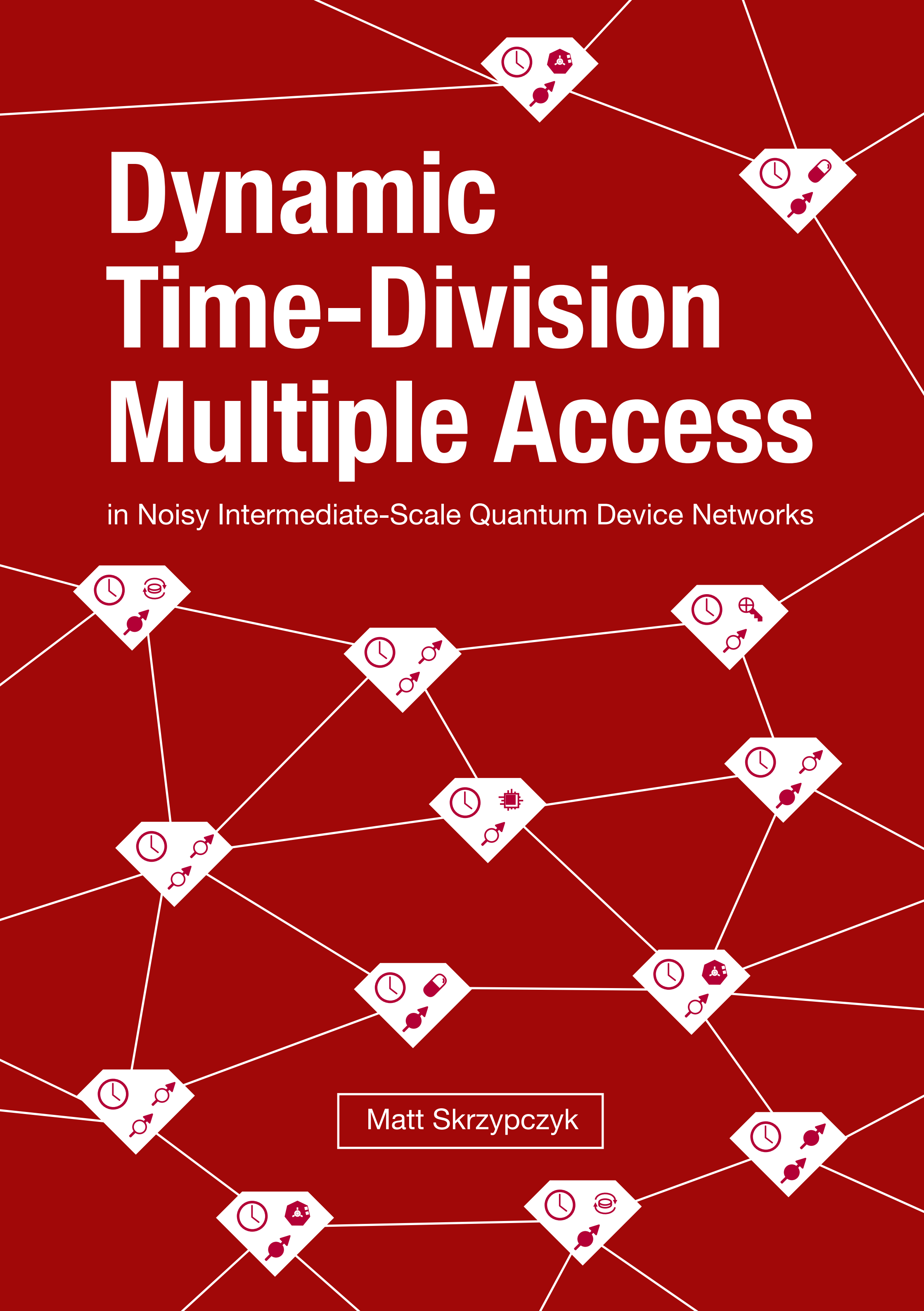
Dynamic Time-Division Multiple Access

in Noisy Intermediate-Scale Quantum Device Networks



Matt Skrzypczyk

Dynamic Time-Division Multiple Access in Noisy Intermediate-Scale Quantum Device Networks

by

Matthew Skrzypczyk

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly at 2:00pm CET August 28th 2020
https://meet.jit.si/defense_mskrzypczyk

Student number:	4777425	
Project Duration:	August 27th-May 15th	
Thesis committee:	Prof. dr. S. D. C. Wehner,	TU Delft, supervisor
	Prof. dr. R. Hanson,	TU Delft
	Dr. P. Pawełczak,	TU Delft
Keywords:	Quantum Networks, Time-Division Multiple Access, Scheduling	
Front & Back:	Beautiful cover art that captures the entire content of this thesis in a single illustration, Matthew Andersen.	

TU Delft Delft University of Technology

delft leiden Casimir research school

List of Publications

4. A. Dahlberg, B. van der Vecht, C. Delle Donne, **M. Skrzypczyk**, W. Kozlowski, S. Wehner. *NetQASM - A low-level instruction set architecture for hybrid quantum-classical programs in a quantum internet.* Manuscript in preparation.

3. T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. de Oliveira Filho, M. Papendrecht, J. Rabbie, F. Rozpędek, **M. Skrzypczyk**, L. Wubben, W. de Jong, D. Podarenau, A. Torres Knoop, D. Elkousse, S. Wehner. *Netsquid, a discrete event simulation platform for quantum networks.* Manuscript in preparation.

2. A. Dahlberg, **M. Skrzypczyk**, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, and R. Hanson. *A link layer protocol for quantum networks.* In Proceedings of the ACM Special Interest Group on Data Communication, pp. 159-173. 2019.

1. T. Coopmans, A. Dahlberg, **M. Skrzypczyk**, F. Rozpędek, R. Ter Hoeven, L. Wubben, R. Knegjens, J. de Oliveira Filho, D. Elkouss, and S. Wehner. *Simulation of a 1025-node quantum repeater chain of NV centres with NetSquid, a new discrete-event quantum-network simulator.* APS 2019 (2019): L28-012.

I found myself looking around at one point, a really bleak winter in New York... it was just so depressing, and I think I needed color.

Jim Carrey

Acknowledgements

The research presented in this Masters thesis is the result of a year's worth of effort and discussion with many invaluable individuals and colleagues at the Technical University of Delft along with the loving support of my friends and family. I would like to express my gratitude for these people from the bottom of my heart as without their assistance this work would not have been possible.

I would like to first give my deepest thanks to my supervisor Prof. Stephanie Wehner for providing me an opportunity to work alongside her and the brilliant individuals in her group. You have shown me great kindness and I have always felt that my voice was heard in our discussions. It is hard to believe that participating in your lectures on Quantum Cryptography from the other side of the Earth could have lead to our collaborative efforts. My gratitude for the experiences and the opportunities that have opened up to me upon moving to the Netherlands is beyond words.

The opportunity to work with Stephanie and her group was not limited to the scope of this thesis but also to previous work with her along with her PhD students Axel Dahlberg, Tim Coopmans, Filip Rozpe⊠dek, and her previous Masters student Leon Wubben. Working together was truly a joy and has deepened my appreciation of academic research. To this end I would also like to thank Prof. Ronald Hanson, Matteo Pompili, and Arian Stolk, Przemysław Pawełczak, Robert Knegjens and Julio de Oliveira Filho for their assistance. Our research on designing a link layer protocol for quantum networks would not have been possible without any of you and I am grateful for having the opportunity to work alongside all of you on this project. Had it not been for this opportunity I may not have found my passion for researching quantum networks.

I would like to give additional thanks to Victoria Lipinska, Kenneth Goodenough, Kaushik Chakraborty, Carlo Delle Donne, and Wojciech Kozłowski for countless discussions throughout this work. These were extremely valuable for working through roadblocks that arose and forming a clear picture of the work in this thesis.

I am thankful to Kenneth and Kaushik for providing my cat, Pip, and I with a place to live in times of uncertainty. When I arrived in the Netherlands to begin my Masters degree I had no housing arranged and Kenneth showed no hesitance in taking Pip and myself into his studio apartment. Later, I found myself between apartments and Kenneth was gracious enough to take us in again before I found a stable living situation with Kaushik. I am forever in debt to these individuals for their limitless kindness, I have no idea what would have happened to us if it were not for the two of you.

In addition to academic acknowledgements I must additionally acknowledge those that have supported me physically. I would like to thank Axel Dahlberg for the many times we have gone bouldering together as well as Kenneth and Ben Criger for supporting me at the gym. I would also like to thank David Meier for the many times he has delivered strong blows to my knees and nose during our Thai boxing sparring sessions. These experiences have had a tremendous impact on my well-being and self-confidence.

While there are many individuals in The Netherlands that I have acknowledged, there are several individuals from my home in the United States that I must pay thanks to. I would like to thank my friends Dan Habel, Trevor Wiegard, Daniel Price, Sam Gottlieb, Matt Shukin, Logan Field, and Julian Ruse for the many days we spent together. I treasure our memories and I am happy that we have remained friends through the years. I must also thank my friends Surya Bakshi, Philip Daian, Will Schellhorn, Kai Demler, Solomon Kell, and Bhargava Manja for supporting me throughout my Bachelors degree in Champaign, Illinois.

I would like to give special thanks to Bryan Quinto for being a dear friend to me for over twenty years. I always find myself amazed that we have remained friends for so long in the face of distance and I will always remember the time that we spent growing up alongside one another on Roslyn Road. Your family is an extension of my own and I am forever grateful to them for raising us to be the people that we are today.

Finally, I would like to give the greatest thanks to my loving family. I thank my mother and father, Joanna and Pawel, for raising me to be who I am and providing loving support for my entire life. I cannot thank you enough for all the work you have done in order to provide for me. I would like to thank my sister Alex for always being supportive of my decisions as well. Our youth was difficult at times but I am so happy that we have grown into the individuals we are and have learned to support one another in whatever we do. I would like to thank my grandmother Maria for taking care of my sister and I, you have shown us an immense

amount of love.

There is so much more I could say about the individuals mentioned, but this would result in doubling the length of this thesis. Once again, many thanks to everyone I have mentioned here for helping me produce this work.

Abstract

Quantum networks are networks composed of quantum processors that facilitate the exchange of information in the form of quantum bits, also called qubits. Qubits observe a physical phenomenon known as entanglement that enables the transmission of quantum information over long distances as well as the realization of novel protocols and applications that are impossible in classical networks such as the Internet. Due to the limitations of state of the art Noisy Intermediate-Scale Quantum (NISQ) devices, the establishment of entanglement over multi-hop networks demands strict coordination among the network nodes that connect two hosts. The delivery of entanglement in multi-hop quantum networks is further complicated when the network must support Quality of Service requirements for multiple users at the same time. The main challenges are ensuring that connected quantum processors agree when to establish entanglement with their neighbors and that processors use the correct pieces of entanglement to connect source/destination pairs. In this thesis, we propose a novel dynamic time-division multiple access (TDMA) method for multiplexing network resources used to connect multiple users in quantum networks. We investigate the behavior of scheduling heuristics in constructing the TDMA schedules and the effects of resource allocation on network performance. We additionally propose a novel scheduling problem and heuristic based on limited preemption that improves achievable network throughput in the case that devices may tolerate interruptions during connection of users.

Contents

1

Introduction

Quantum networks are communication networks that facilitate the end-to-end delivery of information in the form of quantum bits or qubits [1, 2]. These networks are comprised of quantum processing end nodes that run applications using qubits as well as quantum repeater nodes that enable the generation of entanglement between nodes separated over great distances [3]. In quantum communication, such networks enable the transmission and reception of qubits that are used for fulfilling higher level applications such as those in cryptography [4, 5], metrology [6, 7], and distributed computing [8, 9]. In the realm of quantum computing, such quantum networks allow the realization of distributed quantum computing clusters that have more computing power than the individual processors.

The primary mechanism that enables these capabilities is the generation of entangled qubits, or entangled *links*, between distant nodes in the network. An entangled link may then be used to transmit information between nodes running applications. In a quantum network, connected nodes can generate entanglement between one another while unconnected nodes must establish entanglement over a path of nodes between them. Each pair of nodes along such a path must establish an entangled link with one another and then perform a process known as *entanglement swapping* that transforms two adjacent links into one connecting the ends [10, 11]. One particularly interesting property of this process is that link establishment and entanglement swapping may be performed in any order.

This provides flexibility in the way resources are managed and used to connect users in the network. In classical networks, such as the Internet, data must be propagated along a path from one computer to another in a cascading fashion.

Current state-of-the-art quantum processors are deemed as noisy-intermediate-scale quantum (NISQ) devices [12]. These devices are characterized by their limited qubit counts and the noise introduced to qubits due to imperfect control and storage. As such, generating entanglement in a cascading fashion would undoubtedly introduce a large amount of noise in the qubits stored at the source while the rest of the path is built. One thus desires that the end-to-end entanglement is generated with low latency to reduce these effects. This can be achieved by generating entangled links along the path simultaneously and performing entanglement swapping whenever a node has established two adjacent links.

An important aspect in designing quantum networks is ensuring entanglement establishment can be performed to meet the Quality of Service demands of applications. Similarly to classical networks, applications in quantum networks may have differing service requirements in order to operate properly. Quantum key distribution (QKD) [5, 13] is an example of an application that only requires the availability of a single entangled link at any point in time in order to maintain correctness while blind quantum computing [8, 9] may require several entangled links at the same time. The means of coordinating entanglement establishment in a quantum network must thus be flexible in order to allow the support of many different applications. Furthermore, this coordination must be capable of supporting different applications between multiple users concurrently.

In this thesis we explore the use of channel access methods to coordinate multi-hop entanglement generation in networks of quantum processors. Our contributions can be summarized as follows:

- Propose an architecture for utilizing dynamic time-division multiple access (TDMA) channel access methods in networks of quantum processors (Chapter 3). To our knowledge, this is the first proposal of TDMA techniques for coordinating entanglement delivery in multi-user quantum networks.

- Formulate the problem of constructing TDMA schedules to satisfy QoS demands between multiple users in quantum networks (Chapter 3). We are not aware of any formulations for this aspect of TDMA scheduling in quantum networks.

- Propose a new periodic task scheduling problem known as limited preemption budget task scheduling as well as a heuristic scheduling technique (Chapter 4).
- Investigate the use of techniques the from periodic task scheduling and resource-constrained project scheduling (RCPSP) problems for constructing TDMA schedules (Chapter 4). For example, we show that RCPSP schedule construction techniques achieve higher network throughput and resource utilization over periodic task scheduling techniques at the cost of greater variation in worst-case response time and jitter.
- Evaluate the performance of scheduling techniques under different network parameters (Chapter 5). We compare scheduling techniques based on network throughput, worst-case response time, and jitter experienced in entanglement delivery.
- Investigate the effects of resource allocation to congested repeater nodes (Chapter 5). We show that quantum networks can achieve higher throughput by increasing the number of qubits in repeater nodes without increasing the number of qubits in processing end nodes.
- Investigate the effects of resource allocation to protocols in congested networks (Chapter 5). Allocating a subset of network resources to connect peers rather than using all resources along a path between two nodes allows higher network throughput.

The remainder of this thesis will be structured as follows. In chapter 2 we will provide background on quantum information theory, devices, and networks to arm the reader with an understanding of the setting of quantum networks. We also provide information on medium access control and scheduling to understand the methodology used to evaluate our proposed architecture.

Chapter 3 will present the motivation for our approach and a description of the models used. We will discuss the architecture and infrastructure of various network components that enable the use of dynamic TDMA channel access methods in quantum networks. Here, we will also pose the primary problem statement of constructing TDMA schedules for connecting nodes in quantum networks.

Chapter 4 will provide an in-depth discussion of the methodology used in reformulating the TDMA schedule construction as a periodic task scheduling problem and as a resource-constrained project scheduling problem. Here, we will describe how we construct network traffic for evaluating our scheduling methods and additionally introduce a new task scheduling problem as well as our novel scheduling heuristic that may be useful for future quantum networks.

A comparison and analysis of different scheduling techniques will be showcased in chapter 5. We will provide an analysis of the effects of different network parameters on the achievable latency in connecting nodes in a quantum network. We then evaluate and compare the investigated scheduling techniques and showcase how resource allocation for repeater protocols plays a role in the achievable throughput of a network. Chapter 6 concludes the thesis with a discussion of the results as well as future work that may follow from our contributions.

References

[1] S. Wehner, D. Elkouss, and R. Hanson, *Quantum internet: A vision for the road ahead,* Science **362**, eaam9288 (2018).

[2] H. J. Kimble, *The quantum internet,* Nature **453**, 1023 (2008).

[3] W. J. Munro, K. Azuma, K. Tamaki, and K. Nemoto, *Inside quantum repeaters,* IEEE Journal of Selected Topics in Quantum Electronics **21**, 78 (2015).

[4] C. H. Bennett and G. Brassard, *Quantum cryptography: Public key distribution and coin tossing,* arXiv preprint arXiv:2003.06557 (2020).

[5] A. K. Ekert, *Quantum cryptography based on bell's theorem,* Physical review letters **67**, 661 (1991).

[6] D. Gottesman, T. Jennewein, and S. Croke, *Longer-baseline telescopes using quantum repeaters,* Physical review letters **109**, 070503 (2012).

[7] P. Komar, E. M. Kessler, M. Bishof, L. Jiang, A. S. Sørensen, J. Ye, and M. D. Lukin, *A quantum network of clocks,* Nature Physics **10**, 582 (2014).

[8] A. Broadbent, J. Fitzsimons, and E. Kashefi, *Universal blind quantum computation,* in *2009 50th Annual IEEE Symposium on Foundations of Computer Science* (IEEE, 2009) pp. 517–526.

[9] J. F. Fitzsimons and E. Kashefi, *Unconditionally verifiable blind quantum computation,* Physical Review A **96**, 012303 (2017).

[10] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, *Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels,* Physical review letters **70**, 1895 (1993).

[11] M. Zukowski, A. Zeilinger, M. A. Horne, and A. K. Ekert, *" event-ready-detectors" bell experiment via entanglement swapping.* Physical Review Letters **71** (1993).

[12] J. Preskill, *Quantum computing in the nisq era and beyond,* Quantum **2**, 79 (2018).

[13] C. H. Bennett and G. Brassard, *Proceedings of the ieee international conference on computers, systems and signal processing,* (1984).

2 Background

This chapter will provide background on several topics that the reader may find helpful in understanding our contributed work. It will touch on key details of networks of quantum processors from a computer science perspective as well as a summary of medium access control (MAC) in classical networks. Here, we will distinguish a quantum network from a typical computer network by referring to the latter as a *classical* network. We then provide an overview of the periodic task scheduling and resource-constrained project scheduling problems before concluding with a summary of work related to the contents of this thesis.

2.1. Quantum Networks

Quantum networks are collections of quantum processors that are connected to one another to allow the transmission of quantum data in the form of quantum bits (qubits). Such networks enhance communication technology and enable protocols and applications that are more efficient than their classical counterparts or cannot be performed classically. In this section, we will give a brief overview of quantum computing and networking concepts that are relevant to the work in this thesis in addition to a summary of quantum network structures and elements. The goal is to present the reader with an idea of how the notions of connection establishment and data transmission operate in comparison to classical networks. We refer the curious reader to [1] for additional information.

2.1.1. Qubits and Entanglement

The first distinguishing property between quantum networks and classical networks begins at the level of data representation. Data in classical networks (*classical data*) is encoded using a sequence of bits that take on values of either 0 or 1. Data in a quantum network is encoded with *quantum* bits or *qubits*. In contrast, qubits do not need to take on discrete values and the information in a qubit can be a linear combination of these values.

Definition 2.1.1. A qubit $|\psi\rangle$ is represented as

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$

where $|\psi\rangle$ refers to the encoded data known as a *quantum state*. Specifically, $|0\rangle$ and $|1\rangle$ are referred to as *basis states* and the symbol '$|\cdot\rangle$' is commonly referred to as a *bra*. Here, $|0\rangle$ and $|1\rangle$ represent the following basis states:

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

known as *computational* basis states and α_0 and α_1 are complex numbers called *amplitudes* with the requirement that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. In addition to $|\cdot\rangle$, we define $\langle\cdot|$ to be the conjugate transpose $|\cdot\rangle^\dagger$ of the vector $|\cdot\rangle$ and refer to it as a *ket*. This is commonly referred to as *bra-ket* notation [1]

Using this definition we may encode classical 0's and 1's in a qubit $|\psi\rangle$ by choosing α_1 or α_0 to be zero respectively.

To extract information that is stored within a qubit, one may perform a *measurement* of the system.

Definition 2.1.2. A *projective measurement* in basis $\{|0\rangle, |1\rangle\}$ is defined as a set of projectors $\{\Pi_0 = |0\rangle\langle 0|, \Pi_1 = |1\rangle\langle 1|\}$, satisfying $\Pi_i^2 = \Pi_i$ and $\sum_i \Pi_i = \mathbb{1}$. When a qubit is measured, the quantum state is projected onto one of the basis states as indicated by the resulting *measurement outcome*.

Example 2.1.1. As an example, suppose we are measuring in the *standard basis* denoted by the basis states above. There are two possible outcomes, namely measuring $|0\rangle$ or $|1\rangle$, of which each outcome is commonly labelled by $+1$ and -1. If $|\psi\rangle = |0\rangle$ the result of measuring the system will produce an outcome of $+1$ corresponding to the $|0\rangle$ basis vector. Similarly, measuring the state $|\psi\rangle = |1\rangle$ will return a measurement outcome of -1.

With respect to the standard basis, when both α_0 and α_1 are non-zero, the state $|\psi\rangle$ is said to be in a *superposition*.

Measuring the state $|\psi\rangle$ in the standard basis yields a measurement outcome of $+1$ with probability $|\alpha_0|^2$ and a measurement outcome of -1 with probability $|\alpha_1|^2$ (hence our previous requirement that $|\alpha_0|^2 + |\alpha_1|^2 = 1$). Upon performing this measurement, the qubit state is said to *collapse* into the basis state corresponding to the measurement outcome, effectively destroying the previous state $|\psi\rangle$.

When considering multiple qubits, an n-qubit state encodes a linear combination of all 2^n possible n-bit binary strings and may be represented as $\sum_1^{2^n} \alpha_k |k\rangle$ with the requirement that $\sum_{k=1}^{n} |\alpha_k|^2 = 1$.

Example 2.1.2. Suppose we have two single-qubit states $|\psi\rangle_A = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|\psi\rangle_B = |0\rangle$. The two-qubit state $|\psi\rangle_{AB}$ may be written as $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)_A|0\rangle_B = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)_{AB}$. Here, $|\psi\rangle_{AB}$ may be expressed as a tensor product of the single-qubit states $|\psi\rangle_A$ and $|\psi\rangle_B$.

Definition 2.1.3. An n-qubit state $|\psi\rangle$ is said to be *separable* if it may be written as a tensor product of single qubit states:

$$|\psi\rangle = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \ldots \otimes |\psi\rangle_n$$

Thus we would refer to the previous example state $|\psi\rangle_{AB}$ as a separable state. However, there are multiple-qubit states that are not separable and cannot be decomposed into a tensor product of single-qubit states. In this case, we say that the qubits are *entangled*.

Definition 2.1.4. The qubits composing a quantum state $|\psi\rangle$ are said to be *entangled* if $|\psi\rangle$ is not separable.

Example 2.1.3. Suppose we have the two-qubit state $|\psi\rangle_{AB} = \frac{1}{\sqrt{2}}(|00\rangle_{AB} + |11\rangle_{AB})$, in this case we cannot separate $|\psi\rangle_{AB}$ into a tensor product of states $|\psi\rangle_A$ and $|\psi\rangle_B$ in order to describe the overall system. If we now try to measure qubits A and B in the standard basis, we will find that they always observe the same measurement outcome, regardless of how spatially separated the two qubits are physically. For example, if we measure qubit A in the standard basis we will obtain a measurement outcome corresponding to the $|0\rangle$ or $|1\rangle$ state with equal probability. From $|\psi\rangle_{AB}$ we know that if $|\psi\rangle_A = |0\rangle$ then $|\psi\rangle_B$ must be $|0\rangle$ after the measurement as well and if we measure it in the same basis we will obtain the same measurement outcome. This particular property of qubits is one of the fundamental advantages that allow the execution of novel protocols with quantum networks.

Similarly to classical networks, data in quantum networks may be corrupted due to noise introduced in storage or transmission. Classically, such noise manifests itself in the form of bit-flipping in the data or in failed reception of data. In quantum systems, noise may perturb the data stored in a qubit to slightly deviate from its original state. For example, a qubit that was originally in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ may change due to environmental noise into a new state $|\psi\rangle = \delta|0\rangle + \gamma|1\rangle$. In order to quantify the quality of a quantum state in relation to an ideal state, a metric known as *fidelity* is often used.

Definition 2.1.5. For two pure states $|\psi\rangle, |\phi\rangle$ the fidelity F is computed as:

$$F = |\langle\psi|\phi\rangle|^2 = |\langle\phi|\psi\rangle|^2$$

Decoherence is the loss of fidelity due to manipulation or environmental noise and the rate at which decoherence reduces the fidelity of a quantum state depends on the physical process used to realize the qubit. As a consequence, the measurement statistics of the system may be changed which may lead to unexpected measurement outcomes. One wishes to mitigate the effects of decoherence so that applications operate correctly.

2.1.2. Qubit Transmission

One of the ways to transmit a qubit is to encode its quantum state in a photon over optical fiber. Several encodings are possible such as using the polarization of a photon [2], presence/absence of a photon [3–5], or a time-bin encoding [6, 7] that depends on the early/late arrival time of a photon. However, the probability of successful transmission of a qubit decreases exponentially with the distance the photon must travel, making the transmission of a qubit in a single attempt unlikely. Classically, such lossy behavior is overcome by using store-and-forward mechanisms where the data to be transmitted is copied and repeatedly transmitted until success. Unfortunately, these mechanisms do not translate into quantum networks due to the *no-cloning theorem* which effectively states that an arbitrary unknown quantum state $|\psi\rangle$ cannot be duplicated. Instead, transmission of qubits can be achieved using an indirect method known as teleportation.

Teleportation

Quantum teleportation [8] is a mechanism to transmit an arbitrary quantum state $|\psi\rangle$ by consuming a previously existing entangled pair of qubits. Figure 2.1 shows the idea of quantum teleportation. Suppose that two nodes A and B in a quantum network share an entangled pair of qubits and A wishes to send B a quantum state. First, A prepares the desired state locally and then performs a special measurement known as a *Bell measurement* on its local qubits. This measurement is performed on both qubits at the same time and is known as a *joint* measurement. This measurement yields one of four measurement outcomes (one measurement outcome per qubit resulting in four combinations) and the outcome is

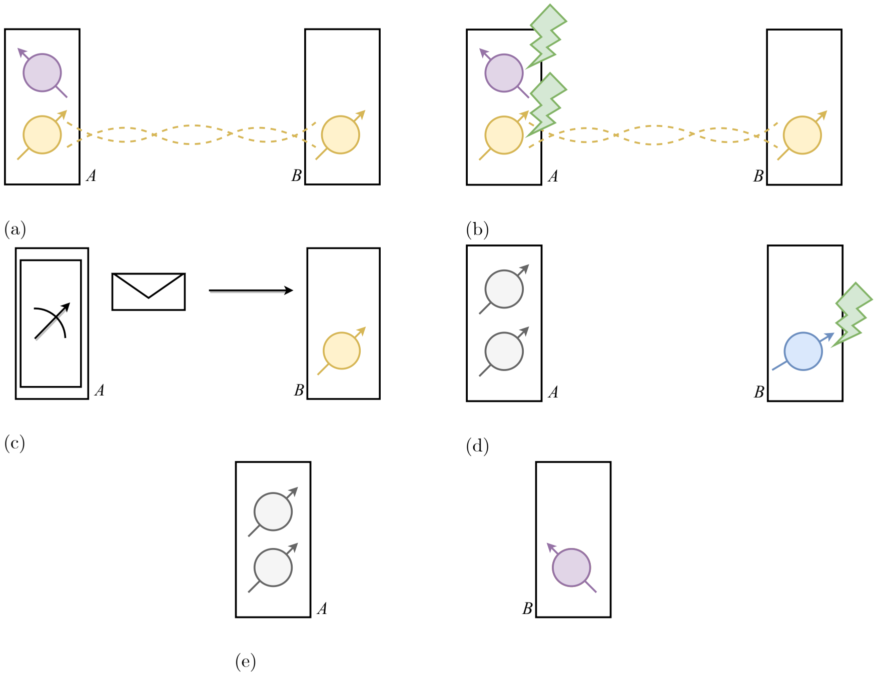


Figure 2.1: Control flow of quantum teleportation of an arbitrary qubit (purple) using an entangled pair of qubits (yellow). a) A and B share a pair of entangled qubits and A prepares a qubit it wishes to transmit. b) A locally performs the joint Bell-state measurement on its qubits. c) A measures the qubits and sends the measurement outcomes to B thereby consuming the entanglement that was shared. d) B performs corrections on its qubit based on the measurement outcomes. e) B now holds the qubit that A had originally prepared.

sent to B. Upon receiving the measurement outcome, B may complete the teleportation process by applying corrections to its local qubit which was previously entangled with A's qubit. Note that the entanglement between A and B is consumed in this process.

Teleportation allows the transmission of qubits between quantum devices in the network without physically transmitting the qubit itself. This may be done if the devices each hold one of two qubits that are entangled and have access to a classical channel by which they may communicate measurement outcomes. One may thus think of the entanglement used for such transmission as an *entangled link* over which data may be sent. Classical channels between devices are realized using classial networks, like the Internet. We may thus proceed to describe how the required entanglement may be shared between spatially separated devices.

Entanglement Swapping and Distillation

While direct transmission of qubits may be avoided for sending data in quantum networks, it is required for establishing entangled links between devices. Figure 2.2 shows a process by which such an entangled link may be established using *heralded entanglement generation* [5]. Suppose that there are two quantum devices A and B that wish to create an entangled link and they each have optical fibers that connect to a special device known as a *heralding station*. Both A and B prepare a two-qubit entangled state such that they hold one of the qubits locally while transmitting the other to the heralding station using a photon. When the photons arrive at the heralding station, a Bell state measurement is performed using simple linear optics and photon detectors. When measured in this fashion, the state of one of the photons is teleported to the other device thereby establishing an entangled pair of qubits between A and B. To complete this process, it is necessary for the heralding station to inform A and B of the measurement outcome to confirm entanglement was established.

We stress that it is essential that the photons arrive at the heralding station in synchronicity in order for this process to succeed, thus strict coordination on photon transmission is required between the nodes. If only one of the nodes transmits a photon to the midpoint or they arrive asynchronously, entanglement cannot be generated. Heralding station midpoints are a component of the link infrastructure and do not appear as nodes in the network. This means that coordination is only required between the nodes establishing an entangled link.

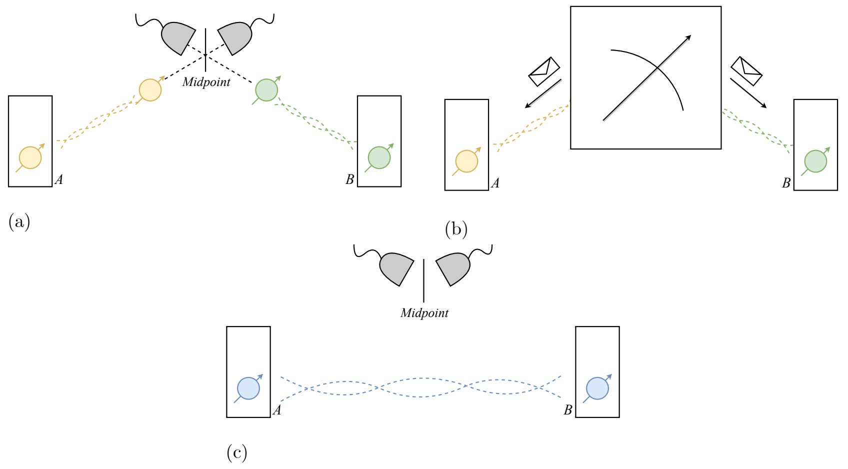


Figure 2.2: Heralded entanglement generation between two quantum devices. a) A and B transmit a qubit that is entangled with one they hold locally to the midpoint heralding station. b) The heralding midpoint performs a Bell measurement and sends the outcome to A and B. c) A and B share entanglement.

Due to the lossy nature of optical fibers it is unlikely that the photons from A and B will arrive at the heralding station in a single attempt. Thus, A and B may repeatedly prepare their two qubit states and transmit photons to the midpoint until both have successfully reached the midpoint and an entangled pair is successfully established. This method of establishing entanglement is commonly referred to as heralded entanglement generation as the entanglement is heralded by the midpoint's measurement outcome. The resulting established entanglement is referred to as an *elementary link*. For the remainder of the text we use the terms establishing and generating equivalently when referring to creation of an elementary link.

Heralded entanglement generation provides a method for establishing entanglement between devices that have access to a common heralding station. Now suppose we have three devices A, B and C such that a midpoint additionally lies between B and C. In order to generate entanglement between A and C we may have the pairs of devices (A, B) and (B, C) execute the heralded entanglement generation procedure to first establish entanglement with one another. At this point, B may perform the teleportation process in order to send one of it's entangled qubits to either A or C in order to establish entanglement. When teleportation is used in this manner to establish entanglement between two network devices, it is referred to as *entanglement swapping* [8, 9].

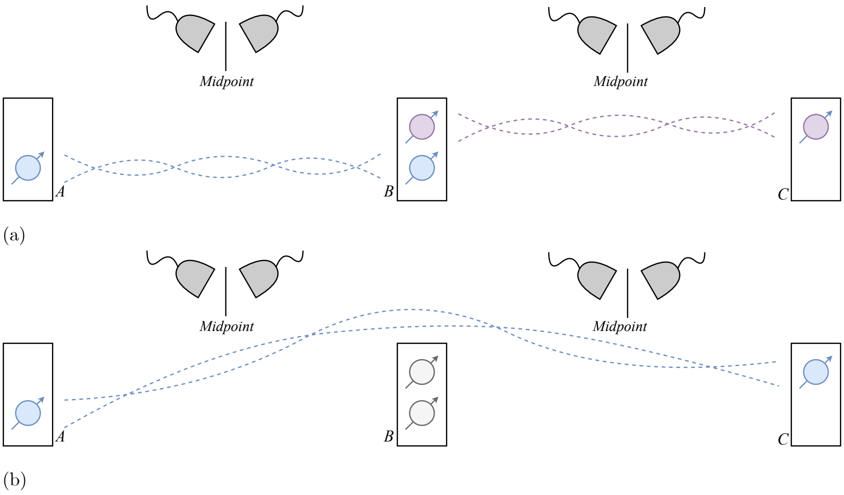


Figure 2.3: Establishing entanglement between A and C by performing entanglement swapping at B. a) The pairs (A, B) and (B, C) create entanglement with one another using heralded entanglement generation. b) B executes teleportation in order to deliver shared entanglement between A and C.

An important remark regarding the use of entanglement swapping in establishing entanglement between devices is that the fidelity of the final pair of qubits becomes lower than the fidelity of either of the links consumed in the process. Additionally, the amount of delay between successfully generating the entangled links used for entanglement swapping introduces decoherence on the qubits and reduces their resulting fidelity. In order to overcome these obstacles, a technique known as *entanglement distillation* [10–12] is used whereby several pairs of entangled qubits may be combined together to produce a fewer number of entangled qubits with higher fidelity than the underlying pairs.

In order to provide high-fidelity entanglement in multi-hop quantum networks where the elementary links are not capable of producing perfect fidelity entangled pairs, it is crucial that entanglement distillation is performed to increase the fidelity of the entangled qubits delivered to nodes.

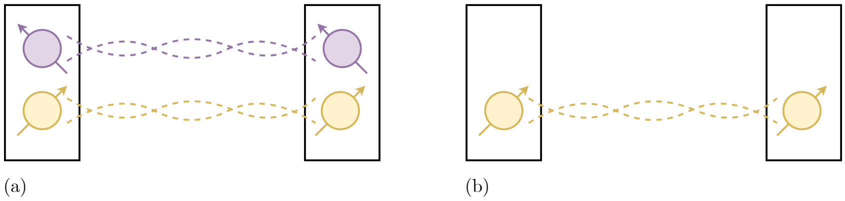


Figure 2.4: Entanglement distillation turns multiple pairs of entangled qubits (a) into a single pair of entangled qubits with higher fidelity (b).

Quantum Repeater Protocols

A *quantum repeater chain* is a path of quantum network devices that execute sequences of elementary link generations, entanglement swaps, and entanglement distillations known as *quantum repeater protocols* [13–16] to establish an entangled pair of qubits between the devices at the end of the repeater chain. Depending on how the entanglement is to be used, the protocol must deliver entangled qubits that satisfy a minimum fidelity with a maximum latency. The devices internal to the path are commonly referred to as *quantum repeaters*. These devices function analogously to repeaters found in classical networks used to relay information.

2.1.3. Network Elements

Here we summarize the elements of a quantum network and terminology that will be used throughout the remainder of the thesis. We refer to a node in the network as an *end node* if it runs applications which consume entanglement provided by the network whereas a *repeater node* is solely dedicated to executing repeater protocols. Pairs of nodes in a quantum network that can perform elementary link generation, by e.g. heralded entanglement generation, are said to be *directly* connected whereas nodes that are connected over multiple hops are *indirectly* connected. It is assumed that the classical communications involved in a quantum network take place over a classical network such as the Internet and any pair of nodes in the network are capable of exchanging messages with one another.

2.1.4. State of the Art Devices

We conclude the background on quantum networks with a short summary of state of the art technologies. Depending on the supported application protocols, a quantum network may be categorized into a stage

[17]. Currently, devices used for distributing secret key (QKD [18, 19]) at distances of approximately 100 km have been deployed [20, 21] while longer distances have been reached using coiled fiber [22, 23]. Such devices realize the prepare-and-measure stage of quantum networks and cannot be used to perform end-to-end transmission of qubits but may be chained together to realize a trusted repeater stage network [24]. These provide secure communications between the end nodes assuming that all intermediate nodes are trusted.

Devices that may be used for delivering end-to-end entanglement in entanglement generation networks are still under development. The current record of producing heralded entanglement is 1.3 km in solid state quantum devices built from nitrogen-vacancy (NV) centres in diamond [25]. Demonstrations of devices for quantum memory and few-qubit fault-tolerant stage networks have been performed in a lab setting [26].

At present, these quantum devices fall under a classification deemed as Noisy Intermediate-Scale Quantum (NISQ) devices [27] and are characterized by their imperfect control of qubits and limited number of qubits. The noise introduced due to imperfect control places limitations on what is achievable with such devices in the near future and necessitates considerable effort in engineering systems that may become more and more useful. Our contribution in this work is targeted at near-term NISQ devices but may additionally see use in future devices as well.

2.2. Medium Access Control

Classical communication networks, like wireless networks, make use of shared transmission mediums to propagate data between nodes in the network. In order to allow multiple data streams and signals to be active simultaneously, it is necessary to coordinate the usage of shared transmission mediums. Such networks achieve this through the use of medium access control (MAC) protocols found in the data link layer of the OSI model. In this section, we will provide background on the usage of MAC protocols in classical networks with specific attention to time-division multiple access protocols. Understanding the motivation and concepts behind MAC protocols will be useful for understanding how entanglement generation can be coordinated for repeater protocols in quantum networks.

2.2.1. MAC Protocols and Channel Access Methods

The MAC sublayer is responsible for handling interactions with a transmission medium connected to a device through the use of flow control and multiplexing. These transmission mediums may be wired, optical, or wireless mediums and are referred to as *collision domains* when simultaneous data transmissions collide with one another. For example, wireless communications share airspace as a medium of broadcasting data using radio frequencies. If two wireless transmitters attempt transmission at the same time to a common receiver, the radio frequencies overlap at the receiving antennae resulting in corrupted reception of data.

MAC protocols make use of coordination mechanisms that are referred to as *channel access methods* to coordinate transmissions between network nodes. These access methods come in a variety of forms and the specific type used depends on the type of shared medium as well as the desired behavior of the network. An example of a commonly used channel access method is frequency-division multiple access (FDMA) where data streams are allocated to different frequency bands to avoid collisions [28]. A corresponding MAC protocol is then responsible for ensuring that outgoing data streams adhere to their frequency band assignment. While there are many other flavors of channel access methods, this thesis will focus on time-division multiple access (TDMA).

2.2.2. Time-Division Multiple Access

Time-division multiple access (TDMA) is a channel access method that builds upon time-division multiplexing [29]. This type of multiplexing partitions the time domain into recurrent time slots of fixed length and assigns data streams to time slots in order to grant dedicated access to the shared medium. Doing so provides the appearance of logically handling multiple data streams simultaneously while physically the streams take turns propagating on the medium. Time-division multiplexing was first used in telegraphy in order to route multiple transmissions over a single transmission line and continues to see use in satellite communication systems [30, 31], wireless mesh networks [32], as well as wireless sensor networks [33, 34] due to several advantages they hold over other channel access methods.

Wireless sensor networks are composed of many small, low-power, measurement devices that collect data from an environment and aggregate it at a sink node in the network which is responsible for forwarding the data to an observer [35]. Due to their low-power and low-cost nature, these wireless devices are often restricted to a single transmission frequency and make use of TDMA protocols in order to avoid transmission collisions. In addition to providing collision-free transmission of data, these protocols have the added advantage that nodes are only required to listen and broadcast in their own time slot. For example, if a node in a wireless network does not relay data for a stream in the current time slot, the node may enter a low-power state to reduce unnecessary power consumption. Figure 2.5 shows a simple illustration of TDMA in a wireless sensor network where A is the sink node connected to an observer. In practice, a schedule for such a network is derived by first taking the network graph (figure 2.5a) and constructing the corresponding conflict graph (2.5b) that represents how simultaneous transmission collide. Note in this case that nodes connected in the original graph are also connected in the conflict graph as they are not able to receive transmissions at the same time they attempt to transmit them.

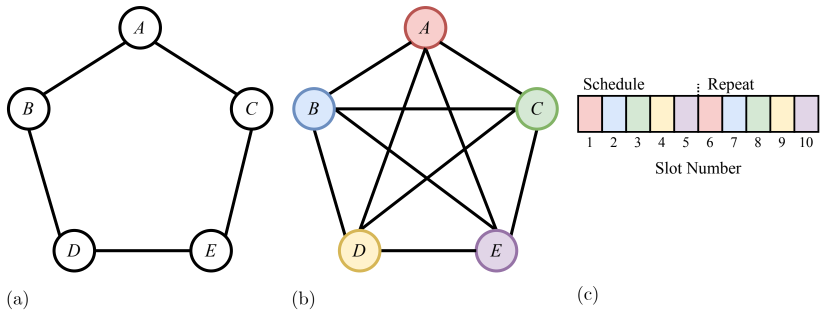


Figure 2.5: A TDMA scheduling example on a wireless network demonstrating collision avoidance. a) Wireless sensor network of five nodes. Edges represent pairs of nodes that may transmit data to one another. b) Transmission conflict graph of wireless network. Edges represent pairs of nodes that experience transmission collisions when transmitting simultaneously. c) TDMA transmission schedule to avoid transmission collision. The schedule repeats after slot five.

Once a conflict graph has been derived, a TDMA schedule may be constructed by coloring the resulting conflict graph. The resulting coloring is mapped to slots in a TDMA schedule where nodes of the same color may transmit in the same slot. In this example, the conflict graph is a complete graph and only one node may transmit in any slot of the TDMA schedule to guarantee successful reception at all times. In figure 2.5c we see that the schedule repeats after the fifth slot and as a result, the schedule held by the nodes need only consist of these five slots and may be executed in a cyclic fashion.

A variant of TDMA known as dynamic TDMA has the added advantage of being able to provide higher qualtiy of service (QoS) guarantees. In wireless mesh networks, nodes have many interconnections and the majority of research pertains to routing and transmission coordination in order to provide high QoS for data streams [36]. The nodes in mesh networks are often placed close to one another and can have connections that support high data rates. By dynamically constructing schedules and allocating proportionate shares of the slots in the schedule to different streams, different levels of QoS requirements can be accompanied alongside one another.

Suppose that we have a mesh network as shown in figure 2.6a with links that support a throughput of 12 Mbps and that there are two demands on the network. A and B desire a connection that provides a throughput of 2 Mbps while D and E desire a connection that provides a throughput of 1 Mbps. Note that both connections share C as a relay. We can use a TDMA schedule as shown in 2.6c where the slot length is $\frac{1\text{ s}}{12} \approx 85$ ms long to satisfy these demands. Specifically, timeslots 1-2 allow A to transmit 1 Mb of data to B while time slots 3-4 permit B to transmit 1 Mb of data to A. Slots 5-8 operate similarly between D and E. Slots 9-12 provide an additional 1 Mb of data transfer between A and B and the schedule may be repeated.

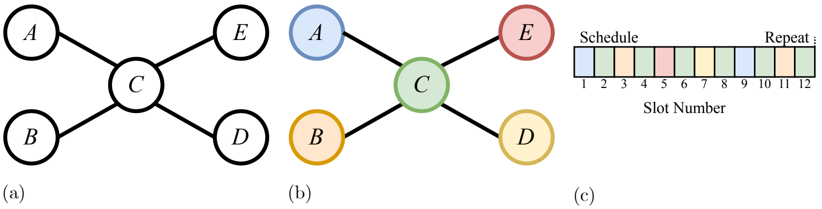


Figure 2.6: A TDMA scheduling example on a wireless network demonstrating QoS. a) Wireless sensor network of five nodes. Edges represent pairs of nodes that may transmit data to one another. b) Colored network for transmissions. c) TDMA transmission schedule to satisfy QoS demands. The path between A and B occurs twice as frequently than the path between C and D.

We remark that TDMA protocols suffer from disadvantages compared to other channel access methods. First, high levels of timing synchronization are required to ensure nodes adhere to the TDMA schedule appropriately. Improper calibration of clocks may result in clock drift and as a result nodes may have disagreeing views on what the current time slot is. Second, while TDMA schedule can provide higher QoS guarantees on data streams, underutilized time slots can cause a decrease in overall network throughput when nodes overallocate data streams or do not have sufficient data to fully utilize reserved time slots. Finally, in dynamic TDMA there is an inherent trade-off between the quality of a produced schedule and the complexity of the scheduling algorithm that is used to construct it.

2.3. Scheduling

While TDMA protocols provide a method for *how* transmissions should be coordinated and scheduled in a network, it does not provide a mechanism for *when* these transmissions should occur. This is left as a responsibility to an underlying TDMA scheduling algorithm. In dynamic TDMA, transmission schedules need to be constructed on the fly as demands arise in the network. Here, we provide an overview of real-time scheduling [37] and the resource-constrained project scheduling problems [38] as well as terminology that will be used when we discuss how TDMA schedules can be constructed for delivering entanglement in networks of quantum processors.

2.3.1. Real-Time Scheduling

A system is considered *real-time* if the correctness of its behavior depends not only on the logical correctness of its operations but additionally on the time at which they are executed. Such operations are typically associated with a deadline before which they must complete. Real-time systems are considered *hard* if missing a deadline causes a catastrophic failure in the system whereas a system is *soft* if missing a deadline causes a degradation in quality of service. When operations are repeated periodically with specific rates, the problem of scheduling the operations in a cyclic manner is known as periodic real-time scheduling.

A *periodic task* τ_i corresponds to a generic operation that needs to be performed in a real-time system. An *instance* $\tau_{i,j}$ of a periodic task τ_i corresponds to the jth instance of τ_i in the system. Depending on the cyclic schedule, a periodic task may correspond to a single or multiple instances in the system.

The *release time* $r_{i,j}$ of a task instance $\tau_{i,j}$ denotes the earliest time at which the jth instance of τ_i is able to start. An instance's *absolute deadline* $d_{i,j}$ corresponds to the deadline of instance $\tau_{i,j}$ and specifies the latest point at which $\tau_{i,j}$ must complete. Each periodic task τ_i is associated with a *phase* Φ_i denoting the release time $r_{i,1}$ of the first instance $\tau_{i,1}$ of a periodic task τ_i. A *relative deadline* D_i for task τ_i denotes the amount of time between the release time $r_{i,j}$ and absolute deadline $d_{i,j}$ of all task instances $\tau_{i,j}$. Each task is also associated with a worst-case execution time C_i that specifies the maximum amount of time an instance $\tau_{i,j}$ of τ_i takes to complete. $\sigma_{i,j}$ and $f_{i,j}$ denote the *starting* and *finishing* times of a task instance $\tau_{i,j}$. Figure 2.7 shows a visual aid for understanding these quantities. A set of tasks is denoted with T.

In addition, several assumptions are made about the real-time system. Namely,

1. The instances of a periodic task τ_i are released into the system at a constant rate known as the *period* T_i. That is, two consecutive instances $\tau_{i,j}$ and $\tau_{i,j+1}$ have a difference in their release times $r_{i,j+1} - r_{i,j}$ equal to T_i.

2. All instances of a periodic task τ_i have the same relative deadline D_i, which is equal to the period T_i.

3. All instances of a periodic task τ_i observe the same worst-case execution time C_i.

4. All periodic tasks in the set of tasks T are independent and observe no precedence relations. That is, they may be executed in any order.

5. All overheads in the system are assumed to be zero. This means the system is capable of starting a task without any delay.

6. All parameters of the task set T are integer-valued and time is assumed to pass in discrete steps.

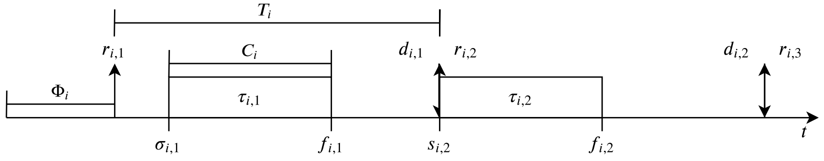


Figure 2.7: Visual representation of parameters associated with a periodic task τ_i. There is a single task τ_i shown with two instances $\tau_{i,1}$ and $\tau_{i,2}$.

The goal of periodic task scheduling is to produce a schedule $\mathcal{S}$ that assigns start times to periodic task instances such that the release and deadline constraints imposed by a task set T are satisfied. A schedule is said to be *optimal* if it satisfies the constraints of the task set and additionally results in a schedule of minimum length. To this end, determining an optimal schedule for a task set is NP-hard [39]. This means that there is no algorithm for computing an optimal schedule in time that scales polynomially with the number of tasks. Thus most instances of periodic task scheduling make use of scheduling heuristics to give approximate solutions to schedule construction.

Tasks are said to be *non-preemptable* if they may not be interrupted once they have been started. We say that a set of tasks T is *feasible* if it may be scheduled on a uniprocessor without violating any of the release or deadline constraints of the tasks. A non-preemptive scheduling algorithm is said to be *optimal* if it can schedule any task set that is feasible on a uniprocessor using a scheduling decision that runs polynomially in the number of tasks. A non-preemptive scheduling algorithm is said to be *work conserving* if it keeps the uniprocessor busy when there are tasks to be executed. In this domain, Jeffay et. al [39] have shown that non-preemptive earliest-deadline first (EDF) is optimal in the case when a task set T is scheduleable for any assignment of phases $\Phi_1, ..., \Phi_n$. Nasri et. al [40] have refined this claim to state that it is not optimal in the case of concrete periodic task sets that specify a set of release offsets. In this case, the problem of deciding feasibility of a concrete periodic task set is known to be NP-hard in the strong sense. This means there is no efficient algorithm for deciding feasibility regardless of the numerical parameters of the set of tasks.

A task set T is often associated with additional parameters such as a *hyperperiod* H denoting the minimum interval of time after which the schedule repeats. If H is the length of this interval then a schedule in the interval $[0, H]$ will be the same as in the interval $[kH, (k+1)H]$ for $k > 0$. When $\Phi_i = 0, \forall \tau_i \in \mathrm{T}$, H may be computed as the least common multiple (lcm) of the task periods $lcm(T_1, ..., T_n)$.

The *instance response time* $R_{i,j}$ of a task instance $\tau_{i,j}$ is the amount of time from the instance's release time $r_{i,j}$ to its finishing time $f_{i,j}$ (computed as $f_{i,j} - r_{i,j}$) and the *task response time* of periodic task τ_i is the maximum response time of all its instances. The worst-case response time is the maximum of all task response times.

The *processor utilization factor* of a task set T of n tasks is the fraction of time spent executing the task set. Since we assume that all task instances $\tau_{i,j}$ corresponding to task τ_i observe the same worst-case execution time C_i, the fraction of time spent executing task τ_i is $\frac{C_i}{T_i}$. Furthermore, because C_i is the worst-case execution time, this fraction also corresponds to a worst-case. The processor utilization factor for the entire set of n tasks T is $U = \sum_{i=1}^{n} \frac{C_i}{T_i}$.

2.3.2. Resource-Constrained Project Scheduling

The resource-constrained project scheduling problem (RCPSP) is traditionally used to represent project management in the presence of scarce resource constraints and precedence relations between activities and can be explained in the following way. A project is represented by a set J of activities $j_1, ..., j_n$ and each activity j_i is associated with a processing time p_i. Activities are non-preemptable and may not be interrupted once they have been started. Technological requirements may impose precedence relations on the set of activities. With each activity j_i we associate a set of activities P_i that must be completed before j_i is permitted to begin. These precedence relations of a project are typically represented using an activity-on-node network.

In addition to precedence constraints, activities require certain amounts of resources to be performed. Resources are said to be renewable when their full capacity is available in every period. The set of renewable resources is represented as K and each resource $k_i \in K$ additionally specifies a capacity H_i that is assumed to be constant over time. Activity j_l requires h_{li} units of resource k_i during each period it is executed. A project additionally contains two dummy activities j_0 and $j_{|J|+1}$ denoting the start and end time of a project. Neither of these activities require any resources and have processing times $p_0 = p_{|J|+1} = 0$.

In RCPSP, all information is assumed to be known in advance and the parameters are assumed to be non-negative and integer-valued. A schedule $\mathcal{S}$ is an assignment of start times $s_0, s_1, ..., s_n, s_{|J|+1}$ to the activities $j_0, j_1, ..., j_n, j_{|J|+1}$ that satisfies the resource and precedence constraints of the project. The *makespan* of such a schedule is the time difference between starting and completing the entire set of activities.

Precedence relations may additionally specify timing constraints in the form of minimal and maximal time lags between jobs. That is, a minimal time lag d_{ik} imposes the constraint that activity j_k starts no earlier than d_{ik} time units after the completion time f_i of j_i. A maximal time lag $\bar{d}_{ik}$ imposes the constraint that activity j_k has a starting time s_k no later than $\bar{d}_{ik}$ time units after the completion time f_i of activity j_i. More formally, these may be written as $f_i + d_{ik} \leq s_k$ and $f_i + \bar{d}_{ik} \geq s_k$

respectively.

Example 2.3.1. Figure 2.8 shows an example of an activity-on-node network with processing times, resource requirements, and timing constraints depicted. Figure 2.9 shows a valid scheduling for the project when $H_1 = 1$ and $H_2 = 2$ for resources k_1 and k_2 respectively. Note that if $H_1 = 2$ then a shorter schedule could be produced by having j_1 and j_2 execute in parallel as well as j_5 and j_6.

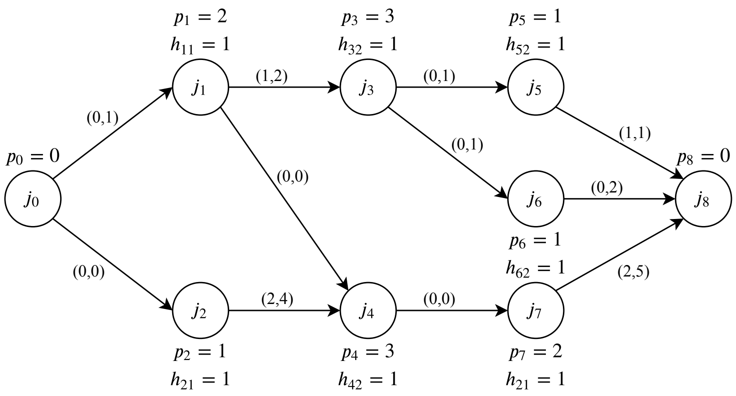


Figure 2.8: Example of an activity-on-node network for a project. Edges are labeled with pairs $(d_{ik}, \bar{d}_{ik})$ to denote minimal/maximal time lags between activities (e.g. j_4 must begin at least 2 time units after j_2 and also at most 4 time units after j_2). j_0 and j_8 are dummy start and end activities with 0 execution time.

The standard RCPSP assumes that each activity can only be executed in a single way as determined by a single execution time with fixed resource requirements. Activities can be extended to allow several combinations of execution time and resource requirements called modes in which the activity can be performed. In this formulation, each activity j_i must be performed in one of its modes denoted by a label $1, ..., M_{j_i}$ where M_{j_i} is the total number of modes for j_i. Once a mode has been selected, the activity may not switch modes and may not be preempted. The execution of an activity j_i in mode m has execution time p_{im} and denotes the requirement of resource k_j as h_{imj}. This version of the RCPSP is often denoted as multi-mode resource-constrained project scheduling (MRCPSP) and in the case where each activity has a single mode we obtain the standard RCPSP.

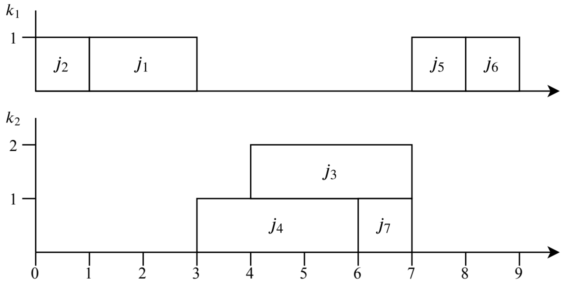


Figure 2.9: A visualization of a schedule for the activity-on-node network in 2.8. j_0 and j_8 start at $t = 0$ and $t = 9$ respectively. These timelines show the amount of resource usage over time as the activites are executed.

Similarly to periodic task scheduling, determining if an instance of RCPSP with timing constraints can be feasibly scheduled is known to be NP-hard as shown by Bartusch et al. in [41] and constructing an optimal schedule that minimizes the makespan is NP-complete [42]. Typical strategies for solving an RCPSP involve exact and heuristic methods. We refer the reader to [38] and [43] for a more comprehensive description of the RCPSP as well as these exact and heuristic methods.

2.4. Related Work

We conclude this chapter with a summary of related works. The domain of quantum network architecture is not well studied and several works have been published that propose designs and protocols to meet this end. Dahlberg et al. propose a functional allocation of a quantum network stack (figure 2.10) as well as physical and link layer protocols in [44]. In this work, a detailed network simulation is used to analyze the performance of the protocols using models of NV centres in diamond and heralded entanglement generation at the physical layer. In [45], Matsuo et. al present and simulate a RuleSet-based quantum link bootstrapping protocol that may be used to install rules that provide flexibility when establishing connections in quantum networks. Their procedure also uses link tomography to quantify quantum link fidelity and throughput which may be used for routing. Aparicio et. al investigate various multiplexing schemes for quantum repeater networks in [46]. They simulate a thirteen node network with up to five different flows and recommend the use of statistical multiplexing for shared quantum repeater networks.

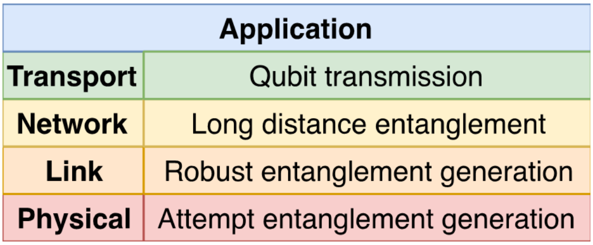

Application
Transport
Qubit transmission
Network
Long distance entanglement
Link
Robust entanglement generation
Physical
Attempt entanglement generation

Figure 2.10: Proposed functional allocation of quantum network stack [44]. Physical and Link Layer protocols are responsible for establishing entanglement between connected nodes in the network. The Network Layer is responsible for using entanglement between connected nodes to connect nodes separated by multi-hops using repeater protocols. The Transport Layer is responsible for managin data transmission while the Application Layer makes use of these services to execute higher-level protocols with other nodes in the network.

Dynamic TDMA protocols have been studied in the context of many different network technologies such as in ad-hoc networks [47], wireless ATM networks [48], wireless powered communication networks [49], and satellite communication systems [50]. There is an extensive amount of literature on TDMA schemes that range from centralized to distributed models [51, 52].

References

[1] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information,* (2002).

[2] J.-C. Boileau, R. Laflamme, M. Laforest, and C. Myers, *Robust quantum communication using a polarization-entangled photon pair,* Physical review letters **93**, 220501 (2004).

[3] H.-W. Lee and J. Kim, *Quantum teleportation and bell's inequality using single-particle entanglement,* Physical Review A **63**, 012305 (2000).

[4] D. T. Pegg, L. S. Phillips, and S. M. Barnett, *Optical state truncation by projection synthesis,* Physical review letters **81**, 1604 (1998).

[5] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. Taminiau, M. Markham, D. J. Twitchen, L. Childress,

et al., *Heralded entanglement between solid-state qubits separated by three metres,* Nature **497**, 86 (2013).

[6] I. Marcikic, H. de Riedmatten, W. Tittel, V. Scarani, H. Zbinden, and N. Gisin, *Femtosecond time-bin entangled qubits for quantum communication,* arXiv preprint quant-ph/0205144 (2002).

[7] I. Marcikic, H. De Riedmatten, W. Tittel, H. Zbinden, M. Legré, and N. Gisin, *Distribution of time-bin entangled qubits over 50 km of optical fiber,* Physical Review Letters **93**, 180502 (2004).

[8] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, *Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels,* Physical review letters **70**, 1895 (1993).

[9] M. Zukowski, A. Zeilinger, M. A. Horne, and A. K. Ekert, *" event-ready-detectors" bell experiment via entanglement swapping.* Physical Review Letters **71** (1993).

[10] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, *Purification of noisy entanglement and faithful teleportation via noisy channels,* Physical review letters **76**, 722 (1996).

[11] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, *Quantum privacy amplification and the security of quantum cryptography over noisy channels,* Physical review letters **77**, 2818 (1996).

[12] J. Dehaene, M. Van den Nest, B. De Moor, and F. Verstraete, *Local permutations of products of bell states and entanglement distillation,* Physical Review A **67**, 022310 (2003).

[13] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, *Quantum repeaters: the role of imperfect local operations in quantum communication,* Physical Review Letters **81**, 5932 (1998).

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

[15] L. Jiang, J. M. Taylor, N. Khaneja, and M. D. Lukin, *Optimal approach to quantum communication using dynamic programming,* Proceedings of the National Academy of Sciences **104**, 17291 (2007).

[16] W. J. Munro, K. Azuma, K. Tamaki, and K. Nemoto, *Inside quantum repeaters,* IEEE Journal of Selected Topics in Quantum Electronics **21**, 78 (2015).

[17] S. Wehner, D. Elkouss, and R. Hanson, *Quantum internet: A vision for the road ahead,* Science **362**, eaam9288 (2018).

[18] C. H. Bennett and G. Brassard, *Proceedings of the ieee international conference on computers, systems and signal processing,* (1984).

[19] A. K. Ekert, *Quantum cryptography based on bell's theorem,* Physical review letters **67**, 661 (1991).

[20] A. Extance, *Fibre systems.(2017),* Retrieved May **27**, 2019 (2017).

[21] T. Inagaki, N. Matsuda, O. Tadanaga, M. Asobe, and H. Takesue, *Entanglement distribution over 300 km of fiber,* Optics express **21**, 23241 (2013).

[22] A. Boaron, G. Boso, D. Rusca, C. Vulliez, C. Autebert, M. Caloz, M. Perrenoud, G. Gras, F. Bussières, M.-J. Li, *et al.*, *Secure quantum key distribution over 421 km of optical fiber,* Physical review letters **121**, 190502 (2018).

[23] P. A. Hiskett, D. Rosenberg, C. G. Peterson, R. J. Hughes, S. Nam, A. Lita, A. Miller, and J. Nordholt, *Long-distance quantum key distribution in optical fibre,* New Journal of Physics **8**, 193 (2006).

[24] L. Salvail, M. Peev, E. Diamanti, R. Alléaume, N. Lütkenhaus, and T. Länger, *Security of trusted repeater quantum key distribution networks,* Journal of Computer Security **18**, 61 (2010).

[25] B. Hensen, H. Bernien, A. E. Dréau, A. Reiserer, N. Kalb, M. S. Blok, J. Ruitenberg, R. F. Vermeulen, R. N. Schouten, C. Abellán, *et al.*, *Loophole-free bell inequality violation using electron spins separated by 1.3 kilometres,* Nature **526**, 682 (2015).

[26] S. Barz, E. Kashefi, A. Broadbent, J. F. Fitzsimons, A. Zeilinger, and P. Walther, *Demonstration of blind quantum computing,* science **335**, 303 (2012).

[27] J. Preskill, *Quantum computing in the nisq era and beyond,* Quantum **2**, 79 (2018).

[28] J. Olenewa, *Guide to wireless communications* (Cengage Learning, 2013).

[29] G. Miao, J. Zander, K. W. Sung, and S. B. Slimane, *Fundamentals of mobile data networks* (Cambridge University Press, 2016).

[30] K. Maine, C. Devieux, and P. Swan, *Overview of iridium satellite network,* in *Proceedings of WESCON'95* (IEEE, 1995) p. 483.

[31] M. Mazzella, M. Cohen, D. Rouffet, M. Louie, and K. Gilhousen, *Multiple access techniques and spectrum utilisation of the globalstar mobile satellite system,* in *Fourth IEE Conference on Telecommunications 1993* (IET, 1993) pp. 306–311.

[32] C. Vallati, *Dynamic resources allocation in wireless mesh networks,* in *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks* (IEEE, 2011) pp. 1–3.

[33] J. Degesys, I. Rose, A. Patel, and R. Nagpal, *Desync: self-organizing desynchronization and tdma on wireless sensor networks,* in *Proceedings of the 6th international conference on Information processing in sensor networks* (2007) pp. 11–20.

[34] G. Pei and C. Chien, *Low power tdma in large wireless sensor networks,* in *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No. 01CH37277)*, Vol. 1 (IEEE, 2001) pp. 347–351.

[35] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, *Wireless sensor networks: a survey,* Computer networks **38**, 393 (2002).

[36] D. D. Falconer, F. Adachi, and B. Gudmundson, *Time division multiple access methods for wireless personal communications,* IEEE Communications Magazine **33**, 50 (1995).

[37] R. I. Davis and A. Burns, *A survey of hard real-time scheduling for multiprocessor systems,* ACM computing surveys (CSUR) **43**, 1 (2011).

[38] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch, *Resource-constrained project scheduling: Notation, classification, models, and methods,* European journal of operational research **112**, 3 (1999).

[39] K. Jeffay, R. Anderson, and C. Martel, *On optimal, non-preemptive scheduling of periodic and sporadic tasks* (Department of Computer Science, University of Washington, 1988).

[40] M. Nasri and G. Fohler, *Non-work-conserving non-preemptive scheduling: motivations, challenges, and potential solutions,* in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)* (IEEE, 2016) pp. 165–175.

[41] M. Bartusch, R. H. Möhring, and F. J. Radermacher, *Scheduling project networks with resource constraints and time windows,* Annals of operations Research **16**, 199 (1988).

[42] S. Hartmann and D. Briskorn, *A survey of variants and extensions of the resource-constrained project scheduling problem,* European Journal of operational research **207**, 1 (2010).

[43] K. Neumann and J. Zimmermann, *Methods for resource-constrained project scheduling with regular and nonregular objective functions and schedule-dependent time windows,* in *Project Scheduling* (Springer, 1999) pp. 261–287.

[44] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, *et al.*, *A link layer protocol for quantum networks,* in *Proceedings of the ACM Special Interest Group on Data Communication* (2019) pp. 159–173.

[45] T. Matsuo, C. Durand, and R. Van Meter, *Quantum link bootstrapping using a ruleset-based communication protocol,* Physical Review A **100**, 052320 (2019).

[46] L. Aparicio and R. Van Meter, *Multiplexing schemes for quantum repeater networks,* in *Quantum Communications and Quantum Imaging IX*, Vol. 8163 (International Society for Optics and Photonics, 2011) p. 816308.

[47] S. Kamruzzaman and M. S. Alam, *Dynamic tdma slot reservation protocol for cognitive radio ad hoc networks,* in *2010 13th International Conference on Computer and Information Technology (ICCIT)* (IEEE, 2010) pp. 142–147.

[48] J.-F. Frigon, V. C. M. Leung, and H. C. B. Chan, *Dynamic reservation tdma protocol for wireless atm networks,* IEEE Journal on Selected Areas in Communications **19**, 370 (2001).

[49] X. Kang, C. K. Ho, and S. Sun, *Optimal time allocation for dynamic-tdma-based wireless powered communication networks,* in *2014 IEEE Global Communications Conference* (IEEE, 2014) pp. 3157–3161.

[50] D. K. Petraki, M. P. Anastasopoulos, A. D. Panagopoulos, and P. G. Cottis, *Dynamic resource calculation algorithm in mf-tdma satellite networks,* in *2007 16th IST Mobile and Wireless Communications Summit* (IEEE, 2007) pp. 1–5.

[51] E. Hossain and V. K. Bhargava, *A centralized tdma-based scheme for fair bandwidth allocation in wireless ip networks,* IEEE Journal on Selected Areas in Communications **19**, 2201 (2001).

[52] T. Salonidis and L. Tassiulas, *Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks,* in *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing* (2005) pp. 145–156.

3 Coordinating End-to-End Entanglement Generation

This chapter presents our contribution of a newly proposed network architecture that allows the use of time-division multiple access (TDMA) methods for coordinating end-to-end entanglement generation in multi-hop quantum networks. First, we will discuss the motivation for using dynamic TDMA to accommodate varying Quality of Service (QoS) demands that may arise from end nodes in a quantum network as well as advantages and disadvantages of this method. We then provide a description of our network model and its components as well as assumptions made. We then conclude this chapter with a formal problem statement that we address in the remainder of this thesis.

3.1. Motivation

There are several aspects of quantum networks that require consideration when designing a mechanism for managing entanglement establishment. Here, we will provide a summary of these considerations and motivate the use of TDMA to address them.

Agreement on Entanglement Generation Recall from chapter 2 that generating entanglement between pairs of nodes using heralded entanglement generation is a heavily coordinated process. A scheme like heralded entanglement generation requires that a pair of directly connected nodes transmit their photons to the midpoint such that the photons arrive synchronously [1–3]. In order to connect nodes over a multi-hop repeater chain, each pair of directly connected nodes along the chain must agree when they will generate entanglement with one another. In the case where qubit resources are limited and links may not be generated in parallel, additional agreement on the order that elementary links are generated is necessary.

Meeting Quality of Service Demands Due to environmental noise and decoherence, current state of the art quantum devices observe limited storage times of entanglement [3, 4]. It is necessary to perform elementary link generations, entanglement swaps, and entanglement distillations that underly multi-hop repeater protocols in a timely fashion to minimize noise that could cause an end-to-end entangled link to not meet QoS demands.

Connecting Multiple Users The problem of coordinating repeater protocols is further complicated when multiple end nodes desire entanglement from the network at the same time. Repeater chains that intersect at common network nodes contend for resources that are used to generate elementary links and it becomes necessary to implement a strategy that manages how such resources may be shared. In this sense, the resources at quantum network nodes form a collision domain analagously to shared transmission mediums found in e.g. wireless sensor networks. Channel access methods may be used to grant exclusivity to qubit resources that are needed for the successful generation of entanglement.

Service Flexibility Applications in quantum networks have differing QoS requirements in order to be executed correctly. Some applications, like QKD [5, 6], require a large number of entangled qubits but have no constraint on the rate at which they are delivered in order to satisfy correctness of the application. As a result, these applications may generate demands that tolerate higher latency and lower rate requirements. Other applications, like distributed quantum computing, may require multiple entangled qubits be present at the same time [7, 8]. In this case it is necessary that entanglement can be delivered at high rates with low latency in order to maintain correctness. In addition to latency and rate requirements, applications may require varying levels of minimum fidelity in order to correctly execute.

Ultimately, the desired fidelity limits the minimal latency of a repeater protocol while network congestion limits the rate at which it can be executed. By providing varying levels of fidelity, a quantum network can allocate resources to repeater protocols in a more flexible manner to increase the throughput of the network.

In order to provide flexibility, it is necessary that entanglement coordination is capable of accommodating different QoS demands that may arise from different applications. Directly connected nodes that need to generate entanglement with one another must hold schedules that specify which underlying resources should be used to establish entanglement with their neighbors and when to do so. Recall that entanglement swapping [9, 10] reduces the fidelity of a delivered entangled link. Such a schedule must additionally specify the required fidelity of the generated elementary links to make sure the effects of entanglement swapping do not reduce the fidelity of the final end-to-end link below acceptable thresholds. Furthermore, the improvement of fidelity upon performing entanglement distillation [11–13] depends on the fidelity of distilled links, thus we also wish to ensure that distillation improves fidelity to desired levels.

Shared Computing and Communication Devices As mentioned in the context of wireless sensor networks, TDMA schemes have the added advantage that nodes are only required to participate in networking activities when their slot schedule dictates so. Some quantum communication devices like NV in diamond operate as computational devices in addition to communication devices. By specifying a priori the time slices at which a device is required to participate, a node may decide when an appropriate time exists to utilize the local device for computations. This becomes more relevant in networks that do not have dedicated repeater nodes for all repeater chains and must rely on the cooperation of end nodes to facilitate repeater protocols. In this case, the applications compete with repeater protocols for usage of the device's resources.

Requirements While on the surface it appears that TDMA schemes may be used to address the previously mentioned considerations, there are many requirements needed in order to facilitate a successful realization. Maintaining time synchronization in TDMA schedules is a non-trivial problem [14, 15] and requires ample amounts of engineering in order to implement. Even in the case of a single link of a quantum network, tight timing synchronization is needed between the devices in order to coordinate the delivery of photons to a midpoint in a synchronized manner [1]. By scaling the network to larger sizes, timing synchronization becomes an increasingly important aspect in the feasibility and success of utilizing these techniques.

Another nontrivial matter is the construction of TDMA schedules as new demands arrive in real-time. In order to accommodate demands that are dynamically supplied to the network, the constructed schedule must prevent overconsumption of resources from different demands. A centralized controller is a convenient construct to collect demands and has the additional benefit of being a repository for node availability and link capabilities between nodes in the network. With knowledge of the network topology and the link capabilities, such a central controller may jointly perform routing calculations, repeater protocol construction, and scheduling in order to disperse resource utilization across the network's resources. Quantum network end nodes need to be able to request the set up of a new connection in the network in addition to modifying or removing connections when application requirements have changed or have been met. This incurs the additional requirements that the quantum network nodes have a mechanism with which to communicate with a central controller and install TDMA schedules that have been computed.

In order to construct a TDMA schedule in a dynamic fashion a quantum network may be viewed as a hard real-time system where repeater protocols are periodic tasks that must be scheduled onto network resources of the nodes in the network. Under this model, a central controller is responsible for collecting the end-to-end demands of the end nodes in the network and coordinating the execution of repeater protocols in the network.

3.2. Network Model

Here we will describe our model of a quantum network along with the architecture and infrastructure within the network. We first discuss a generic quantum network model that represents the physical layer of the network before moving to our assumptions of the link layer and network layer that operate within a quantum network stack at each node in the network. We then describe our representation of a repeater protocol that describes the series of operations performed on a repeater chain to connect two end nodes in a network. We then introduce the necessary network infrastructure and architecture needed to realize a dynamic TDMA scheme that can provide the features described in our motivation.

3.2.1. Physical Layer Model

We represent a quantum network as a topology graph $G(V, E, C_V, S_V)$ where V is the set of vertices representing the set of quantum network nodes, each edge $(u, v) \in E$ represents pairs of nodes in the networks that are able to generate elementary links between one another, C_V is a map from a node $v \in V$ to a set of communication resources $C_v = \{R^c_{v,1}, ..., R^c_{v,n}\}$ and S_V is a map from a node to a set of storage resources $S_v = \{R^s_{v,1}, ..., R^s_{v,m}\}$.

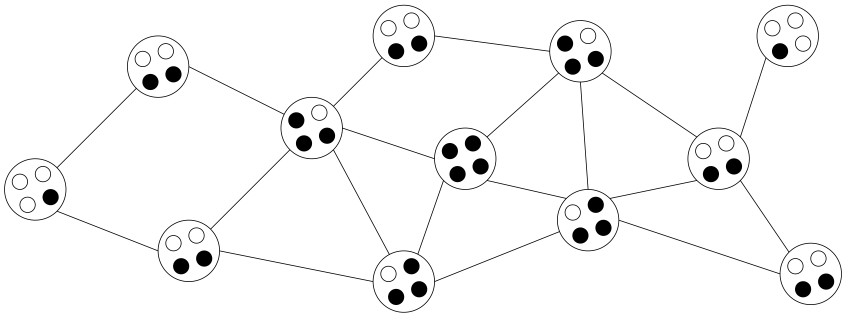


Figure 3.1: Visualization of a generic quantum network. Nodes contain communication qubits (black circles) and storage qubits (white circles).

Communication qubits are resources used for generating elementary links between two directly connected nodes whereas storage qubits are used for storing previously generated links in order to free communication qubit resources. In this work we make the following assumptions on the network resources:

- Any communication qubit contained in a node v may be used for generating an elementary link with any neighbor u if $(u, v) \in E$.
- The state stored in any communication qubit at a node v can be stored in any unused storage qubit at the same node v.
- Any pair of qubits (communication or storage) holding entangled links at a node v may be used for performing entanglement swapping or entanglement distillation.
- Any operations performed on a node's qubits may be performed in parallel. This includes generating entanglement as well as performing entanglement swapping and entanglement distillation.

3.2.2. Link Layer Model

We assume that in order for two nodes u, v to generate entanglement, both nodes must have an available communication qubit. If at some time a node u has no available communication qubits, u may not generate any elementary links in the current time slot. In addition to the connectivity information in E of the topology graph, each edge (u, v) is associated with a set of *link capabilities* $(F, R) \in \mathbb{R} \times \mathbb{R}$ that describe the fidelity/rate pairs for which an entangled link may be produced. Only positive rates are allowed and because entanglement generally ceases to be useful when $F \leq 0.5$ we assume that $0.5 < F \leq 1$.

Quantum devices like NV in diamond offer a trade-off between the rate at which an elementary link may be generated and the fidelity of the resulting link [16]. These devices may thus specify multiple link capabilities whereas links between devices like atomic ensembles [17] offer a static fidelity/rate pair determined at device fabrication and placement. The architecture specified in [18] is assumed to provide desired link layer functionality with a modification to the scheduler which now references a TDMA schedule for determining what requests to process at any moment. We assume that elementary link generation is probabilistic and that rates correspond to the average amount of time needed to a single elementary link.

l/F	0.88	0.83	0.79	0.75	0.7	0.66	0.62	0.57
5 km	14.16	20.84	27.83	33.98	39.18	45.6	51.26	57.73
10 km	7.79	11.07	14.83	18.4	22.62	24.47	29.21	31.07
15 km	5.33	7.94	10.1	12.29	14.49	16.91	20.39	22.04
20 km	3.92	6.02	7.39	9.45	11.26	12.97	14.63	17.0
25 km	3.23	4.43	6.53	7.6	9.71	10.7	12.27	13.32
30 km	2.78	4.02	5.11	6.04	7.27	8.37	10.37	10.64
35 km	2.19	3.24	4.34	5.43	6.54	7.25	8.18	10.01
40 km	2.03	3.14	3.91	4.74	5.3	6.82	7.63	8.91
45 km	1.77	2.59	3.53	4.22	5.11	5.99	6.69	7.35
50 km	1.66	2.45	3.17	3.97	4.48	5.12	5.72	6.67

Table 3.1: Entanglement generation rates (Hz) for various link lengths l and fidelity F acquired from simulations of NV centers in diamond using parameters in [18].

Throughout the remainder of this thesis, we assume that the underlying quantum hardware is built on nitrogen-vacancy (NV) centres in diamond. Using the simulation model and results of [18] we characterize the link capabilities by varying the bright state population used in the entanglement generation process. Table 3.1 shows our assumed link capabilities for various link lengths in the network.

3.2.3. Network Layer Model

The network layer is responsible for managing the execution of repeater protocols of multi-hop repeater chains. Thus, we assume that:

- The network layer handles the exchange of control messages over the repeater chain and executing entanglement swapping and distillation.
- The network layer is responsible for declaring failure in the case when any step of the scheduled protocol fails. This involves notifying the nodes along the path so that they may clean up any state and release their resources for subsequent protocols.

Specifically, control messages that contain measurement outcomes from the distillation and entanglement swap processes must be propagated to nodes along the repeater chain. In the case of successful operation, entanglement swapping may additionally involve performing some correction operations on the entangled qubit held by a device in the network. When applications at the end nodes do not demand more qubits, the network layer is responsible for preventing nodes internal to the repeater chain from continuing execution of the protocol. Finally, elementary link generation generation, distillation, and entanglement swapping may be probabilistic processes that result in protocol failure. These assumptions ensure that repeater protocols are adhere to their QoS demands and that the network layer does not provide entanglement to applications when they have failed along the repeater chain.

3.2.4. Repeater Protocol Model

We model a repeater protocol as a directed acyclic graph $P(A, I)$ where A is the set of vertices corresponding to actions in the protocol and I is the set of edges describing the dependency relations between actions of the protocol. Each action $a \in A$ is represented as a tuple (a_{ID}, a_V, a_F, a_R).

$a_{ID} \in \{link, swap, distill\}$ is an identifier of the type of action and denotes whether elementary link generation, entanglement swapping, or entanglement distillation should be performed.

$a_V \subset V$ specifies the network nodes involved in the action. For $a_{ID} = link$ and $a_{ID} = distill$, a_V specifies the pair of nodes that generate an elementary link with one another or perform entanglement distillation respectively. For $a_{ID} = \mathit{swap}$, a_V specifies the single node that performs entanglement swapping. Since repeater protocols may require nodes to perform these actions several times, a node may be involved in few or many actions of the protocol. Note that no qubit resources have been assigned to any action at this level of protocol description.

$0.5 < a_F \leq 1$ specifies the fidelity of the link upon performing the action while $a_R \in \mathbb{R}$ denotes the rate at which the action is performed. Specifically, for $a_{ID} = \mathit{link}$ the value of a_R specifies the rate at which a single link should be produced while for $a_{ID} \in \{\mathit{swap}, \mathit{distill}\}$ the value of a_R denotes the rate of performing an entanglement swap or distillation. In this thesis, the rates of an entanglement swap and distillation are the inverse of the corresponding operation's latency.

In this thesis, the set of possible actions encoded in a vertex a includes generating an elementary link between two nodes $u, v \in V$, performing two-to-one entanglement distillation at two nodes $u, v \in V$, or performing entanglement swapping at a node $v \in V$. The set of edges I represents precedence constraints between actions of the protocol. Elementary link generation does not depend on previous actions and thus the entire set of sources within a protocol P are comprised solely of elementary link generations. Distillation consumes two links that exist between the same nodes u, v and produces a single, higher quality link between nodes u and v. Entanglement swaps consume two entangled links shared between nodes u, v and v, w that share a common node v and produce a link between u and w.

Example 3.2.1. Figures 3.2a and 3.2b show a six-node network and an example of a protocol over the three-node repeater chain (A, B, C). The source actions in the protocol are link generations that occur between the pairs of nodes (A, B) and (B, C) with a fidelity F of 0.88 at a rate R of 14.16 entangled links per second. These links are then distilled to create higher fidelity links between (A, B) and (B, C). Once distillation has completed, B performs an entanglement swap using the two distilled links to complete the protocol and provide A and C with an end-to-end entangled link.

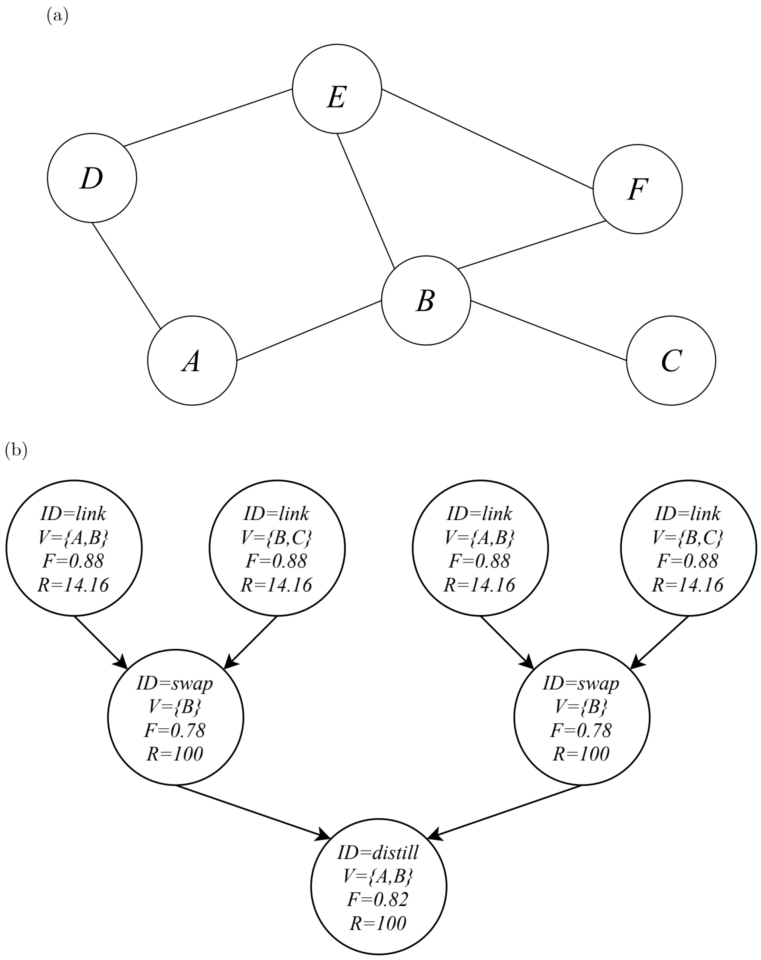

(a)
E
D
F
B
A
C
(b)
ID=link V={A,B} F=0.88 R=14.16
ID=link V={B,C} F=0.88 R=14.16
ID=link V={A,B} F=0.88 R=14.16
ID=link V={B,C} F=0.88 R=14.16
ID=swap V={B} F=0.78 R=100
ID=swap V={B} F=0.78 R=100
ID=distill V={A,B} F=0.82 R=100

Figure 3.2: a) Example network composed of six nodes. Edges show pairs of nodes that can perform heralded entanglement generation. b) Example of a repeater protocol over the three-node repeater chain (A, B, C). Here, all elementary links are generated with fidelity 0.88 and rate 14.16 $\frac{ebit}{s}$. These are then used by entanglement swaps to produce a link with fidelity 0.82. A and B then perform an entanglement distillation producing an entangled link with fidelity 0.82.

While repeater protocols may specify a way to generate an end-to-end entangled link, they do not specify what qubits should be used at each node along a repeater chain for performing the protocol actions. In the case when resources are limited, some protocols may require that elementary links are generated serially as there are not enough communication qubits to support concurrent generation. Some repeater protocols may not even be possible to execute on a limited number of resources as they may require many links to be stored simultaneously to perform the repeater protocol. We thus introduce the notion of a *concrete* repeater protocol.

Definition 3.2.1. A repeater protocol is *concrete* if it is accompanied with a relative offset mapping $M : A \to \mathbb{R}$ which specifies the activation time for each action $a \in A$ and a resource mapping Q which specifies the set of qubits used. Specifically, the activation time $M(a)$ of an action is relative to the start of the protocol. Q maps an action $a = (a_{ID}, a_V, a_F, a_R) \in A$ to a subset of qubit resources $\cup_{v \in a_V} C_V(v) \cup S_V(v)$ held by the set of nodes a_V responsible for performing the action. Each node u is thus involved in any action a if $C_u \cap Q(a) \neq \emptyset$ or $S_u \cap Q(a) \neq \emptyset$.

Due to the dependency of entanglement swapping and entanglement distillation on previously generated entanglemnet, we treat each communication qubit and storage qubit in the network as a unique resource. This simplifies the mapping of qubits to actions as nodes will not need to dynamically decide which qubits should be used for each action of the protocol. We denote a concrete repeater protocol as $P(A, I, M, Q)$.

3.2.5. Central Controller Model

In order to construct repeater protocol TDMA schedules for the network we introduce a central controller. The responsibility of the centralized controller is to perform admission control of new demands and to perform routing and construction of concrete repeater protocols that adhere to the admitted QoS demands. We make the following assumptions on the network using such a central controller:

- Quantum network nodes have a method of communicating their demands to a centralized controller. This can be achieved using a local reservation manager that tracks local demands and communicates with the central controller over a classical network such as the Internet.

- The controller is aware of the link capabilities between all pairs of connected nodes as well as the available network resources (communication and storage qubits) at each node.
- All nodes are synchronized to slot boundaries. This may be achieved by using GPS clocks or by having the central control perform IEEE 1588 Precision Time Protocol [19] as seen in time-sensitive networking (TSN) networks [20].

First, applications at end nodes specify connection details and QoS requirements. Demands are represented by a tuple $(source, dest, F_{min}, R_{min})$ that specify the source and destination end nodes as well as the minimum fidelity and rate requirement. When an application communicates these demands to the local quantum network stack, an entanglement manager is responsible for assessing whether a connection exists that supports the demands. In the case when no connection is set up, the entanglement manager consults a reservation manager that contacts the central controller in order to set up a connection in the network.

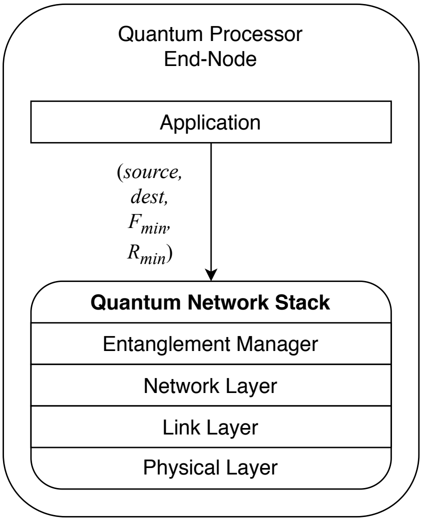


Figure 3.3: Application at quantum processor end node setting up connection. When an application desires entanglement it first provides connection details to the local network stack. If the TDMA schedule is installed successfully the application may subsequently request entanglement from the network stack (not shown).

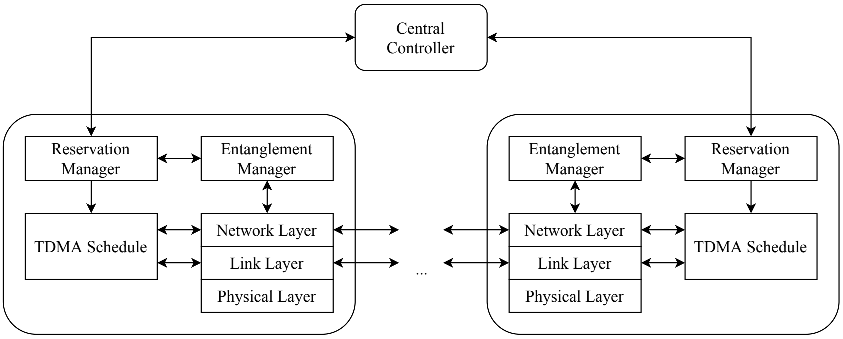


Figure 3.4: Network node architecture for supporting dynamic TDMA repeater protocol scheduling. End nodes have a quantum network stack (entanglement manager, network layer, link layer, physical layer) that is used for executing repeater protocols and generating entanglement. An entanglement manager tracks connections in the network and delivers entanglement information to local applications. A reservation manager is responsible for negotiating with the central controller for establishing new network connections.

Once a set of network demands have been collected, the controller will utilize network information to perform routing so as to select the set of repeater chains for each demand. The demands and their assigned repeater chains are then fed into a protocol selection step where concrete repeater protocols are computed for the repeater chains such that they meet the QoS requirements of the demands. The set of concrete protocols and demands are then used to construct a TDMA schedule that is distributed to the reservation managers across the nodes in the network.

The TDMA schedule returned to a node contains the slot information for its local network resources that are used for repeater protocols in the network. Each network resource has its own set of slot information for the schedule and contains:

- *Protocol Action* - Specifies the action of the protocol to be performed $(link, swap, distill)$ along with the fidelity and rate details of the action.

- *Circuit ID* - An identifier of the end-to-end repeater protocol that the network resource is being used for.

Upon receiving the TDMA schedule from the central controller, the reservation manager installs the schedule so that the network layer and link layer have access to the provided information. Specifically, the net-

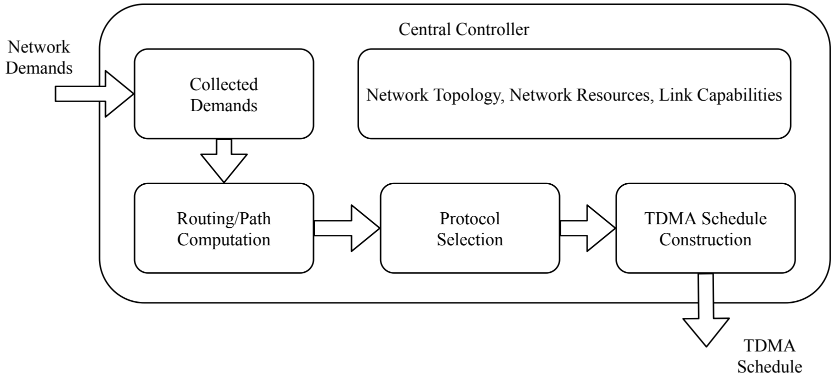


Figure 3.5: Flow diagram of central controller operations. Network demands are received from reservation managers at end nodes and are processed jointly to compute routes and concrete repeater protocols that allow the resources in the network to be shared. Concrete repeater protocols are then fed into a scheduling step to construct a TDMA schedule for the network.

work layer uses the action information for executing entanglement swapping and distillation on the local resources and the circuit ID information for communicating the results of the actions to the correct nodes that form the repeater chain. The link layer uses the action information for generating elementary links that adhere to the QoS requirements of the repeater protocol.

While the TDMA schedule provides coordination for the repeater protocol, it is not necessarily the case that the repeater protocols are executed as soon as the schedule is received. Initiation of the repeater protocol is handled by the network layer so that entanglement is delivered to the end nodes in the network only when applications need it. Upon successful installation of the TDMA schedule, the application may proceed to request entanglement from the network stack. The mechanism by which the network stack provides information about the generated entanglement to the application remains out of the scope of this thesis. Here, we focus solely on the mechanism for coordinating entanglement generation which assists the network layer and link layer in generating application-requested entanglement.

Centralizing network information and coordination has the advantage that routing, protocol construction, and scheduling may be done jointly in order to distribute demands over the resources in the network and prevent congestion. Since the goal of this thesis is to investigate the construction of TDMA schedules, we model the routing and protocol computation as independent processes that then provide a set of concrete repeater protocols into the scheduling step.

3.3. TDMA Scheduling Problems

In this section we formally pose the TDMA scheduling problems for non-preemptive repeater protocols (NPRP) as well as limited preemption repeater protocols (LPRP). Given a set of demands and their corresponding concrete repeater protocols, the NPRP problem attempts to construct a conflict-free TDMA schedule that satisfies the set of demands. In future NISQ devices it may be possible to tolerate some delay introduced in between actions of a repeater protocol, thus the LPRP attempts to find a conflict-free schedule that satisfies a set of demands that additionally specify an upper bound on the amount of delay that may be introduced to the relative activation times of actions in the concrete protocol.

3.3.1. NPRP TDMA Scheduling

We represent a TDMA network schedule with a map $\mathcal{S} : \mathcal{P} \to \mathbf{s}$, where $\mathcal{P}$ is the set of concrete protocols and $\mathbf{s}$ is a set of activation times corresponding to which slots a concrete protocol begins execution. For a concrete protocol $P(A, I, M, Q)$ the set of absolute activation times of an action $a \in A$ are $t_a = \{s + \lceil \frac{M(a)}{t_{slot}} \rceil, \forall s \in \mathcal{S}(P)\}$ where s is an activation time of protocol P and t_{slot} is the duration of a slot in seconds. Thus from a TDMA schedule and the set of concrete protocols, nodes may determine the activation times of all actions in the protocol along with the network resources used. The problem of NPRP TDMA scheduling may be formulated as follows,

Definition 3.3.1. Given a quantum network $G = (V, E, C_V, S_V)$, a set of demands D and their concrete protocols $\mathcal{P}$, construct a schedule $\mathcal{S}$ such that the demands $\mathcal{D}$ are satisfied.

Because the concrete protocols provided to the scheduling step are constructed in order to satisfy the fidelity demanded, it is the scheduler's responsibility to ensure that the constructed TDMA schedule executes the protocols frequently enough to satisfy their demanded rates. In chapter 4 we shall describe how the problem of constructing the TDMA schedule may be formulated as both a periodic task scheduling problem as well as an instance of the resource-constrained project scheduling problem (RCPSP) with timing constraints.

3.3.2. LPRP TDMA Scheduling

As previously stated, the LPRP TDMA scheduling problem is a modification of the NPRP with the additional flexibility of delaying actions in the concrete protocol. The purpose of investigating this problem is to see how improvements in qubit storage times can improve the schedulability of repeater protocols in quantum networks. By delaying actions in a protocol, it may be possible to accommodate additional demands in the network by reducing resource contention. Here, we represent the amount of delay a concrete protocol P may experience with a value $B_{delay} \in \mathbb{R}$ referred to as a *delay bound*. The problem of LPRP TDMA scheduling may be formulated as follows,

Definition 3.3.2. Given a quantum network $G = (V, E, C_V, S_V)$, a set of demands $\mathcal{D}$, their concrete protocols $\mathcal{P}$, and a delay bound mapping $\mathcal{B} : \mathcal{P} \to \mathbb{R}$ construct a schedule $\mathcal{S}$ such that the demands D are satisfied.

Similarly to the NPRP, chapter 4 will present a formulation of the LPRP in the form of a limited-preemption periodic task scheduling as well as a modified RCPSP that changes the maximal time lags of activities to limite the amount of delay that can be introduced into a repeater protocol.

References

[1] H. Bernien, B. Hensen, W. Pfaff, G. Koolstra, M. S. Blok, L. Robledo, T. Taminiau, M. Markham, D. J. Twitchen, L. Childress, *et al.*, *Heralded entanglement between solid-state qubits separated by three metres,* Nature **497**, 86 (2013).

[2] N. Kalb, P. C. Humphreys, J. Slim, and R. Hanson, *Dephasing mechanisms of diamond-based nuclear-spin memories for quantum networks,* Physical Review A **97**, 062330 (2018).

[3] A. Reiserer, N. Kalb, M. S. Blok, K. J. van Bemmelen, T. H. Taminiau, R. Hanson, D. J. Twitchen, and M. Markham, *Robust quantum-network memory using decoherence-protected subspaces of nuclear spins,* Physical Review X **6**, 021040 (2016).

[4] C. Bradley, J. Randall, M. Abobeih, R. Berrevoets, M. Degen, M. Bakker, M. Markham, D. Twitchen, and T. Taminiau, *A ten-qubit solid-state spin register with quantum memory up to one minute,* Physical Review X **9**, 031045 (2019).

[5] C. H. Bennett and G. Brassard, *Proceedings of the ieee international conference on computers, systems and signal processing,* (1984).

[6] A. K. Ekert, *Quantum cryptography based on bell's theorem,* Physical review letters **67**, 661 (1991).

[7] A. Broadbent, J. Fitzsimons, and E. Kashefi, *Universal blind quantum computation,* in *2009 50th Annual IEEE Symposium on Foundations of Computer Science* (IEEE, 2009) pp. 517–526.

[8] J. F. Fitzsimons and E. Kashefi, *Unconditionally verifiable blind quantum computation,* Physical Review A **96**, 012303 (2017).

[9] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, *Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels,* Physical review letters **70**, 1895 (1993).

[10] M. Zukowski, A. Zeilinger, M. A. Horne, and A. K. Ekert, *" event-ready-detectors" bell experiment via entanglement swapping.* Physical Review Letters **71** (1993).

[11] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, *Purification of noisy entanglement and faithful teleportation via noisy channels,* Physical review letters **76**, 722 (1996).

[12] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, *Quantum privacy amplification and the security of quantum cryptography over noisy channels,* Physical review letters **77**, 2818 (1996).

[13] J. Dehaene, M. Van den Nest, B. De Moor, and F. Verstraete, *Local permutations of products of bell states and entanglement distillation,* Physical Review A **67**, 022310 (2003).

[14] S. M. Lasassmeh and J. M. Conrad, *Time synchronization in wireless sensor networks: A survey,* in *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)* (IEEE, 2010) pp. 242–245.

[15] P. P. Nuspl, K. E. Brown, W. Steenaart, and B. Ghicopoulos, *Synchronization methods for tdma,* Proceedings of the IEEE **65**, 434 (1977).

[16] P. C. Humphreys, N. Kalb, J. P. Morits, R. N. Schouten, R. F. Vermeulen, D. J. Twitchen, M. Markham, and R. Hanson, *Deterministic delivery of remote entanglement on a quantum network,* Nature **558**, 268 (2018).

[17] L.-M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller, *Long-distance quantum communication with atomic ensembles and linear optics,* Nature **414**, 413 (2001).

[18] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, *et al.*, *A link layer protocol for quantum networks,* in *Proceedings of the ACM Special Interest Group on Data Communication* (2019) pp. 159–173.

[19] K. Lee, J. C. Eidson, H. Weibel, and D. Mohl, *Ieee 1588-standard for a precision clock synchronization protocol for networked measurement and control systems,* in *Conference on IEEE*, Vol. 1588 (2005) p. 2.

[20] K. B. Stanton, *Distributing deterministic, accurate time for tightly coordinated network and software applications: Ieee 802.1 as, the tsn profile of ptp,* IEEE Communications Standards Magazine **2**, 34 (2018).

4 Constructing TDMA Schedules of Repeater Protocols

This chapter presents our contribution of novel methods to construct time-division multiple access (TDMA) schedules for coordinating end-to-end entanglement delivery in quantum networks. We will provide an in-depth description of the methodology taken for evaluating the use of different scheduling heuristics in constructing TDMA schedules of repeater protocols in quantum networks.

Section 4.1 will describe our procedure for generating concrete repeater protocols that satisfy rate and fidelity demands between quantum processing end nodes in the network. We first show how non-concrete repeater protocols can be constructed for a repeater chain that connect a source and destination pair. We then show how these repeater protocols can be mapped to qubit resources in order to make them concrete.

Section 4.2 describes our periodic task scheduling formulation of TDMA schedule construction. We describe how periodic task sets that represent the set of network demands and their corresponding concrete repeater protocols may be constructed for both the non-preemptive repeater protocol (NPRP) and limited-preemption repeater protocol (LPRP) TDMA scheduling problems. We then show how a valid schedule of these periodic tasks can be used to construct the appropriate TDMA schedule for the network.

Our periodic task scheduling formulation includes a novel task scheduling problem with timing constraints named the limited preemption budget scheduling problem as well as a heuristic for constructing a schedule. This scheduling problem has applications beyond quantum networks in real-time systems where data freshness is required for system behavior, and may be of independent interest.

Section 4.3 shows our resource-constrained project scheduling problem (RCPSP) formulation of TDMA schedule construction. This section describes how the set of network demands and their corresponding concrete repeater protocols can be used to construct an activity-on-node network for the NPRP and LPRP TDMA scheduling problems. We then present how the scheduled project can be turned into the desired TDMA schedule and additionally show how one may transform the activity-on-node network in order to reduce computational complexity of the scheduling step.

4.1. Concrete Protocol Generation

In order to evaluate the performance of scheduling algorithms for constructing TDMA schedules in quantum networks it is necessary to have a set of tasks that are representative of multi-hop repeater protocols. We begin by describing how we generate a concrete repeater protocol given a network demand and a description of the network resources and link capabilities. In chapter 5 we will show how various network parameters influence the achievable fidelity and rate of the repeater protocols produced using the methods outlined here.

4.1.1. Repeater Protocol Generation

Repeater protocols are composed of actions that include elementary link generation, entanglement distillation, and entanglement swapping. The resulting fidelity and rate of a protocol depends on a number of device and network parameters. This includes the rate and fidelity at which the elementary links are generated, the number of communication qubits and storage qubits a node has, as well as the quality of the devices in the network. In contrast to classical networks, these protocols do not need to be performed in a cascading fashion from the source to the destination nodes. In principle, elementary links may be generated in any order along a repeater chain. Similarly, the entanglement swaps and entanglement distillations may be performed arbitrarily as long as the needed entangled links are available as input.

As the focus of this thesis is on scheduling repeater protocols onto network resources, we limit the scope of repeater protocol generation to depend on the number of resources available at each node in the network as well as the link capabilities. In practice, selecting a protocol and the resources used for its actions may depend on the quality and control of individual qubits. Some devices may have communication qubits and storage qubits that observe limited interactions with one another which places restrictions on the choice of resources that are used for entanglement swapping or entanglement distillation. We assume that entanglement distillation and swapping can be performed in a deterministic manner with no additional gate noise reducing the fidelity of the entangled link. These assumptions result in protocols that have optimistic performance in terms of fidelity but provide a sufficient depiction of how resources are allocated in order to execute a repeater protocol.

Searching For Repeater Protocols

We generate repeater protocols using a scheme built on the entanglement swapping scheme search (ESSS) presented in [1] that makes use of a constant fidelity entanglement flow heuristic. ESSS can be understood in the following way. Given a repeater chain of nodes beginning at a source node s and ending at a destination node d along with a desired minimum fidelity F_{min} and rate R_{min}, we select a "pivot" node p internal to the repeater chain and recursively find protocols on the two repeater chains induced by the pivot choice. The repeater protocols found on these smaller chains are then turned into one for the full chain by using entanglement swapping at the pivot node.

In order to ensure that the discovered repeater protocol satisfies the rate demanded, we require that both of the induced repeater chains satisfy the originally demanded rate R_{min}. If the link to the left of the pivot is generated at a rate R_L and the link to the right is generated at a rate R_R then a valid protocol must satisfy $R_{min} \leq \min(R_L, R_R)$ in order to satisfy the demand as the rate of the full protocol is limited by the lowest rate of its components. To satisfy the demanded fidelity F_{min}, a new fidelity F'_{min} is chosen for both of the induced repeater chain such that performing an entanglement swap with two entangled links of fidelity F'_{min} is at least F_{min}. This choice of equal fidelity on both of the induced repeater chains gives rise to the name of the constant fidelity entanglement flow heuristic.

The process of choosing a pivot on the induced repeater chains is performed recursively until the chain in question consists of only two directly connected nodes. At this level, the repeater protocol contains only elementary link generations and entanglement distillations as no node lies between two directly connected nodes to perform an entanglement swap.

Example 4.1.1. We now present an example to help the reader visualize the ESSS algorithm from [1]. Suppose we have a path of five nodes as shown at the top of figure 4.1 where the end nodes S, D are the source and destination and the internal nodes are repeaters. We first choose R_2 as a pivot and break the repeater chain into two smaller chains, one from S to R_2 and one from R_2 to D. For each of these chains we choose another pivot, here R_1 and R_3, and recursively break the chain in two again. At the bottom level there are only elementary link generations.

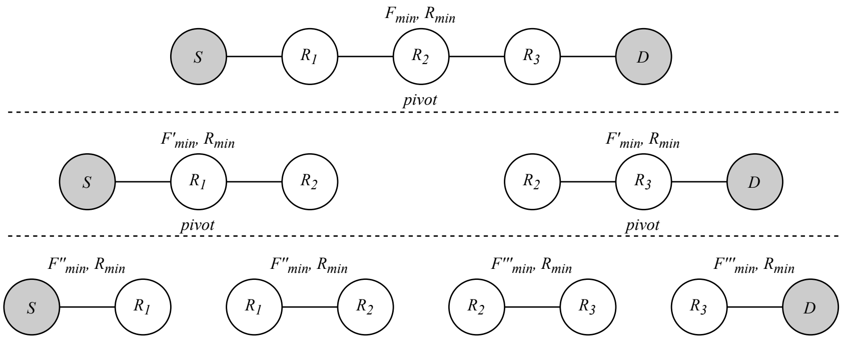


Figure 4.1: Visual depiction of the operation of ESSS. The algorithm begins with the full repeater chain (top). A pivot is then chosen and the chain is split in two (middle). New pivots are chosen at each level until elementary links remain (bottom).

Should it be found that the achieved rates of the repeater protocols on either side of the pivot differ, we can track the temporary achieved rate $R_{tmp} = \min(R_L, R_R)$ and select a new pivot to attempt to increase the minimum achieved rate and choose a protocol that maximizes the end-to-end rate on the chain. In the previous example, after choosing R_2 as the initial pivot we may find that the repeater chain (R_2, R_3, D) has a higher rate than (S, R_1, R_2), thus we can try to find a new protocol by choosing R_3 as the pivot to try and balance the rates on either side of the pivots.

Evaluating Repeater Protocols

While ESSS provides a mechanism for searching for repeater protocols, it does not provide a method for mapping the protocol to network resources nor evaluating its achieved fidelity and rate. To generate protocols for a given network we extend the search algorithm with a more concrete mechanism for considering the available communication and storage resources at a node as well as a simple heuristic for tracking the effects of distillation and swapping. While computing the state of an elementary link may be simple, tracking the quantum states upon subsequent entnaglement swaps and entanglement distillations can become computationally difficult.

To simplify tracking protocol fidelity and selecting fidelity requirements, we assume that the states in the network are of the Werner form [2]

$$\rho_{werner} = \frac{1-F}{3}\mathbb{1}_2 + \frac{4F-1}{3}|\Phi^+\rangle\langle\Phi^+| \tag{4.1}$$

where F is the fidelity of the state, $\mathbb{1}_2$ is the maximally mixed state for two qubits, and $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Using these states we may use simple numerical formulas to evaluate the fidelity of a link resulting from an entanglement distillation step or an entanglement swapping step. From [3], the resulting fidelity F_D of two-to-one distillation of two states with fidelity F_1 and F_2 can be calculated as

$$F_D = \frac{F_1F_2 + \frac{(1-F_1)}{3}\frac{(1-F_2)}{3}}{F_1F_2 + F_1\frac{(1-F_2)}{3} + F_2\frac{(1-F_1)}{3} + 5\frac{(1-F_1)}{3}\frac{(1-F_2)}{3}} \tag{4.2}$$

while the resulting fidelity F_S of swapping two Werner states with fidelities F_1, F_2 can be calculated as

$$F_S = F_1F_2 + \frac{(1-F_1)(1-F_2)}{3} \tag{4.3}$$

When selecting the minimum fidelity F'_{min} of repeater chains induced by a pivot, we use equation 4.3 with $F_S = F_{min}$ and $F_1 = F_2 = F'_{min}$ to solve for F'_{min}.

Choice of Entanglement Distillation To perform entanglement distillation, our search uses *nested entanglement pumping* at the elementary link level and *entanglement pumping* [4–6] for repeater chains of length greater than two. This choice was made due to the trade-off between rate and resource requirements in these methods. Standard entanglement pumping has minimal resource requirements but may require more entangled links to be generated to reach a minimum fidelity F_{min}. In fact, this distillation method reaches an asymptotic limit depending on the initial link fidelity. Nested entanglement pumping on the other hand has a moderate increase in resource requirements that scales logarithmically with the number of entangled links used but can achieve higher fidelities than the non-nested version.

As mentioned previously, the achievable rate of a repeater protocol is limited by the rate of protocols on the left and right side of its pivot. In the base case of a single link, the repeater protocol is limited by the rate achievable by an elementary link. Thus we wish to use the node resources to perform nested entanglement distillation at the elementary link level to maintain high rates. In order to ensure elementary links have sufficient resources to perform nested entanglement distillation we use standard pumping for multi-hop repeater chains as they have lower resource requirements.

In order to properly evaluate the rate of a repeater protocol we need to consider how the protocol is mapped to the available resources in the network. When splitting a repeater chain in two using a pivot, the repeater protocols on each induced chain are constructed using all network resources along the chain. The problem with this is that the pivot node must share its resources among the two repeater protocols, which may reduce the rate of the end-to-end protocol. In the next section, we will describe how we can take a repeater protocol and map it to network resources to accurately characterize the latency and achievable rate of the protocol.

Protocols generated using this scheme can be represented by a directed acyclic graph where the nodes represent actions of the protocols and edges represent dependencies between actions. Due to the structure of the ESSS algorithm, these protocols will result in a tree-like structure following the model described in chapter 3. Figure 4.2 shows a protocol that may be generated on a three-node chain. The protocols generated from this mechanism only specify which nodes are involved in each action and do not detail which hardware resources are used for each step. We next detail how the repeater protocols are made into concrete protocols and mapped to physical resources held at each node.

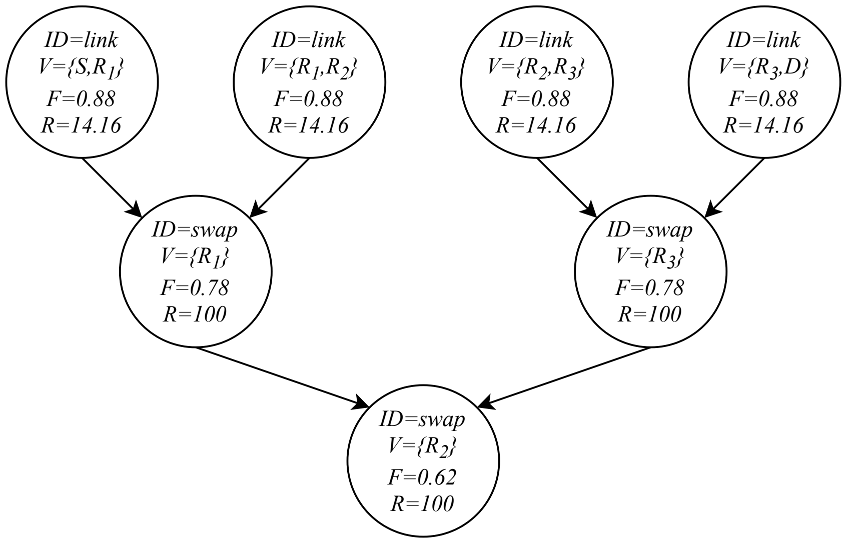


Figure 4.2: An example repeater protocol generated using the ESSS search on the repeater chain in figure 4.1 when $F_{min} = 0.6$. A single elementary link is generated between each consecutive pair of nodes and the internal repeaters perform entanglement swapping.

4.1.2. Mapping Construction

As mentioned in chapter 2, experimental realizations of qubits are susceptible to noise and decoherence which reduces the fidelity of a stored quantum state. In the context of repeater protocols, entangled links decohere as they are stored before being used for entanglement swapping or entanglement distillation. As a consequence, the construction of a concrete repeater protocol and its mapping to physical resources has implications on the resulting end-to-end rate and fidelity of the delivered entanglement. Here, we will explain how we obtain the resource mapping Q and the relative offset mapping M for a repeater protocol $P(A, I)$.

Once a protocol has been generated during the ESSS search it is necessary to map the protocol to the resources held at each node in the repeater chain. Since the protocols are represented as a tree of actions, we map the actions to node resources by performing a post-order traversal of the protocol tree. When a leaf node (elementary link generation) is reached, we map the link generation to communication and storage qubits held at each of the two nodes and propagate these assignments to the distillation and swap steps built on them. The post-order traversal is performed with a preference to subtrees that have higher depth. This choice ensures that entangled links that have a higher generation latency are performed earlier in the repeater protocol so as to minimize introduced decoherence on stored links.

Mapping Elementary Links When deciding which resources to use for an elementary link generation, we greedily reserve storage resources to store the resulting link. This frees communication qubit resources for subsequent elementary link generations. While this may introduce inefficiencies if entangled links do not need to be stored, it reduces any additional complexity of backtracking through the resource mapping to use storage qubits when all communication qubits are storing links. Our protocol generation treats all communication qubits and storage qubits as equal so no intelligent decisions must be made on the selection of resources.

Once a set of resources have been selected for an action a performing elementary link generation, we can construct an entry in the resource mapping Q for the concrete repeater protocol that reflects this choice of resources. Recall from chapter 3 that an action a is associated with a rate a_R. To ensure the qubits are reserved for the expected duration of the action, they remain occupied for $\frac{1}{a_R t_{slot}}$s where t_{slot} is the duration of a slot in the TDMA schedule. Furthermore, if a single communication qubit is exists at a node, it is not possible to generate additional links until the communication qubit is freed by an entanglement swap or an entanglement distillation that consumes it. By assigning qubits to elementary link generations, propagating assignments to entanglement swaps and distillations, and tracking the occupation time of each resource for each action, we may determine the resource mapping $Q(a)$ for the action as the set of selected resources and the time mapping $M(a)$ as the point in time when the resources are available.

Mapping Entanglement Swapping and Distillation Once the leaf nodes of a subtree have been mapped to communication qubits and storage qubits, the mapping is propagated to entanglement distillations and entanglement swaps that build upon the elementary links. Entanglement swaps and entanglement distillations have two preceding actions b and c that are consumed in the action. If an action a is an entanglement swap or distillation then the resources in $Q(a)$ are those held at the nodes in a_V upon completion of the previous actions b and c. Discovering these resources involves either tracking the set of qubits holding the link at each vertex of the protocol tree or backtracking through preceding actions to find the resources held at the nodes in a_V.

Selecting resources for entanglement swaps and entanglement distillations in this fashion ensures that the correct resources storing entangled links are used for these operations. Entanglement swapping operations consume and free both resources held locally at a node while entanglement distillations store the resulting link in one of the qubits of the local node. The resource where a distilled link is stored is further propagated along the repeater protocol tree to the remaining operations until we have reached the root and completely mapped the protocol.

Determining $M(a)$ for an entanglement swap or an entanglement distillation is simply the earliest moment where both of the preceding actions have completed.

Mapping Summary To summarize the flow of the mapping construction, the protocol is first scheduled onto the node resources in an as-soon-as-possible (ASAP) fashion to find the latency of the protocol. The protocol generation mechanism may cause entanglement swap actions to be dependent on other entanglement swaps and distillations or vice versa. Often times, the resulting swap or distillation action does not have a direct data dependency on the resources used and may result in a non-optimal schedule. To alleviate this, the latency obtained from the ASAP scheduling is then used to produce an as-late-as-possible (ALAP) schedule which allows performing actions part of the protocol in parallel with one another when no data dependencies are observed. One issue that results in performing an ALAP scheduling of the protocol tasks is that distillations and swaps that consume the resources produced by link generation may be pushed far into the future, thus introducing decoherence on the resources generated. The task generation step is thus completed by performing one more pass over the schedule and shifting all distillation and swap actions as early as possible to reduce the decoherence introduced.

The output of our mapping procedure produces a concrete repeater protocol that specifies when each action of the protocol is performed and which qubits at each node are used for each action. Through the remainder of this thesis, we visualize concrete repeater protocols using resource timelines that show how relevant resources are used for executing the concrete repeater protocol. We identify qubit resources at each node using an identifier (e.g. $A\text{-}C_0$) that indicates 1) the node holding the qubit (A), 2) the type of qubit (C for communication, S for storage), and 3) an enumeration of the type of resource to uniquely identify qubits of the same type. Actions are distinguished by labeling them with an identifier (e.g. L_1) that indicates the type of action (L for *link*, S for *swap*, D for *distill*) and an enumeration to distinguish actions of the same type.

Example 4.1.2. Figure 4.3 provides an example of the labeling scheme for a repeater protocol on a three-node repeater chain with nodes A, R, and B. Here, nodes A and B have one communication qubit, $A\text{-}C_0$ and $B\text{-}C_0$ respectively, and node R has a communication qubit $R\text{-}C_0$ and a storage qubit $R\text{-}S_0$. The elementary link generation actions in figure 4.3a are identified by L_1 and L_2 while the entanglement swap action is identified by S_1. For simplicity, assume that each action in the repeater protocol occupies a single time slot. The mapping visualized in figure 4.3b can thus be understood as follows.

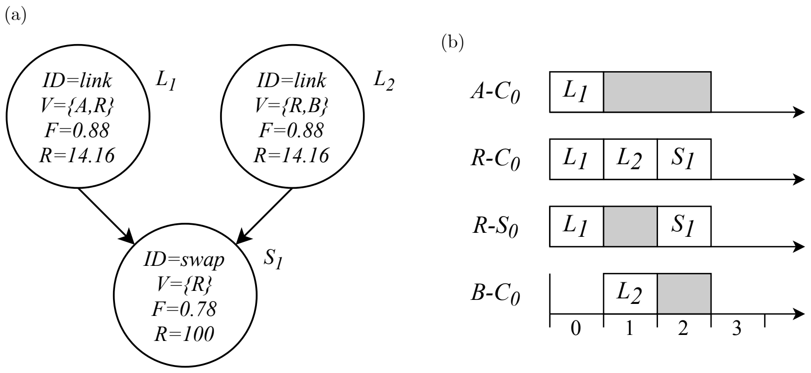

(a)
ID=link V={A,R} F=0.88 R=14.16
L1
ID=link V={R,B} F=0.88 R=14.16
L2
ID=swap V={R} F=0.78 R=100
S1
(b)
A-C0
L1
R-C0
L1 L2 S1
R-S0
L1 S1
B-C0
L2
0 1 2 3

Figure 4.3: Example of the identifier scheme and mapping construction for a repeater protocol on a three-node repeater chain. a) The non-concrete repeater protocol composed of two elementary link generations and an entanglement swap. L_1 and L_2 correspond to elementary link generations between node pairs (A, R) and (R, B) respectively while S_1 corresponds to an entanglement swap operation performed at R. b) A set of timelines for the qubits used in executing the actions in the repeater protocol. Labeled regions indicate which action the qubits are used for while shaded regions indicate that the qubit is occupied with previously generated entanglement.

First, at time slot 0 we see that qubit $A\text{-}C_0$ held by node A and qubits $R\text{-}C_0$ and $R\text{-}S_0$ held by node R are used for elementary link generation

L_1. The reason R uses both a communication qubit and a storage qubit is so that it may store the entanglement generated by R-C_0 in R-S_0 to free the communication qubit on subsequent link generations.

At time slot 1, R uses R-C_0 and B uses B-C_0 to perform elementary link generation L_2. Note that in the meantime, the previous entanglement is stored in A-C_0 and R-S_0 as shown by the shaded region.

Finally, the entanglement swap action S_1 is performed at time slot 2 using both R-C_0 and R-S_0 and the protocol is complete. Qubits A-C_0 and B-C_0 remain occupied until the end of the protocol at which point the entanglement may be used by higher level applications.

Example 4.1.3. Let us now demonstrate the resource mapping using the protocol shown in figure 4.4a on a three-node repeater chain (A, R, B) under two different sets of network resources. For simplicity, we will assume that each step of action in the repeater protocol only requires one time slot.

Suppose in the first instance that each node has one communication qubit and one storage qubit. Figure 4.4b shows a timeline depicting when each action occurs and which resources are used. Figure 4.4c shows a mapping when the repeater node has two communication qubits instead of one. We see in the latter case that the latency of the protocol is reduced as elementary links can be generated in parallel. This reduces the amount of time the links from elementary link generations L_1 and L_2 are stored, resulting in less decoherence of the links.

The mapping produced may be used to discover the latency of the repeater protocol and consequently the rate at which it can be performed. Specifically, for a given slot size t_{slot} the latency of the protocol is computed as the number of time slices n_s required to execute the protocol multiplied by the slot duration t_{slot}. The rate of the protocol may be computed as the inverse of the latency.

At the highest level of the ESSS search, we may use this mapping technique to verify a repeater protocol can indeed achieve a desired rate and to produce the concrete repeater protocol to be used for the full repeater chain. For the remainder of this thesis we will assume that the end-to-end fidelity of the protocol takes into consideration such decoherence in its reported achieved fidelity.

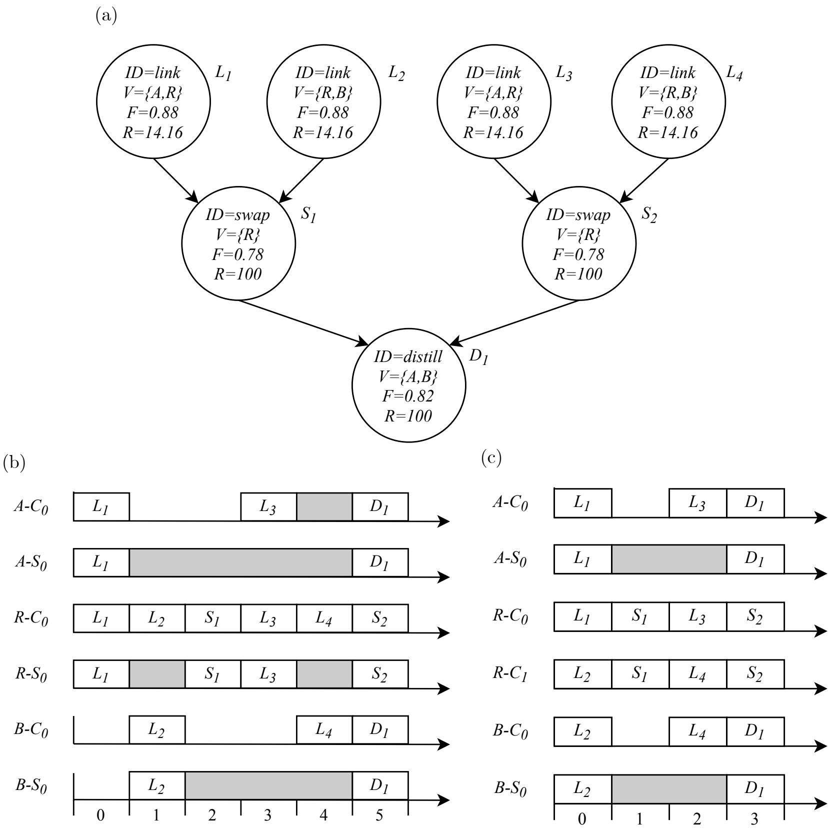

(a)
ID=link V={A,R} F=0.88 R=14.16 L_1
ID=link V={R,B} F=0.88 R=14.16 L_2
ID=link V={A,R} F=0.88 R=14.16 L_3
ID=link V={R,B} F=0.88 R=14.16 L_4
ID=swap V={R} F=0.78 R=100 S_1
ID=swap V={R} F=0.78 R=100 S_2
ID=distill V={A,B} F=0.82 R=100 D_1
(b)
A-C_0: L_1, L_3, D_1
A-S_0: L_1, D_1
R-C_0: L_1, L_2, S_1, L_3, L_4, S_2
R-S_0: L_1, S_1, L_3, S_2
B-C_0: L_2, L_4, D_1
B-S_0: L_2, D_1
0 1 2 3 4 5
(c)
A-C_0: L_1, L_3, D_1
A-S_0: L_1, D_1
R-C_0: L_1, S_1, L_3, S_2
R-C_1: L_2, S_1, L_4, S_2
B-C_0: L_2, L_4, D_1
B-S_0: L_2, D_1
0 1 2 3

Figure 4.4: Example mappings of a three-node repeater protocol. a) The non-concrete repeater protocol to be scheduled onto the three-node repeater chain (A, R, B). b) Mapping of the repeater protocol when each node has one communication qubit (C_0) and one storage qubit (S_0). c) Mapping of the repeater protocol when R has two communication qubits (C_0 and C_1) while A and B have one communication qubit (C_0) and one storage qubit (S_0).

4.2. Periodic Task Scheduling Formulation

With a mechanism for generating input to the TDMA scheduling problems, we may now consider our first formulation for constructing a TDMA schedule of repeater protocols to satisfy network demands. We begin with the simpler formulation of framing the TDMA scheduling problem as two periodic task scheduling problems. The first, the non-preemptive case, assumes that the timing of actions in the concrete repeater protocols must be strictly adhered to in order to satisfy the end-to-end fidelity requirement due the characteristics of quantum devices and networks discussed in chapters 2 and 3. The second, the limited preemption budget case, is a near-term approach where decoherence times of quantum devices have improved so that delays within protocol actions can be tolerated while still satisfying the end-to-end fidelity requirement.

This section will first describe how we can construct a set of periodic tasks that represent the concrete repeater protocols to be executed within the TDMA schedule. We will then describe our approach for constructing the TDMA schedule using non-preemptive scheduling techniques. Afterwards, we introduce our contribution of the limited preemption budget task scheduling problem and its periodic task formulation. We conclude with a description of how we construct the TDMA schedules with this approach using a heuristic for solving the limited preemption budget task scheduling problem.

4.2.1. Taskset Construction

Recall from chapter 2 that the non-preemptive periodic task scheduling problem takes as input a set of periodic tasks T and produces a schedule $\mathcal{S}$ that assigns a set of start times to each task $\tau_i \in \mathrm{T}$ that adheres to the task constraints. In order to use periodic task scheduling methods for producing a TDMA schedule of repeater protocols, a set of periodic tasks representing the concrete repeater protocols must be constructed. First, we will provide the task notation used and then we will describe how the input set of demands $\mathcal{D}$ and their concrete repeater protocols $\mathcal{P}$ can be converted to a periodic taskset.

We denote a set of tasks as T. Each task $\tau_i \in \mathrm{T}$ specifies an offset Φ_i, execution time C_i, and period T_i. We represent an instance of τ_i as $\tau_{i,j}$ and each instance $\tau_{i,j}$ has a release time $r_{i,j} = \Phi_i + T_i(j-1)$ and deadline $\delta_{i,j} = \Phi_i + T_i j$. We denote the start time of a task instance $\tau_{i,j}$ as $\sigma_{i,j}$ and the finish time as $f_{i,j}$. In the case of non-preemptive scheduling, $f_{i,j} = \sigma_{i,j} + C_i$. All quantities are assumed to be integer as TDMA schedules divide time into integer-numbered slots.

We now construct a taskset T for a set of demands $\mathcal{D}$ and their concrete protocols $\mathcal{P}$. Each demand $D_i \in \mathcal{D}$ is represented as a tuple (s_i, d_i, F_i, R_i) where s_i is the source, d_i is the destination, F_i is the minimum end-to-end fidelity, and R_i is the minimum end-to-end rate. For TDMA scheduling, we assume that time is partitioned into slices of size

t_{slot}. For each demand $D_i \in \mathcal{D}$ and its corresponding concrete protocol $P_i(A_i, I_i, M_i, Q_i) \in \mathcal{P}$ we construct a task τ_i with phase $\Phi_i = 0$, period $T_i = \lfloor \frac{1}{R_i} \frac{1}{t_{slot}} \rfloor$, and execution time $C_i = \max_{a_j \in A_i}(M_i(a_j) + \lceil \frac{1}{a_{j,R}} \frac{1}{t_{slot}} \rceil)$ where $a_{j,R}$ is the rate at which action a_j is performed. That is, the execution time of a task τ_i is equal to the maximum relative finish time of any of the protocol actions.

We choose $\Phi_i = 0, \forall \tau_i \in \mathrm{T}$ so that we may predetermine the length of the schedule as the hyperperiod $H = lcm(T_1, ..., T_n)$. From this we may compute the number of instances of each task $\tau_i \in \mathrm{T}$ that must be scheduled as $\frac{H}{T_i}$. Finding a schedule $\mathcal{S}$ for the set of periodic tasks thus informs us how many times and when to schedule each $P_i \in \mathcal{P}$ in the TDMA schedule to satisfy the demanded rate R_i.

Symbol	Explanation
τ_i	A task.
$\tau_{i,j}$	The jth instance of task τ_i.
Φ_i	The phase of task τ_i, specifies the release time of task instance $\tau_{i,1}$.
$r_{i,j}$	The release time of task instance $\tau_{i,j}$, the earliest time at which a task instance may begin.
C_i	The worst-case execution time of task τ_i, each instance $\tau_{i,j}$ of τ_i has this worst-case execution time.
T_i	The period of task τ_i, specifies the amount of time between subsequent releases of instances of τ_i.
$\delta_{i,j}$	The absolute deadline of task instance $\tau_{i,j}$, the latest time at which a task instance may complete.
$\sigma_{i,j}$	The start time of task instance $\tau_{i,j}$.
$f_{i,j}$	The completion time of task instance $\tau_{i,j}$.
t_{slot}	The duration of a time slot.
H	The hyperperiod of a set of tasks.

Table 4.1: Summary of periodic task scheduling notation used.

4.2.2. NPRP TDMA Scheduling

To construct a TDMA schedule for the non-preemptive repeater protocol (NPRP) TDMA scheduling problem we construct a periodic taskset T using the method described previously and then use non-preemptive periodic task scheduling techniques to construct the schedule $\mathcal{S}$. The set of starting times described by $\mathcal{S}$ can then be used to determine when each repeater protocol starts in the TDMA schedule.

The standard non-preemptive periodic task scheduling problem assumes that there is a single resource (typically a CPU) shared among all tasks in the system. Thus, only a single task may execute at any moment in time. In a quantum network, the set of resources are the communication qubits and storage qubits of the network nodes. Constructing a TDMA schedule directly from the set of tasks may result in poor network utilization when the repeater protocols execute in independent portions of the network.

One can achieve better network utilization by preprocessing the taskset T and constructing a path-vertex intersection graph of the repeater chains for the set of repeater protocols. Each vertex in the path-vertex intersection graph represents a concrete repeater protocol while the edges connect vertices that share common network resources. Disjoint components of the path-vertex graph represent sets of repeater protocols that operate on network resources independent of one another. A network-wide TDMA schedule may be created by constructing a TDMA schedule for each set of repeater protocols corresponding to a disjoint component of the path-vertex intersection graph. In the case where there are many demands in the system and all repeater chains overlap with one another, it is not possible to decompose the problem further and extract parallelism from the task set. This results in treating the network as a single resource where only a single repeater protocol is active in the network at any time.

Construction of a path-vertex intersection graph has a complexity of $O(|C||\mathcal{P}|^2)$ by constructing a vertex for each repeater chain used by a concrete repeater protocol and then connecting pairs of vertices that have a common node in their repeater chain. Here, $|C|$ is the maximum number of nodes in any repeater chain and $\mathcal{P}$ is the set of repeater protocols where each protocol corresponds to a repeater chain. Once constructed, finding the set of disjoint components can be done in linear time in the number of vertices and edges in the path-vertex intersection graph.

Example 4.2.1. To demonstrate constructing the TDMA schedule using non-preemptive scheduling, suppose we have the four-node chain as in figure 4.5a and that we want to execute the repeater protocols P_1, P_2 shown in 4.5b with a time slot size of 10 ms. We distinguish the actions belonging to P_1 and P_2 with an apostrophe in the action label (i.e. L_1' for P_2 and L_1 for P_1). Suppose further that the demands for each protocol desire a rate of 25 $\frac{ebit}{s}$. With the specified slot size, this means the protocols have a period $T_1 = T_2 = 4$ slots while the periodic task τ_1 for protocol P_1 has phase $\Phi_1 = 0$ and worst-case execution time $C_1 = 3$. Periodic task τ_2 for protocol P_2 has phase $\Phi_2 = 0$ and worst-case execution time $C_2 = 1$. We see that the repeater protocols overlap at node B so they must be scheduled in the same taskset.

Using a non-preemptive scheduling heurstic known as non-preemptive earliest deadline first (NP-EDF), we obtain the task schedule shown in 4.5c which can then be transformed into the TDMA schedule shown in figure 4.5d. As we can see from the TDMA schedule, it is actually possible to construct a shorter schedule by placing P_2 in the vacant space during time slot 0. This inefficiency is a consequence of using the periodic task scheduling formulation and we will later show how the RCPSP formulation can overcome this.

To observe the performance of non-preemptive scheduling we implemented a simulated non-preemptive EDF algorithm that splits the taskset using a path-vertex intersection graph with complexity $O(N + |C||\mathcal{P}|^2)$ where N is the number of task instances induced by the taskset. Specifically, for a hyperperiod $H = lcm_{\tau_i \in \mathrm{T}}(T_i)$, the number of tasks $N = \sum_{\tau_i \in \mathrm{T}} \frac{T_i}{H}$. We also simulated the Clairvoyent EDF (CEDF) heuristic in [7] to see how non-work conserving scheduling techniques impact the quality of the produced TDMA schedule.

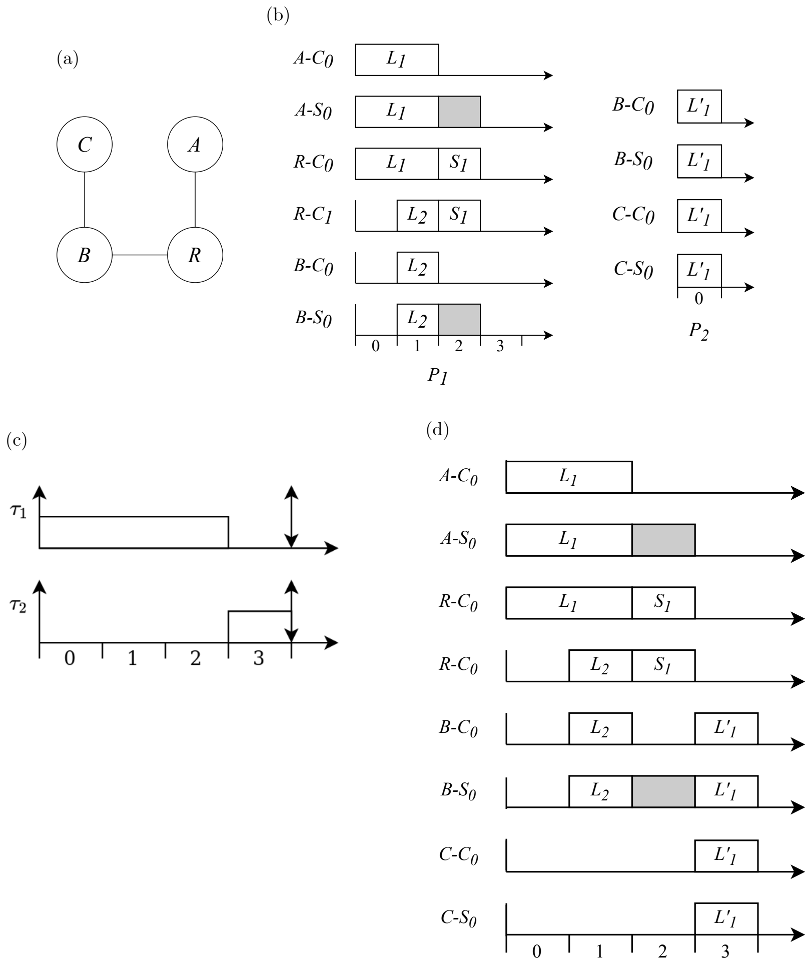


Figure 4.5: Example of constructing a TDMA schedule using the non-preemptive periodic task scheduling formulation. a) Four-node repeater chain network. b) The concrete repeater protocol to connect A, B (left) and the concrete repeater protocol to connect B, C (right). c) A valid schedule produced for periodic tasks τ_1 (P_1) and τ_2 (P_2). Upward arrows represent release times of the tasks while downward arrows represent deadlines. d) The TDMA schedule for the network obtained from the task schedule in c.

4.2.3. LPRP TDMA Scheduling

The non-preemptive scheduling taken in the previous section corresponds to a scenario where quantum devices observe limited qubit counts and are extremely sensitive to environmental noise. These limitations require strict adherence to protocol specification in order to meet QoS demands in quantum networks.

In their seminal work, Liu and Layland [8] showed that earliest deadline first (EDF) scheduling is optimal in the case of fully preemptive task scheduling. Unfortunately, introducing arbitrary preemption may delay actions in a repeater protocol beyond a tolerable limit and cause a repeater protocol to not meet QoS demands due to the reduction in the delivered end-to-end fidelity. Here, we consider the case where quantum memories have improved and permit more flexibility to the protocol behavior. Specifically, we consider the case where the order of operations in the protocol may not be altered, but the starting times of the individual operations may be delayed.

By delaying repeater protocol actions we effectively preempt the repeater protocol to allow the network resources to be used by a contending protocol. In this case, each repeater protocol P_i additionally specifies a *delay budget* $B_{i,delay}$ that specifies the maximum cumulative time that actions in the protocol may be delayed. For a slot size t_{slot} this corresponds to tolerating a delay of $\lfloor \frac{B_{i,delay}}{t_{slot}} \rfloor$ slots.

Previous research in limited preemption policies focus on methods such as preemption thresholds [9], deferred preemption [10], and fixed-point preemption [11]. These methods either disable preemption by other tasks or restrict preemption to specific times during a tasks execution but do not resolve the problem of limiting the amount of time a task remains preempted for once it has already been started. In practice, real-time systems such as sensor networks may depend on the freshness of collected data for proper operation and may require that sensor readings be processed within some maximum time window after they have been collected in order to be valid. In task scheduling, this effectively places a constraint on the completion time of a task once a starting time has been selected.

In the context of repeater protocols, we would like to make sure that a protocol is completed within some maximum amount of time once it has started in order to respect the demanded fidelity. In this section we first introduce the limited preemption task scheduling problem and its periodic task scheduling equivalent as well as a heuristic solution for producing a schedule. We then show a modification to the taskset construction for the NPRP case to suit the limited preemption repeater protocol (LPRP) TDMA scheduling problem.

System Model and Notations

We consider a uniprocessor system with a set T of preemptable independent tasks. Note that here we focus on *non*-periodic tasks before moving onto the periodic task formulation in later sections. The task set has n tasks denoted by $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$. Each task τ_i is identified by $\tau_i = (\Phi_i, C_i, D_i, B_i)$ where Φ_i is the release offset of task τ_i, C_i is the worst-case execution time, D_i is the absolute deadline, and B_i is the *preemption budget* of the task. B_i specifies the maximum cumulative amount of time τ_i is allowed to be preempted for once it has begun. More concretely, if τ_i is started at some time t_0 then it must complete execution before $t = t_0 + C_i + B_i$ regardless of the number of times the task is preempted. We assume that all parameters are integer-valued and that time is partitioned into slots.

Limited Preemption Budget Task Scheduling

Limited preemption budget task scheduling is the problem of constructing a map $\mathcal{S}$ such that $\mathcal{S}(\tau_i) = \{(t^i_{s1}, t^i_{e1}), ..., (t^i_{sk}, t^i_{ek})\}$ is an ordered set of non-overlapping time intervals that speciy when τ_i executes on the uniprocessor such that the following properties hold:

- $\Phi_i \leq t^i_{sj} < t^i_{ej} \leq D_i$, $1 \leq j \leq k$
- $\sum_j (t^i_{ej} - t^i_{sj}) = C_i$
- $t^i_{ek} - t^i_{s1} \leq C_i + B_i$.
- No two tasks τ_i, τ_j have intervals $(t^i_{sl}, t^i_{el}) \in \mathcal{S}(\tau_i), (t^j_{sm}, t^j_{em}) \in \mathcal{S}(\tau_j)$ overlap with one another

Thus we want a set of execution intervals for each task such that the sum of the intervals is the execution time of the task, the period of time over which the intervals occupy do not exceed the computation time and preemption budget, and no two tasks are executing at the same time.

We now introduce terminology that will be used throughout the remainder of this section.

Definition 4.2.1. Consider a task τ_i and let $E^i = \{(t^i_{s1}, t^i_{e1}), ..., (t^i_{sk}, t^i_{ek})\}$, $t^i_{sj}, t^i_{ej} \in \mathbb{Z}, 1 \leq j \leq k$ be a set of execution intervals for τ_i such that $\Phi_i \leq t^i_{s1} < t^i_{e1} < t^i_{s2} < ... < t^i_{sk} < t^i_{ek} \leq D_i$ and $\sum_{j=1}^{k}(t^i_{ej} - t^i_{sj}) = C_i$. Then τ_i *adheres* to its preemption budget *iff* $t^i_{ek} - t^i_{s1} \leq C_i + B_i$.

Definition 4.2.2. Consider a set of tasks $\mathrm{T} = \{\tau_1, ..., \tau_n\}$ and a function $\mathcal{S}$ that maps each task τ_i to a set of execution intervals E^i. Then we say $\mathcal{S}$ is a *valid preemption budget schedule* if the following hold:

1. $\forall \tau_i \in \mathrm{T}, (t^i_{sj}, t^i_{ej}) \in \mathcal{S}(\tau_i); \Phi_i \leq t^i_{sj} < t^i_{ej} \leq T_i$
2. $\forall \tau_i \in \mathrm{T}, \forall (t^i_{sj}, t^i_{ej}), (t^i_{sk}, t^i_{ek}) \in \mathcal{S}(\tau_i); ((t^i_{sk} \geq t^i_{ej}) \vee (t^i_{sj} \geq t^i_{ek}))$
3. $\forall \tau_i; \sum_{(t^i_{sj}, t^i_{ej}) \in \mathcal{S}(\tau_i)}(t^i_{ej} - t^i_{sj}) = C_i,$
4. $\forall \tau_i; \max_{(t^i_{sj}, t^i_{ej}) \in \mathcal{S}(\tau_i)}(t^i_{ej}) - \min_{(t^i_{sj}, t^i_{ej}) \in \mathcal{S}(\tau_i)}(t^i_{sj}) \leq C_i + K_i$
5. $\forall \tau_i, \tau_j \in \mathrm{T}, (t^i_{sk}, t^i_{ek}) \in \mathcal{S}(\tau_i), (t^j_{sl}, t^j_{el}) \in \mathcal{S}(\tau_j); ((t^i_{sj} \geq t^j_{el}) \vee (t^j_{sl} \geq t^i_{ek}))$

Where the first two conditions constrain the set of execution intervals to lie within the interval $[\Phi_i, D_i]$, the next two conditions ensure the task is reserved enough execution time to complete without exceeding its preemption budget, and the last condition ensures no two tasks are executing at the same time.

Using these conditions we can formulate this problem as an integer program in the following way. We use binary decision variables $X = \{x_{il}, i = 1, ..., n, l = 1, ..., \lambda\}$ where λ is the maximum length of the schedule, i denotes the task τ_i and l indicates a slot number. x_{il} is true when τ_i is active in slot l.

$$\min 1 \tag{4.4}$$

$$\text{s.t.} \sum_{l=\Phi_i}^{D_i} x_{il} = C_i \quad 1 \leq i \leq n \tag{4.5}$$

$$\sum_{i=1}^{n} x_{il} \leq 1 \quad 1 \leq l \leq \lambda \tag{4.6}$$

$$x_{il}x_{ij}(j - l) \leq C_i + B_i \quad 1 \leq i \leq n, 1 \leq l \leq \lambda \tag{4.7}$$

$$x_{il} \in \{0, 1\}, \tag{4.8}$$

where we use $\tau_i = (\Phi_i, C_i, D_i, B_i)$ as the task definition. Constraint 4.5 ensures that τ_i is given enough slots for computation within the interval of time it is available to the system. Constraint 4.6 ensures that only one task is active at a time in any slot. Constraint 4.7 bounds the amount of time spanning any two slots where the task is active, in the extreme this corresponds to the difference between the largest l where $x_{il} = 1$ and the smallest l where $x_{il} = 1$. Given a solution to the IP, a schedule $\mathcal{S}$ can be constructed by constructing the set of intervals E^i for each task τ_i using the decision variables. In this case, the objective function remains trivial as our goal is to come up with any valid schedule.

While we can formulate the limited preemption budget scheduling problem as an IP, it requires full knowledge of the tasks a priori and does not form a practical approach when decisions must be made real-time. It is thus necessary to introduce a heuristic approach that can be performed at run-time. In the following section, we will introduce a heuristic we refer to as *earliest-deadline-first least-budget-first* (EDF-LBF) for scheduling tasks in this setting.

EDF-LBF

Before describing our heuristic we will begin by introducing additional definitions and observations of the problem that motivate the design of the heuristic.

Definition 4.2.3. For a given schedule $\mathcal{S}$ and any task τ_i with $\mathcal{S}(\tau_i) = \{(t^i_{s1}, t^i_{e1}), ..., (t^i_{sk}, t^i_{ek})\} = E^i$ such that $t^i_{s1} < t^i_{e1} < ... < t^i_{sk} < t^i_{ek}$, let $C_i(t)$ denote the amount of outstanding execution time to be completed for τ_i at a time t as follows:

$$C_i(t) = \begin{cases} C_i - \sum_{(t^i_{sj}, t^i_{ej}) \in E^i | t^i_{sj} < t} (\min(t^i_{ej}, t) - t^i_{sj}) & t^i_{s1} \leq t < t^i_{ek} \\ 0 & else \end{cases} \tag{4.9}$$

where $\min()$ is used as t might correspond to a time between two execution intervals.

Definition 4.2.4. A task τ_i is *active* at time t if $C_i(t) > 0$. The set of active jobs at time t is $A(t) = \{\tau_i \in \mathrm{T} | 0 < C_i(t)\}$.

Definition 4.2.5. For a given schedule $\mathcal{S}$ and any task τ_i with $\mathcal{S}(\tau_i) = \{(t^i_{s1}, t^i_{e1}), ..., (t^i_{sk}, t^i_{ek})\} = E^i$ such that $t^i_{s1} < t^i_{e1} < ... < t^i_{sk} < t^i_{ek}$, let

$B_i(t)$ denote the amount of remaining preemption budget for τ_i at time t as follows:

$$B_i(t) = \begin{cases} B_i & t \leq t^i_{s1} \\ B_i - (t - t^i_{s1}) + (C_i - C_i(t)) & t^i_{s1} < t \leq t^i_{ek} \\ B_i(t^i_{ek}) & t > e_k \end{cases} \tag{4.10}$$

Using these definitions we make the following observation of all valid preemption budget schedules $\mathcal{S}$.

Theorem 4.2.1. *For any set of active tasks $A(t) = \{\tau_1, ..., \tau_m\}$ ordered by completion times such that $t^1_{e|E^1|} < t^2_{e|E^2|} < ... < t^m_{e|E^m|}$ in a valid preemption budget schedule $\mathcal{S}$, the set of active tasks must satisfy the following relation:*

$$\forall i, 2 \leq i \leq m; \sum_{j=1}^{i-1} C_j(t) \leq B_i(t) \tag{4.11}$$

That is, the ith task in this order has enough remaining preemption budget $B_i(t)$ to tolerate the cumulative remaining execution times of all tasks that complete before it.

Proof. Given the order of completion times we know that:

$$t < t{+}C_1(t) \leq t_{e|E^1|} < t{+}C_1(t){+}C_2(t) \leq t_{e|E^2|} < ... < t{+}\sum_{j=1}^{m} C_j(t) \leq t_{e|E^m|} \tag{4.12}$$

That is, if the completion of a task τ_i is preceded by the completion of tasks $\{\tau_1, \tau_2, ..., \tau_{i-1}\}$ then the earliest time $t_{e|E^i|}$ where $C_i(t_0) = 0$ is $t_{e|E^i|} = t + \sum_{j=1}^{i} C_j(t)$. Because $\mathcal{S}$ is a valid preemption budget schedule we know that the latest time $t_{e|E^i|}$ where $C_i(t_0) = 0$ is $t_{e|E^i|} = t + C_i(t) + B_i(t)$. We thus have that:

$$t + \sum_{j=1}^{i} C_j(t) \leq t_{e|E^i|} \leq t + C_i(t) + B_i(t) \tag{4.13}$$

$$\sum_{j=1}^{i} C_j(t) \leq C_i(t) + B_i(t) \tag{4.14}$$

$$\sum_{j=1}^{i-1} C_j(t) \leq B_i(t). \tag{4.15}$$

■

When one considers the set of active tasks $A(t) = \{\tau_1, ..., \tau_m\}$ ordered by completion times we can also make the observation that tasks τ_i that complete later require a larger amount of remaining preemption budget $B_i(t)$ in order to tolerate the cumulative remaining execution time.

Definition 4.2.6. A least-budget-first (LBF) ordering $\{\tau_1, ..., \tau_m\}$ of a set of tasks T is an ordering such that if $i < j$ then $B_i(t) \leq B_j(t) \forall i, j$.

With these definitions in place we now introduce the EDF-LBF heuristic for solving the limited preemption budget task scheduling problem. We know that EDF is optimal in the fully preemptive task scheduling case. Here, we use the EDF heuristic to apply a priority ordering to a ready queue of tasks. A second priority queue is maintained for the set of active tasks where the tasks are ordered by LBF. When new tasks are introduced into the system they must pass an admittance test before they may begin. This admittance test is used to ensure that the preemption budgets of the currently active tasks and the new task in question are not violated by starting the new task.

Definition 4.2.7. For a least budget first ordering of the active jobs $A(t) = \{\tau_1, ..., \tau_m\}$ we define the excess budget of a task $\tau_i \in A(t)$ to be:

$$B_{i,ex}(t) = B_i(t) - \sum_{j=1}^{i-1} C_j(t). \tag{4.16}$$

We can admit a new task τ_j into the set of active tasks ordered by LBF at a time t by assigning it an execution interval $(t, t+\delta)$ for $\delta \leq C_j$ if one of the two following sets of conditions holds:

1. $C_j = \delta$: $B_{i,ex}(t) \geq \delta \wedge D_j \leq D_i, \forall \tau_i \in A(t)$
2. $C_j > \delta$:
   (a) $B_{i,ex}(t) \geq \delta, \forall \tau_i \in A(t) | (B_i(t) - \delta) \geq B_j$
   (b) $B_{i,ex}(t) \geq C_j, \forall \tau_i \in A(t) | (B_i(t) - \delta) < B_j$
   (c) $B_j \geq \sum_{\tau_i \in A(t) | (B_i(t) - \delta) \geq B_j} C_i(t)$

(d) $D_j \leq D_i, \forall \tau_i \in A(t)$,

where the first condition states that if we want to run the new task τ_j to completion then it must have an earlier deadline than all other active tasks and the excess budget of all active tasks must be able to tolerate the execution time.

The second condition states that τ_j will be preempted by an active task after δ units of time and inserted into the LBF ordered active tasks. We thus need the tasks of higher priority at the time of preemption to have enough excess budget to accommodate a delay of δ. The tasks of lower priority in the active set at the time τ_j is preempted must have enough excess budget to accommodate the full execution. We also require that the new task τ_j has a preemption budget that is able to accommodate the cumulative execution time of all higher priority tasks in the active set at the time of preemption.

The conditions presented can thus be formulated as an admittance test as shown in algorithm 1. Here, the admittance test returns $(True, \delta)$ if we can preempt the current task and execute the new task for δ time and preempt it afterwards. Returning $(False, 0)$ means we need to postpone τ_i until the constraints imposed by the set of active tasks relax. We now proceed to provide a worst-case response time analysis of this scheduling heuristic.

Algorithm 1: EDF-LBF Admittance Test

```
Result: Returns $True$ and an amount of time to run $\tau_i$ for
        before preempting or $False$.
$\tau_i$ - New task
$t$ - Current Time
$B_{excess} = 0$;
$C_{total} = 0$;
for $\tau_l \in A(t)$ sorted by $B_l(t)$ do
    $B_{excess} = min(B_l(t) - C_{total}, B_{excess})$;
    $C_{total} = C_{total} + C_l(t)$;
    if $D_i > D_l$ then
        return $(False, 0)$;
end
if $B_{excess} \geq C_i$ then
    return $(True, C_i)$;
else if $B_{excess} = 0$ then
    return $(False, C_{delay})$;
else
    $C_{rem} = C_i - B_{excess}$;
    $B_{del} = 0$;
    $C_{total} = 0$;
    for $\tau_l \in A(t) | B_l(t) - B_{excess} \geq B_i$ do
        if $B_l(t) - B_{excess} < C_{rem}$ then
            return $(False, 0)$;
    end
    return $(True, K_{excess})$;
```

Worst-Case Response Time Analysis

Recall from chapter 2 that the worst-case response time of a task τ_i is the difference between it's completion time and the time at which it was released into the system. For EDF-LBF, a task τ_i experiences delay due to being blocked by higher priority tasks similarly to the case in EDF, but can additionally experience delay due to being preempted once it has been admitted into the system. We can thus summarize the worst-case response time as,

$$WCRT(\tau_i) = d_{start} + d_{preempt} + C_i, \tag{4.17}$$

where d_{start} is the delay imposed by higher priority tasks and the constraints of the set of active tasks, $d_{preempt}$ is the delay due to preemption once a task has started, and C_i is the execution time of a task. Due to the admittance test for starting a new task we know that $d_{preempt} \leq B_i$.

To identify d_{start} we consider the critical instant for τ_i. A critical instant for a task is an instant at which that task will have the largest response time. In EDF-LBF, the critical instant for a task τ_i occurs when it is released at the same time as all higher priority tasks and all lower priority tasks have previously been admitted to begin execution. The delay involved before a task τ_i can begin thus comes from 1) waiting to get to the head of the ready queue and 2) waiting to be admitted into the set of active tasks. In the worst case, the preemption budgets of all tasks do not permit any preemption and τ_i must complete after all other tasks have finished. Thus $d_{start} = -\delta n_{lp} + \sum_{\tau_l \in \mathrm{T}} C_l$ where n_{lp} is the number of lower priority tasks than τ_i. The reason we subtract this quantity is because newly admitted tasks must run for some minimum amount of time δ in order to be considered active. Thus we obtain:

$$WCRT(\tau_i) = B_i + C_i - \delta n_{lp} + \sum_{\tau_l \in \mathrm{T}:\tau_l \neq \tau_i} C_l. \tag{4.18}$$

Limited Preemption Budget Periodic Task Scheduling

In order to use the limited preemption budget task scheduling formulation for scheduling repeater protocols, we need to have a periodic task formulation so that we can use similar techniques to those for the NPRP TDMA scheduling case.

Here, a periodic task τ_i is associated with the same quantities Φ_i, C_i, T_i as in the standard periodic task scheduling setting and additionally specifies the preemption budget B_i. In the periodic task scheduling setting, the preemption budget B_i specifies the maximum amount of delay that any instance $\tau_{i,j}$ of τ_i can experience. Specifically, for each instance $\tau_{i,j}$ of a task τ_i we must have $f_{i,k} \leq \sigma_{i,k} + C_i + B_i$.

Before moving onto an example of EDF-LBF, we would like to remind the reader of two very well-studied scheduling heuristics known as preemptive earliest deadline first (EDF) and non-preemptive earliest deadline first (NP-EDF). At each time instance, both of these scheduling heuristics checks the set of released task instances and attempts to schedule the one with the earliest deadline. The key difference between these two heuristics is that EDF allows for switching from a currently executing task instance to a different one while NP-EDF prevents any sort of interruption.

Example 4.2.2. We demonstrate the operation of these two scheduling heuristics with an example. Suppose we have a task set T containing two tasks $\tau_1 = (\Phi_1 = 0, C_1 = 1, T_1 = 3)$ and $\tau_2 = (\Phi_2 = 0, C_2 = 4, T_2 = 9)$. More concretely, task τ_1 (τ_2) has a release offset of $\Phi_1 = 0$ ($\Phi_2 = 0$), a worst-case execution time of $C_1 = 1$ ($C_2 = 4$), and a period of $T_1 = 3$ ($T_2 = 9$). Figure 4.6 demonstrates the operation of these two heuristics.

We begin by explaining the flow of EDF. At time slot 0, the instances $\tau_{1,1}$ and $\tau_{2,1}$ are released into the system as $\Phi_1 = \Phi_2 = 0$. Since $T_1 = 3$, the absolute deadline $d_{1,1}$ of task instance $\tau_{1,1}$ is 3 and similarly we may conclude the absolute deadline $d_{2,1}$ for task instance $\tau_{2,1}$ is 9. EDF chooses the task instance with the earliest deadline first and thus schedules $\tau_{1,1}$ to run. At time slot 1, instance $\tau_{1,1}$ has completed and instance $\tau_{2,1}$ is the only released task instance, so EDF schedules $\tau_{2,1}$ to run. Instance $\tau_{1,2}$ is then released at time slot 3 and has an absolute deadline of 6, thus EDF will preempt $\tau_{2,1}$ from running and switch to $\tau_{1,2}$. Once instance $\tau_{1,2}$ has completed, EDF will switch back to $\tau_{2,1}$ and complete before $\tau_{1,3}$ is released. At this point EDF schedules the lone instance $\tau_{1,3}$ and the schedule repeats at time slot 9.

In contrast, NP-EDF is not allowed to interrupt currently executing task instances when new task instances with earlier deadlines are released into the system. At time slot 0, NP-EDF makes the same choice and schedules $\tau_{1,1}$ followed by $\tau_{2,1}$. At this point, NP-EDF may not schedule any new instances until $\tau_{2,1}$ has completed at time slot 5. Once $\tau_{2,1}$ is completed, $\tau_{1,2}$ may be scheduled followed by $\tau_{1,3}$ and the schedule repeats at time slot 9.

Example 4.2.3. We now present a motivating example of the use of EDF-LBF to show its performance over both non-preemptive and pre-

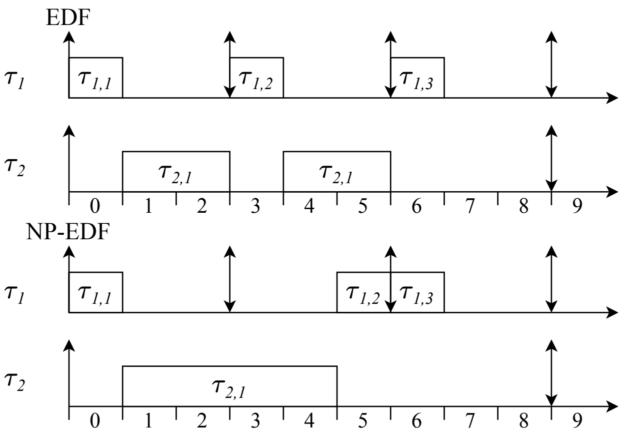


Figure 4.6: An example of scheduling a periodic task set using preemptive earliest deadline first (EDF) and non-preemptive earliest deadline first (NP-EDF). Upward arrows denote release times of task instances while downward arrows correspond to deadlines of task instances. Both heuristics schedule the task set successfully.

emptive EDF. Suppose we have two periodic tasks τ_1 and τ_2 such that $\tau_1 = (\Phi_1 = 0, C_1 = 2, T_1 = 4, B_1 = 0)$ and $\tau_2 = (\Phi_2 = 0, C_2 = 6, T_2 = 12, B_2 = 2)$. Figure 4.7 shows how the three scheduling heuristics would construct the schedule.

The reason that EDF fails to satisfy the limited preemption budget scheduling problem is because τ_2 is preempted for a total of 4 time units as the third instance $\tau_{1,3}$ preempts $\tau_{2,1}$ causing the budget to be violated. Non-preemptive EDF causes $\tau_{1,2}$ to miss its deadline since $\tau_{2,1}$ cannot be preempted. Out of these three, only EDF-LBF was able to satisfy the constraints of the task set.

LPRP TDMA Scheduling

The limited preemption repeater protocol (LPRP) TDMA scheduling problem can be turned into the limited preemption budget task scheduling problem as follows. Given the set of demands $\mathcal{D}$ and their concrete repeater protocols $\mathcal{P}$ along with a specified slot size t_{slot} we can construct the set of periodic tasks T by constructing a periodic task τ_i for each demand, protocol pair (D_i, P_i) by choosing phase $\Phi_i = 0$, period $T_i = \lfloor \frac{1}{t_{slot} R_i} \rfloor$, execution time $C_i = \max_{a_j \in A_i}(M_i(a_j) + \lceil \frac{1}{t_{slot} a_{j,R}} \rceil)$, and preemption budget $B_i = \lfloor \frac{B_{i,delay}}{t_{slot}} \rfloor$.

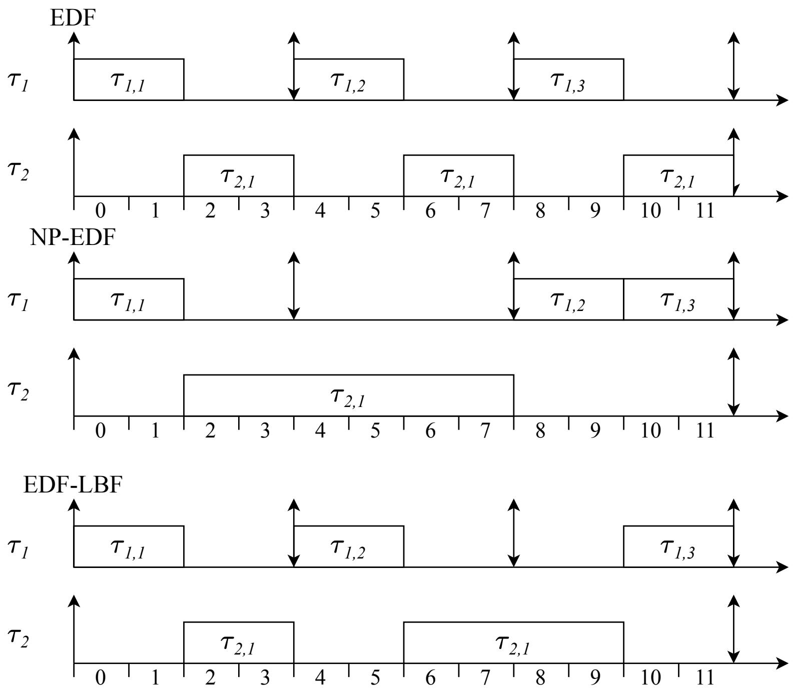


Figure 4.7: An example of scheduling a periodic taskset using EDF, NP-EDF, and EDF-LBF. Here, EDF violates the budget constraints of τ_2 due to its aggressive preemptive nature, NP-EDF causes $\tau_{1,2}$ to miss its deadline, and EDF-LBF successfully schedules within the task constraints.

Our choice of phase $\Phi_i = 0$ for all tasks τ_i is so that we may predetermine the schedule length as the hyperperiod $H = lcm_{\tau_i \in \mathrm{T}}(T_i)$ as well as the number of instances each periodic task produces within this duration of time.

When scheduling repeater protocols using this method it is necessary to prevent preemptions that may occur while a protocol action is in progress or while required resources are still occupied. In the case when the preemption budget of all active tasks permit fully executing the new task, we only need to consider whether the necessary resources are available. If the preemption budgets of active tasks require that a newly started task be preempted before completion, we need to ensure that the required resources are available for resuming tasks. It is thus necessary to augment the admittance test presented in section 4.2.3 to additionally prevent preemptions in these cases.

In order to track resource requirements during a repeater protocol's execution, we may determine a set of preemption points in the protocol where no action is in progress. The set of actions that lie between these preemption points thus form non-preemptive segments of the protocol that have a fixed set of resource requirements reflecting the set of actions within the segment and those that follow the segment.

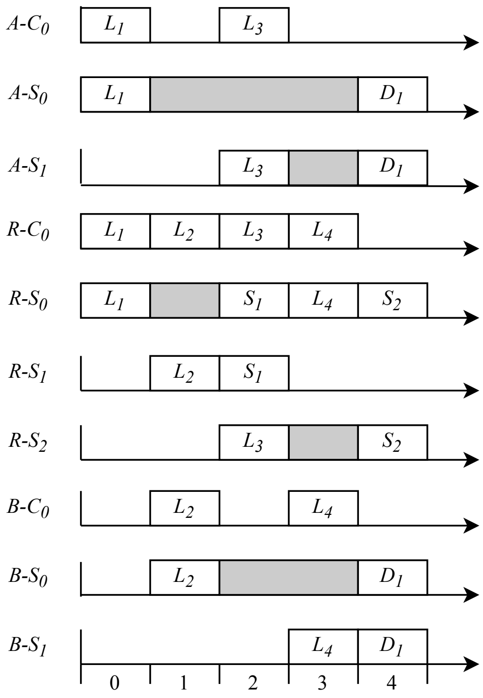


Figure 4.8: An example of a concrete protocol where actions overlap with one another in time. First, A and R generate entanglement and store it in their storage qubits A-S_0 and R-S_0 respectively. R and B then generate entanglement and store it in their storage qubits R-S_1 and B-S_0. R then performs entanglement swapping using qubits R-S_0 and R-S_1 while generating entanglement with A in parallel. After generating entanglement with B again, R performs entanglement swapping using qubits R-S_0 and R-S_1 while A and B perform entanglement distillation using their storage qubits A-S_0, A-S_1, B-S_0, and B-S_1.

Example 4.2.4. We illustrate the idea of protocol segments using the concrete protocol shown in figure 4.8. Here, we see that at time slot 2 the actions L_3 and S_1 occur in parallel while in time slot 4 the actions D_1 and S_2 occur in parallel. Thus each time slot 0,...,4 represents a segment of the protocol and a summary of the actions in each segment can be seen in table 4.2. The required resources are those that are needed for the remainder of the protocol while the occupied resources are those that hold an entangled link at the start of the segment.

Segment	1	2	3	4	5
Actions	L_1	L_2	L_3, S_1	L_4	S_2, D_1
Req. Resources	A-C_0, A-S_0, A-S_1, R-C_0, R-S_0, R-S_1 R-S_2 B-C_0, B-S_0, B-S_1	A-C_0, A-S_0, A-S_1, R-C_0, R-S_0, R-S_1 R-S_2 B-C_0, B-S_0, B-S_1	A-C_0, A-S_0, A-S_1, R-C_0, R-S_0, R-S_1 R-S_2 B-C_0, B-S_0, B-S_1	A-S_0, A-S_1, R-C_0, R-S_0, R-S_2 B-C_0, B-S_0, B-S_1	A-S_0, A-S_1, R-S_0, R-S_2, B-S_0, B-S_1
Occ. Resources		A-S_0, R-S_0	A-S_0, R-S_0 R-S_1 B-S_0	A-S_0, A-S_1 R-S_2 B-S_0	A-S_0, A-S_1 R-S_0 R-S_2 B-S_0 B-S_1

Table 4.2: Table showing the segments corresponding to the protocol in figure 4.8 and a description of the actions, required resources, and occupied resources during each segment. In segment 1, only an elementary link generation is performed and all resources for the full protocol are required while there are no occupied resources. After segment 1, segment 2 performs another elementary link generation and requires the full set of resources but now has entanglement stored in A-S_0 and R-S_0 from segment 1. Segment 3 contains both an elementary link generation and entanglement swap action and occupies additional qubits to store entanglement. In segment 4, an final elementary link is generated and the required resources changes as A-C_0 and R-S_1 are no longer used. The occupied resources no longer include R-S_0 and R-S_1 as entanglement swapping frees the qubits. Segment 5 then completes the protocol by performing an entanglement swap and entanglement distillation in parallel.

For each periodic task τ_i we thus determine a set of fixed preemption points that lie between protocol actions. For each preemption point we additionally determine the set of resources that are occupied by the repeater protocol and the set of resources that are required in order to continue execution of the protocol. Determining the set of preemption points can be done by using an interval tree data structure and inserting intervals for each action in the repeater protocol. We can then merge overlapping intervals into continuous intervals to produce segments of actions that correspond to non-preemptable regions of the task. Using an interval tree we can preprocess a periodic task in $O(n \log(n))$ time where n is the number of actions in the protocol. The preemption points of a task lie at the boundaries between intervals in the interval tree and we can determine the set of occupied resources and required resources by inspecting the previously executed segment of actions and the succeeding segment of actions respectively. This can be done a priori for each task and the information can be reused for each instance of the periodic task.

Using the fixed preemption point information, the admittance test shown in Algorithm 2 can be performed in $O(K|\mathcal{D}|)$ as the number of demands $|\mathcal{D}|$ limits the number of tasks that can be in the active queue and there are K resources. Since the set of active tasks execute in LBF order, we can determine what resources are required for each active task to resume and determine if the new task can begin so that it is inserted into the LBF order without violating such resource requirements. Suppose that a set of periodic tasks T induces a set of N task instances to execute within the schedule of length H. The algorithm can thus run in $O(NSK|\mathcal{D}|)$ where S is the maximum number of segments of any task. This corresponds to the case where we attempt to run the admittance algorithm at every preemption point of an executing task.

Once a schedule $\mathcal{S}$ has been constructed for the tasks we can create a TDMA schedule that segments each repeater protocol into the intervals specified by $\mathcal{S}$ and assign start times to the actions of the repeater protocol based on the intervals in which the segments are executed. To conclude this section, we illustrate an example of how such a preempted schedule may be constructed.

Algorithm 2: Resource-constrained EDF-LBF Admittance Test

```
Result: If $\tau_i$ can begin execution and how much time it should
        run for before preempting into active queue.
$\tau_i$ - New task
$\tau_a$ - Currently active task
$t$ - Current time
$R_C$ - Currently occupied resources
$R_a$ - Resources occupied by $\tau_a$
$B_{excess} = 0$;
$C_{total} = 0$;
if any resources required by $\tau_i$ is in $R_C$ then
    return $(False, 0)$;
for $\tau_l \in A(t)$ sorted by $B_l(t)$ do
    if $\tau_l \neq \tau_a$ and any resources required by $\tau_l$ is in $R_a$ then
        return $(False, 0)$
    else if $\tau_l = \tau_a$ then
        $R_a = \{\}$;
    $B_{excess} = min(B_l(t) - C_{total}, B_{excess})$;
    $C_{total} = C_{total} + C_l(t)$;
    if $D_i > D_l$ then
        return $(False, 0)$;
end
if $B_{excess} \geq C_i$ then
    return $(True, C_i)$;
else if $B_{excess} = 0$ then
    return $(False, 0)$;
else
    $C_{max}$ = max preemption point $\leq B_{excess}$;
    $R_{O,max}$ = resources occupied by $\tau_i$ at preemption point;
    $C_{rem} = C_i - C_{max}$;
    for $\tau_l \in A(t)|$ do
        if $B_l(t) - B_{excess} \geq B_i$ and $B_l(t) - B_{excess} < C_{rem}$ then
            return $(False, 0)$;
        else if $B_l(t) - B_{excess} < B_i$ and $\tau_l$ requires any resource
          in $R_{O,max}$ then
            return $(False, 0)$;
    end
    return $(True, C_{max})$;
```

Example 4.2.5. Suppose that we have the four-node network shown in 4.9a and we wish to schedule the protocol shown in figure 4.8 (here P_2) in addition to a simple link protocol shown in figure 4.9b (here P_1). Let the slot size be 10 ms and the demand for protocol P_1 be 25 $\frac{ebit}{s}$ while the demand for protocol P_2 is 12.5 $\frac{ebit}{s}$ with $B_{2,delay} = 10$ ms. Then task τ_1 has $\Phi_1 = 0, C_1 = 1, T_1 = 4$ while τ_2 has $\Phi_2 = 0, C_2 = 6, T_2 = 8$ and $B_2 = 1$. Note that since $C_1 = 1$ the specification of $B_{1,delay}$ has no impact on the task definition given that the minimum execution time is 1. Using the EDF-LBF heuristic we produce the task schedule shown in 4.10a and we can see that τ_1 preempts τ_2 at a moment when the resource A-C_0 becomes available for use. Once this schedule is constructed, we know that for τ_2 the assigned execution intervals are $(1, 3)$ and $(5, 6)$ so we know that segments 1-3 execute within the first interval and segments 4-5 execute in the second, giving us the TDMA schedule shown in figure 4.10b. We also see that the introduction of shaded regions between segments 3 and 4 of P_2 due to its preemption.

From the produced schedule one may observe that preemption was not entirely necessary as P_1 no longer used A-C_0. This behavior is a consequence of the periodic task scheduling formulation since overlapping repeater chains treat the network as a single resource. We will later show how this can be avoided using the RCPSP formulation.

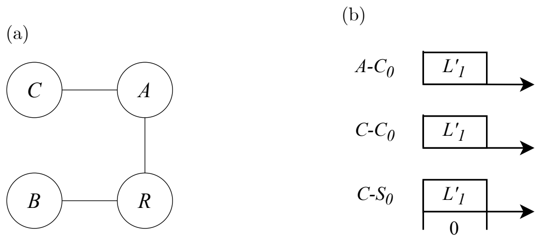


Figure 4.9: a) A four-node network for the example. b) The concrete repeater protocol composed of a single elementary link generation used to connect A and C.

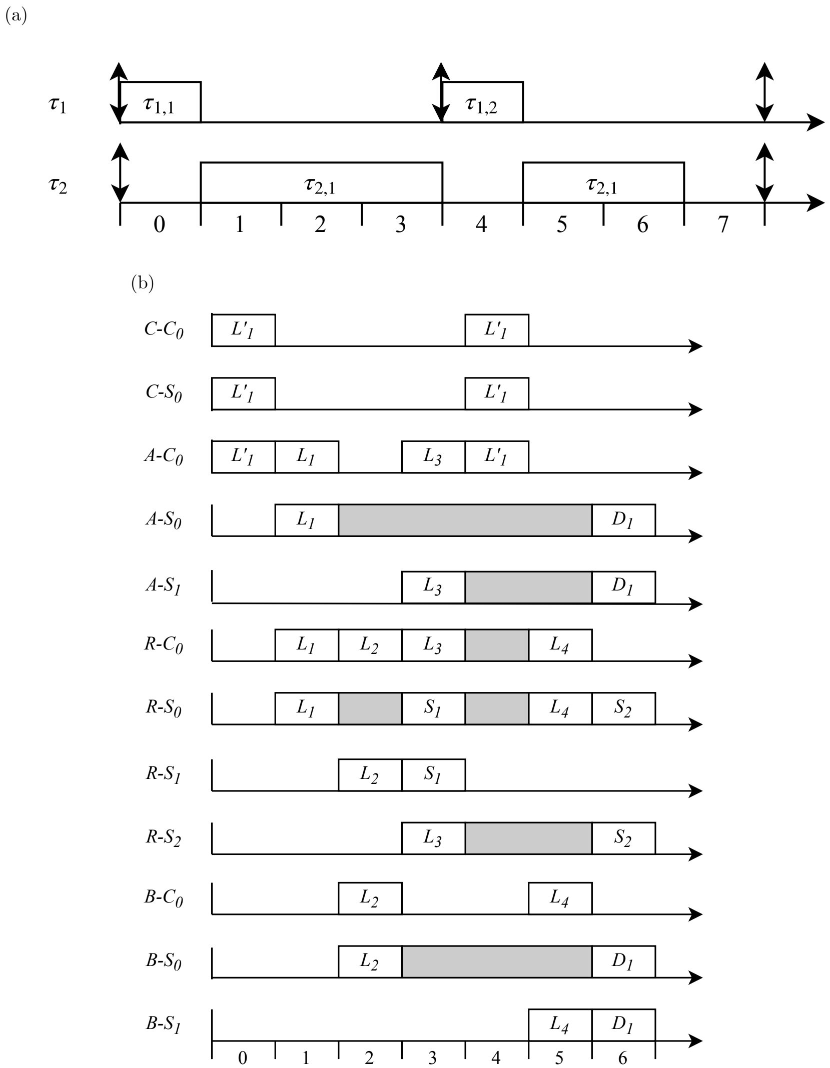


Figure 4.10: An example of using EDF-LBF to produce a TDMA schedule for the LPRP TDMA scheduling problem. a) An EDF-LBF scheduling of the tasks representing the protocols to be executed. b) The resulting TDMA schedule reconstructed from the task schedule.

4.3. RCPSP Formulation

The previous section described how constructing a TDMA schedule can be reduced to a periodic task scheduling problem where the periodic tasks represent the repeater protocols to be executed in the network and the periods of the tasks correspond to the demanded rates of those protocols. As we saw, when the set of repeater chains form a connected path-vertex intersection graph, the amount of parallelism we can extract from the schedule is reduced. This can cause the overall achievable network throughput to be reduced as there are unused portions of the network that may be used to execute repeater protocols.

Consider the case where a network node is an intersection point for two repeater protocols operating on repeater chains of many ndoes, but the intersecting node only generates a single link at the beginning of each overlapping protocol. After generating the link for one protocol, the node sits idle until the repeater protocol has completed, at which point the second protocol may begin. If the node in question immediately began generating the link for the subsequent protocol, then the second protocol could complete earlier, increasing the network throughput.

To alleviate this, we may transform the problem of constructing the TDMA schedule into a resource-constrained project scheduling problem (RCPSP). Recall from chapter 2 that RCPSP is traditionally used to represent project management in the presence of resource constraints and precedence relations between activities. In this formulation, the activities in the activity-on-node network graph correspond to the individual actions for each repeater protocol. Furthermore, we reuse the notion of the hyperperiod H from the periodic task scheduling formulation to determine how many times each repeater protocol action appears in the project.

In contrast to periodic task scheduling, using RCPSP allows greater flexibility in determining start times as network resources are treated individually. This overcomes the limitation of periodic task scheduling where the tasks assume there is a single shared resource for all tasks. In this section, we will first explain how the NPRP TDMA scheduling problem can be formulated as an RCPSP. We then show how the LPRP TDMA scheduling problem may be formulated as a multi-mode RCPSP (MRCPSP). Each of these formulations will additionally provide a project transformation that provides a trade-off between the complexity of computing a schedule and the quality of the schedule.

Symbol	Explanation
J	The set of all activities in the project.
j_i	An activity in the project.
j_s	Dummy start activity in acitivity-on-node network. Has zero processing time and no resource requirements.
j_e	Dummy end activity in acitivity-on-node network. Has zero processing time and no resource requirements.
p_i	Processing time of activity j_i.
h_{ik}	Required number of resource k by activity j_i to execute. j_i may not begin unless h_{ik} units of resource k are available.
d_{ij}	Minimal time lag between activities j_i and j_j. Constraints the starting time of j_j to be at least d_{ij} units after the completion time of j_i.
$\bar{d}_{ij}$	Maximal time lag between activities j_i and j_j. Constraints the starting time of j_j to be at most d_{ij} units after the completion time of j_i.
p_{lm}	Processing time of activity j_l using mode m. An activity may specify several modes and execution times.
h_{lmk}	Required number of resource k by activity j_i in mode m to execute. j_l may not be executed in mode m unless h_{lmk} units of resource k are available.

Table 4.3: Summary of RCPSP notation used.

4.3.1. NPRP TDMA Scheduling

In order to use RCPSP methods for producing a TDMA schedule, one first needs to construct the activity-on-node network representing the scheduling problem. First, we will show how we may construct an activity-on-node network for a concrete repeater protocol before constructing the activity-on-node network for a set of demands $\mathcal{D}$ and their concrete protocols $\mathcal{P}$. We then show a project transformation that may reduce the complexity of computing a valid schedule.

Project Construction

Activity Construction To construct an activity-on-node network for a concrete repeater protocol $P(A, I, M, Q)$ we first begin by constructing the dummy start and end activities j_s and j_e with $p_s = p_e = 0$ and no resource requirements. Then, an activity j_i is constructed for each action $a_i \in A$ such that the execution time p_i of j_i is $\lceil \frac{1}{t_{slot} a_{i,R}} \rceil$ where t_{slot} is the specified duration of a time slot and $a_{i,R}$ is the rate at which the action may be executed. We then set the resource requirements of j_i to be $h_{il} = 1$ for all $k_l \in Q(a_i)$.

Resource Reservation Since repeater protocols store entangled links in qubits it is necessary to make sure actions from other repeater protocols do not attempt to use qubits that hold a link. For each resource k_l used in the repeater protocol we construct a list of actions $A_{k_l} = \{a \in A | k_l \in Q(a)\}$ sorted by the starting offsets $M(a)$. For each pair of consecutive actions $a_i, a_j \in A_{k_l}$, we construct an *occupation activity* j_o with processing time $p_o = (M(a_j) - M(a_i) - p_i)$, resource requirement $h_{ol} = 1$ if k_l holds an entangled link after the execution of a_i. We then specify minimal and maximal time lags $d_{io} = \bar{d}_{io} = d_{oj} = \bar{d}_{oj} = 0$. The purpose of this occupation activity is to ensure that the resource remains occupied between actions of the protocol that produce a link and consume the same link. If the last activity $a_i = (a_{i,ID}, a_{i,V}, a_{i,F}, a_{i,R}) \in A_{k_l}$ has $a_{i,ID} \in \mathit{link}, \mathit{distill}$ and k_l holds a link after execution of action a_i, then we construct an additional occupation activity j_o with $p_o = \max_{a_j \in A}(M(a_j) + p_j) - M(a_i) - p_i$ to occupy the resource until the end of the protocol. Activities j_i and j_o are then associated with minimal and maximal time lags $d_{jo} = \bar{d}_{jo} = 0$. This occupation activity is then connected to the dummy end activity j_e with $d_{oe} = \bar{d}_{oe} = 0$.

Connecting Dummy Activities We finish the construction of the activity-on-node network by adding minimal and maximal time lags such that each activity node has a path to the dummy end. First, we set the minimal and maximal time lags d_{si} and $\bar{d}_{si}$ between the dummy start j_s and activity j_i as $M(a_i)$ for all activities j_i with corresponding action a_i that do not have a predecessor in the activity-on-node network. Next, we set minimal and maximal time lags d_{ie} and $\bar{d}_{ie}$ between activity j_i and dummy end j_e to be $\max_{a_j \in A}(M(a_j) + p_j) - M(a_i) - p_i$ for all activities j_i with corresponding action a_i that do not have a successor in the activity-on-node network. This produces an activity-on-node network for a concrete protocol $P(A, I, M, Q)$ which respects the protocol's non-preemptive nature.

Example 4.3.1. Before continuing to describe the construction of the activity-on-node network for the TDMA scheduling problem, we provide a simple example. Suppose we have a four node network as shown in figure 4.11a and we wish to perform the repeater protocol shown in figure 4.11b over the repeater chain (A, R, B). In this repeater protocol, L_1 represents an elementary link generation between A and R that requires two time slots while L_2 represents an elementary link generation between R and B that requires one time slot. S_1 is an entanglement swap that occurs at R and requires one time slot.

Figure 4.11c shows a mapping of the protocol to the node resources in the case when all network nodes have one communication qubit (C_0) and one storage qubit (S_0). The shaded regions represent periods of time where the entangled links are stored. The corresponding activity-on-node network for this concrete repeater protocol is shown in figure 4.11d and the set of resources is $K = \{A\text{-}C_0, A\text{-}S_0, R\text{-}C_0, R\text{-}S_0, B\text{-}C_0, B\text{-}S_0\}$. For convenience, we have labeled the activities in correspondence to their protocol actions and have labeled the occupation activities with O and the dummy start and end activities as S and E. Table 4.4 summarizes the execution time and resource requirements of the occupation activities in the network while the remaining activities have these parameters specified by the concrete protocol in figure 4.11c.

Activity	O_1	O_2	O_3	O_4	O_5
Proc. Time p	2	0	1	1	0
Res. Req.	$A\text{-}S_0$	$R\text{-}C_0$	$R\text{-}S_0$	$B\text{-}S_0$	$R\text{-}C_0$

Table 4.4: A description of the occupation activities in the activity-on-node network of figure 4.11d. Each occupation activity corresponds to a qubit resource that must be reserved between actions of the protocol.

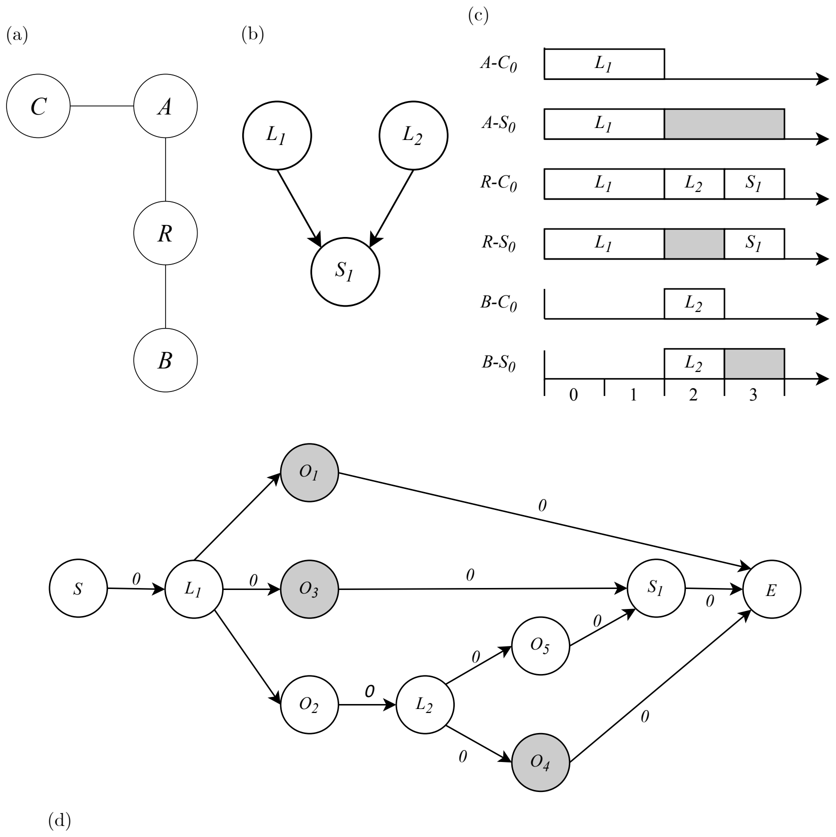


Figure 4.11: Construction of an activity-on-node network for a three-node repeater protocol. a) The network topology used for the example. b) A simplified representation of a repeater protocol to connect A and B. L_1 and L_2 are elementary link generations that take two time slots and one time slot respectively. S_1 is an entanglement swap that takes one time slot. c) A visual representation of the concrete repeater protocol constructed from the repeater protocol in b. d) The activity-on-node network corresponding to the concrete repeater protocol. S and E correspond to the dummy start and end activities while O_1, O_2, O_3, O_4 and O_5 correspond to occupation activites that reserve resources A-S_0, R-C_0, R-S_0, B-S_0, and R-C_0 respectively. O_2 and O_5 remain unshaded as they occupy the resource for 0 time units.

Full Activity-on-Node Network Construction Now that we have a method for constructing the activity-on-node network for a single concrete repeater protocol, we can construct the activity-on-node network for the TDMA scheduling problem in the following way. We begin by initializing the set of activities with a dummy start j_s. Using the demands $\mathcal{D}$ we compute a period $T_i = \lfloor \frac{1}{t_{slot} R_i} \rfloor$ for each demand D_i and the hyperperiod $H = lcm_{D_i \in \mathcal{D}}(T_i)$ similarly to what is done for periodic task scheduling.

Next, for each demand $D_i \in \mathcal{D}$ we construct and enumerate $\frac{H}{T_i}$ instances of the activity-on-node network for the concrete protocol P_i as outlined previously. For each instance $0 \leq l < \frac{H}{T_i}$ of the activity-on-node network, we set a minimal time lag $d_{s,ls} = lT_i$ between the dummy start j_s and the dummy start j_{ls} of the lth instance of the activity-on-node network. We then set a maximal time lag $\bar{d}_{0,lmax} = (l+1)T_i$ between the dummy start j_s and the dummy end j_{lmax} of the lth instance of the activity-on-node network. The minimal time lag $d_{s,ls}$ may be thought analogously to the release offset of a task instance in the periodic task scheduling case while the maximal time lag $\bar{d}_{s,lmax}$ corresponds to the deadline of a task instance.

We finish the construction of the activity-on-node network for the TDMA scheduling problem by adding a dummy end activity j_e and connecting all end activities from each activity-on-node network instance with no minimal and maximal time lag constraints. The complete set of activities J is thus the union of the set of activities J_{il} corresponding to each instance l of the activity-on-node network for protocol P_i with the additional dummy start and dummy end activities added. The set of resources K is the union of all required resources across all instances of the activity-on-node networks.

Example 4.3.2. To illustrate the TDMA schedule construction, consider the same network shown in the previous example. Let there be two demands D_1 and D_2 such that D_1 has source and destination A and B while D_2 has source and destination A and C. Suppose that the protocol in 4.11b corresponds to D_1 while a simple elementary link generation shown in 4.12a corresponds to D_2. Furthermore, suppose that the rate requirements dictate that P_1 is executed once every ten time slots while D_2 must be executed once every five time slots. Figure 4.12b shows the activity-on-node network for D_2 while figure 4.12c shows the activity-on-node network for the TDMA scheduling problem. In this figure, an edge has a single value when it represents both the minimal and maximal time lags while a pair of values x, y represent a minimal time lag of x and a maximal time lag of y. An edge labeled with "-" means that the corresponding time lag constraint is not set.

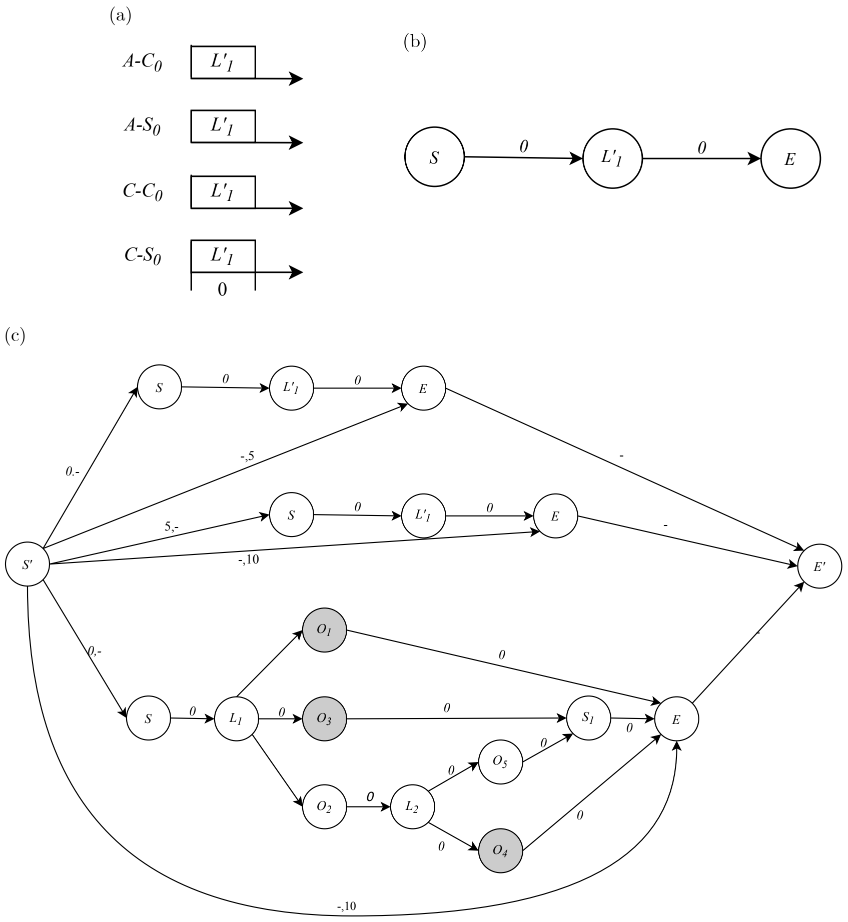


Figure 4.12: Construction of an activity-on-node network that represents the network demands and concrete repeater protocols to connect both pairs (A, C) and (A, B). a) A visualization of the concrete repeater protocol used to connect A and C using an elementary link generation. b) The activity-on-node network for the concrete repeater protocol between A and C. c) The activity-on-node network of the concrete repeater protocols connecting pairs (A, C) and (A, B) while also satisfying the demanded rates. The protocol between A and C appears twice while the protocol between A and B appears once due to the fact that the demanded rate between A and C is twice that between A and B.

This formulation can be thought of as performing periodic task scheduling across the network resources rather than a single uniprocessor shared amongst all nodes. Once a schedule has been computed, the starting times associated to the dummy jobs of each activity-on-node network corresponds to a starting time for the corresponding repeater protocol. Thus, if we can construct a schedule that satisfies the time-constrained RCPSP formulated above then we can construct a TDMA schedule for the repeater protocols by using the set of assigned start times.

Project Transformation - Full Protocol Reservation

The previous project construction contains activities corresponding to every single protocol action across all instances of repeater protocols to be executed. Thus the number of activities in this project formulation scales as $O(NS)$ where N is the number of activity-on-node network instances and S is the maximum number of actions in any repeater protocol. Our solver for the NPRP RCPSP uses an EDF heuristic for selecting activities to schedule and has complexity $O(|J|^2|K|\log|J|)$ where $|J|$ is the total number of activities in the project. If we directly use the formulation above we obtain a run-time complexity of $O(N^2S^2K\log(NS))$ where $|K|$ corresponds to the size of the set of resources used across all repeater protocols and S is the maximum number of actions in any concrete repeater protocol. In practice such a run-time complexity is restrictive of how frequently we can compute a new TDMA schedule for the network. Thus, we may desire a way to way to reduce complexity at the cost of schedule quality.

For the NPRP case, we consider a modification of the project construction described previously which reserves the protocol resources for the full duration of the protocol, regardless of whether the resources are idle or not at any given point in time. To do so, one can construct the activity-on-node network for a repeater protocol instance to be a three-node network consisting of a dummy start and dummy end, as well as a single node that has execution time $p = \max_{a_j \in A}(M(a_j) + \lceil \frac{1}{t_{slot} a_{j,R}} \rceil)$ and resource requirements $h_l = 1$ for each resource k_l used by the protocol. Doing so allows us to reduce the total number of nodes in the activity network to scale as $O(N)$ resulting in a run-time complexity of $O(N^2|K|\log(N)$. Though this resembles the non-preemptive periodic task scheduling approach, it overcomes the problem of scheduling repeater protocols on independent repeater chains when the path-vertex intersection graph results in a single connected component. The performance of this transformation is analyzed further in chapter 5.

4.3.2. LPRP TDMA Scheduling

Our approach for the LPRP TDMA scheduling problem is similar to that taken for the NPRP problem. Instead of an RCPSP, we formulate the problem as a multi-mode resource-constrained project scheduling problem (MRCPSP) where activities in the project support multiple different execution modes as discussed in chapter 2. Recall from the periodic task scheduling problem that here each protocol P_i additionally specifies a quantity $B_{i,delay}$ that represents the maximum cumulative delay that can be introduced in between steps of the protocol.

Project Construction

In order to obtain an MRCPSP that represents the LPRP TDMA scheduling problem we modify the procedure for constructing the project in the NPRP case as follows. For each protocol $P_i(A_i, I_i, M_i, Q_i)$ we convert $B_{i,delay}$ to a number of slots $B_i = \frac{B_{i,delay}}{t_{slot}}$. When constructing the occupation activities between two consecutive actions a_i, a_j using a resource k_l we modify the occupation activity j_o to have B_i execution modes such that each mode m, $0 \leq m \leq B_i$, specifies an execution time $p_{om} = p_o + m$ and resource requirement $h_{oml} = h_{ol}$. This ensures that if activity j_j is delayed then the occupation activity j_o uses a mode that fully occupies the resource between activities j_i and j_j. We then add a maximal time lag d_{se} between the dummy start and end activities j_s, j_e set to $\max_{a_j \in A_i}(M_i(a_j) + \lceil \frac{1}{t_{slot} a_{j,R}} \rceil) + B_i$ to make sure that the set of activities completes without violating the delay bound. It is not necessary to modify any other time lag constraints as we want the choice of modes for the occupation activities to guarantee reservation of the resources between the end and start of consecutive activities.

Example 4.3.3. Figure 4.13 shows the activity-on-node network construction for LPRP with the same repeater protocol shown in the NPRP example. Note that the only visual difference between the activity-on-node network shown in 4.13 and its NPRP counterpart is the addition of the edge from S to E with minimal and maximal time lags. Table 4.5 shows the additional mode specification for the occupation activites $O_1, ..., O_5$.

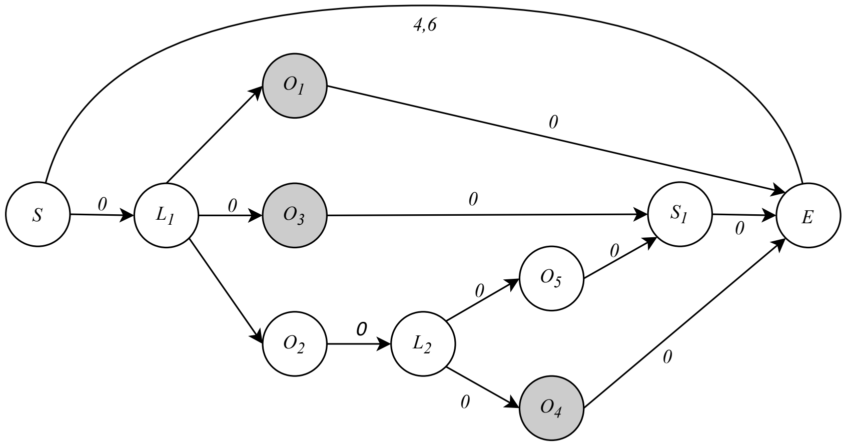


Figure 4.13: The activity-on-node network for the LPRP TDMA scheduling problem using MRCPSP. The activity-on-node network is augmented with a time constraint between the dummy start S and the dummy end E while occupation activities (labeled "O") are augmented with modes shown in table 4.5.

Activity	O_1	O_2	O_3	O_4	O_5
Proc. Time $m = 0$	2	0	1	1	0
Proc. Time $m = 1$	3	1	2	2	1
Proc. Time $m = 2$	4	2	3	3	2
Res. Req.	$A\text{-}S_0$	$R\text{-}C_0$	$R\text{-}S_0$	$B\text{-}S_0$	$R\text{-}C_0$

Table 4.5: A description of the occupation activities in the activity-on-node network of figure 4.13. Each occupation activity is augmented with three modes $m = 0, 1, 2$ that correspond to different durations of occupying the required resources.

Once the individual activity-on-node networks have been modified using the above mechanism, the remaining construction of the full TDMA problem is the same as for the NPRP case.

Project Transformation - Segmented Protocol Reservation

Similarly to the NPRP case, we can modify the procedure for activity-on-node network construction to reduce the complexity in constructing a schedule. Rather than constructing a single activity node to represent the full protocol, we can partition the protocol into activity segments and represent these as nodes. Each segment represents a time-slice of the concrete repeater protocol and the set of actions that occur within the time slice are chosen similarly to the limited preemption budget task scheduling formulation.

Segment Discovery For a given protocol $P_i(A_i, I_i, M_i, Q_i)$, we first construct time intervals (t_s, t_e) for each action $a \in A_i$ such that $t_s = M_i(a)$ and $t_e = M_i(a) + \lceil \frac{1}{t_{slot} a_R} \rceil$. Using an interval tree data structure, we then sort the set of intervals and merge overlapping intervals $(t_{s1}, t_{e1}), (t_{s2}, t_{e2})$ where $t_{s2} < t_{e1}$ into single intervals (t_{s1}, t_{e2}) while tracking the set of actions that overlap. Once no more intervals overlap, we group the protocol's actions into action segments that represent the set of actions occurring within the time time interval (t_s, t_e). This approach to determining segments is similar to what is done in the LPRP TDMA scheduling for the periodic task formulation.

Activity Construction For each set of actions A_{seg} in a segment seg we construct an activity j_{seg} for the activity-on-node network and set its processing time $p_{seg} = \max_{a \in A_{seg}}(M_i(a) + \lceil \frac{1}{t_{slot} a_R} \rceil) - \min_{a \in A_{seg}}(M_i(a))$ and set its resource requirements as the union of the resources used by the contained protocol actions. We additionally construct an occupation activity j_o between each consecutive pair of activity segments j_j, j_l with B_i modes such that the processing time of mode m is $p_m = (t_{s2} - t_{e1}) + m$. Here, t_{e1} corresponds to the completion time $M_i(a_1) + \lceil \frac{1}{a_{1,R} t_{slot}} \rceil$ of the latest action a_1 in the first segment while t_{s2} corresponds to the start time $M(a_2)$ of the earliest action a_2 in the second segment. We then set the minimal and maximal time lags $d_{j_i, j_o} = \bar{d}_{j_i, j_o} = d_{j_o, j_j} = \bar{d}_{j_o, j_j} = 0$.

Resource Reservation Next, we iterate over the set of resources and find the occupation intervals similarly to the project construction for NPRP. For each occupation interval (t_s^o, t_e^o) corresponding to a resource k_l we find the set of segment actions with intervals (t_s^i, t_e^i) that overlap with the occupation interval and modify the resource requirements of the corresponding activity in the network to include k_l and additionally update the preceding occupation activity's resource requirements to include k_l whenever $t_s^o \leq t_s^i$. We finish construction of the activity-on-node network by adding the dummy start and end activities j_s, j_e and set $d_{s,1} = \bar{d}_{s,1} = d_{n,e} = \bar{d}_{n,e} = 0$ and $\bar{d}_{s,e} = \max_{a_j j \in A_i}(M_i(a_j) + \lceil \frac{1}{t_{slot} a_{j,R}} \rceil) + B_i$. Here j_1 and j_n correspond to the first and last segments of the repeater protocol respectively.

This construction is mostly beneficial in the case of repeater protocols over longer repeater chains that perform protocol actions in parallel. If this technique is applied to a repeater protocol where all actions occur in serial fashion, the processing of the TDMA scheduling problem remains $O((NS)^2|K|\log(NS))$. This is because none of the time intervals where protocol actions occur will overlap and we will create a segment activity node for each action in the protocol. The benefits of using this formulation are further explored in chapter 5.

References

[1] T. Bacinoglu, B. Gulbahar, and O. B. Akan, *Constant fidelity entanglement flow in quantum communication networks,* in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010* (IEEE, 2010) pp. 1–5.

[2] R. F. Werner, *Quantum states with einstein-podolsky-rosen correlations admitting a hidden-variable model,* Physical Review A **40**, 4277 (1989).

[3] W. J. Munro, K. Azuma, K. Tamaki, and K. Nemoto, *Inside quantum repeaters,* IEEE Journal of Selected Topics in Quantum Electronics **21**, 78 (2015).

[4] H.-J. Briegel, W. Dür, J. I. Cirac, and P. Zoller, *Quantum repeaters: the role of imperfect local operations in quantum communication,* Physical Review Letters **81**, 5932 (1998).

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

[6] W. Dür and H.-J. Briegel, *Entanglement purification for quantum computation,* Physical review letters **90**, 067901 (2003).

[7] C. Ekelin, *Clairvoyant non-preemptive edf scheduling,* in *18th Euromicro Conference on Real-Time Systems (ECRTS'06)* (IEEE, 2006) pp. 7–pp.

[8] C. L. Liu and J. W. Layland, *Scheduling algorithms for multiprogramming in a hard-real-time environment,* Journal of the ACM (JACM) **20**, 46 (1973).

[9] Y. Wang and M. Saksena, *Scheduling fixed-priority tasks with preemption threshold,* in *Proceedings Sixth International Conference on Real-Time Computing Systems and Applications. RTCSA'99 (Cat. No. PR00306)* (IEEE, 1999) pp. 328–335.

[10] S. Baruah, *The limited-preemption uniprocessor scheduling of sporadic task systems,* in *17th euromicro conference on real-time systems (ECRTS'05)* (IEEE, 2005) pp. 137–144.

[11] A. Burns, *Preemptive priority based scheduling: An appropriate engineering approach* (Citeseer, 1993).

5 Analysis and Comparison

The choice of scheduling technique for constructing the TDMA schedules of a quantum network has strong implications for the performance of the network. In chapter 4 we showed two novel formulations for constructing TDMA schedules of repeater protocols using periodic task scheduling and resource-constrained project scheduling. We saw that the complexity of using resource-constrained project scheduling is higher but offers an opportunity to extract parallelism from the network. Here, we perform the first evaluation and comparison of these scheduling techniques by simulating the construction of TDMA schedules on several network topologies.

In this chapter we begin by analyzing the impact of various network parameters on the concrete repeater protocols in our simulation. We then show how the scheduling techniques presented in chapter 4 compare to one another on three different network topologies for the NPRP and LPRP scheduling problems. Finally, we present simulation results that show the impact of resource allocation in the network on the achievable performance of a network using TDMA scheduling. This is the first thorough analysis of the effects of network parameters on TDMA scheduling systems for quantum networks. Code and data for our analysis are available upon request.

5.1. Concrete Protocol Analysis

In order to motivate the choice of network parameters used for analyzing the different scheduling heuristics presented in chapter 4, it is necessary to first inspect how properties of a quantum network affect the behavior of concrete repeater protocols. Here, we will present an analysis of the effect of the slot size, the elementary link length, the repeater chain length, and the number of network node resources on the latency of a repeater protocol.

Slot Size Selection Recall that time is partitioned into fixed duration slots when using time-division multiple access to multiplex a shared transmission medium. The choice of slot size is an important parameter in representing a TDMA schedule as it has an impact on the latency of a repeater protocol as well as the amount of data necessary to represent the schedule. Choosing a slot size that is longer than the expected time to generate an elementary link will result in storing the link for a longer period of time if the link is generated early. When this happens the fidelity of the generated link will decrease due to decoherence it experiences in storage. On the other hand, choosing a slot size that is too small will mean that the action of generating an elementary link will need to occupy multiple slots, thus potentially increasing the amount of data necessary to represent the schedule.

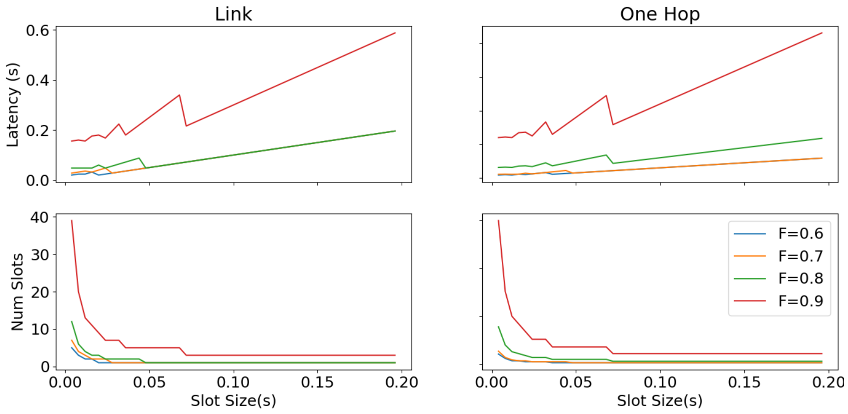


Figure 5.1: Plot of the latency of a repeater protocol over a repeater chain of five nodes for different minimum fidelity F as a function of the slot size. Each node along the repeater chain has one communication qubit and five storage qubits.

The effect of slot size on repeater protocol latency for directly connected nodes and a one-hop repeater protocol can be seen in figure 5.1. Here, we have generated repeater protocols using several choices of minimum fidelity and mapped them into concrete repeater protocols using an increasing choice of slot size when the node resources are one communication qubit and five storage qubits.

First, we see that for the directly connected link that the latency of a repeater protocol increases greatly when changing the fidelity requirement from 0.8 to 0.9. This is due to the fact that the highest fidelity achievable from an elementary link is 0.87 with our used link capabilities. This means that repeater protocols need to use entanglement distillation to achieve the required fidelity. Since there is only a single communication qubit, elementary links need to be generated serially which results in an increased protocol latency.

We also see that sweeping the choice of slot size results in a saw-tooth shaped plot of the repeater protocol latency. Recall from section 4.1.1 that the duration of a concrete protocol action a in slots is $\lceil \frac{1}{a_R t_{slot}} \rceil$. That is, actions occupy an integer number of time slots that depends on the choice of slot size. When actions have a lower rate, they require more time slots than higher rate actions. Because this value needs to be an integer, some choices of slot size result in over-allocation of time to actions. The dips in latency occur when the expected duration of a protocol action is an integer multiple of the slot size, resulting in no wasted time. As the slot size increases beyond this point, there is an overallocation of time to actions which increases the protocol latency.

Eventually, when the slot size is beyond the duration of any protocol action, the latency of the protocol increases indefinitely as additional idle time is introduced to each action. This can be seen in the bottom plots that show the required number of slots to execute the protocol. We see that as the slot sizes increases this number reaches a lower limit where a protocol action only occupies a single time slot. The exponential decrease in required number of slots is due to the fact that the required number of slots for all of the actions decreases as the slot size increases.

Link Length We now show the effect of link length on the achievable rate of a repeater protocol. Recall from chapter 2 that the rate of generating an elementary link decreases exponentially with the link length due to losses of photons in optical fiber. We demonstrate the effect this has by fixing the repeater chain to be three nodes composed of a source,

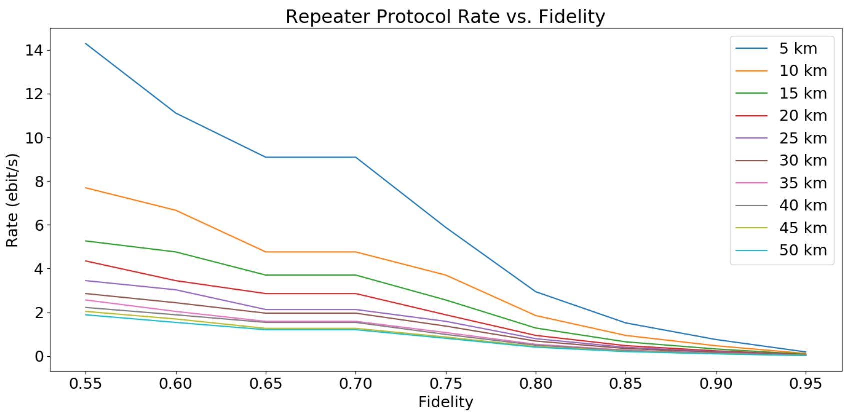


Figure 5.2: Plot of the achievable rate vs. minimum fidelity for different elementary link lengths over a three-node repeater chain. Each node has one communication qubit and five storage qubits.

repeater, and destination and fix the node resources to be one communication qubit and five storage qubits as in the slot size analysis. We then modify the link lengths between the nodes and use our concrete protocol generation scheme to find concrete repeater protocols for these configurations.

Figure 5.2 shows that the rate of a repeater protocol generally drops exponentially as the required fidelity increases. If we fix the fidelity, we also see that increasing the link length from 5 km to 10 km observes a larger reduction in rate than increasing from 10 km to 15 km and so on. This observation agrees with how the rate of elementary link generation decreases exponentially with the length of the link.

Repeater Chain Length The repeater chain length has a similar effect to the elementary link length. Recall that entanglement swapping must be performed at each node along a multi-hop repeater protocol in order to deliver entanglement to the end nodes. Furthermore, the fidelity of a swapped link scales quadratically with the fidelity of the links used for entanglement swapping. Thus when the repeater chain length increases it is necessary to perform additional entanglement distillation to compensate for the effects of entanglement swapping on end-to-end fidelity.

The increase in protocol latency is confirmed by figure 5.3 where we show the effect of increasing the repeater chain length. Here, we vary the length of the repeater chain and fix the elementary link lengths to be 5 km. Each node has one communication qubit and five storage qubits and we use our concrete protocol generation scheme to evaluate the achievable rate for different minimum fidelity. Compared to figure 5.2 we see a similar effect on the rate when fixing the chain length and increasing the minimum fidelity. Specifically, we see that in both of these plots the blue curve corresponds to a three node repeater chain with 5 km elementary links.

If we now consider a five node repeater chain with 5 km elementary links the end-to-end distance is 20 km. Compared to an elementary link of 20 km long we see that the achievable rate of the five node repeater chain is much lower even though the elementary link rates should be much higher. This is the result of additional latency from generating additional links to use for entanglement distillation in order to meet the minimum fidelity.

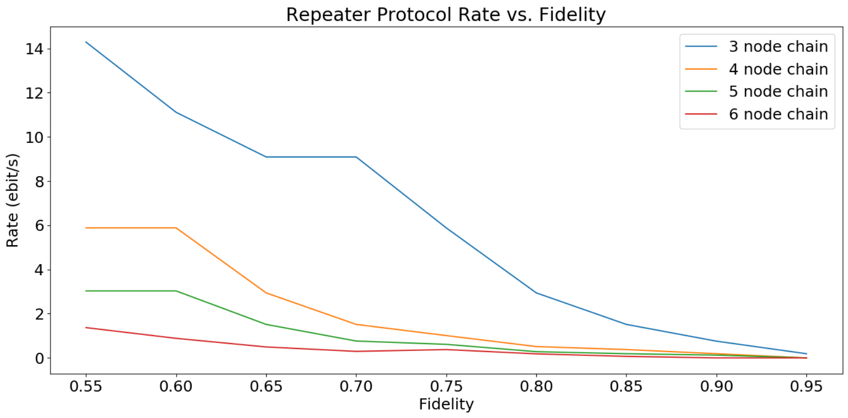


Figure 5.3: Plot of the achievable rate vs. minimum fidelity for different repeater chain lengths when the elementary link length is fixed to be 5 km. Each node has one communication qubit and five storage qubits.

Nodal Resources We conclude the protocol analysis by showing how node resources impact the rate of a repeater protocol. With additional communication qubits, directly connected nodes may generate elementary links in parallel, reducing the latency of the protocol. Furthermore, when the number of storage qubits increases the nodes may perform higher levels of nested entanglement distillation at the elementary link level, reducing the effects of entanglement swapping and thereby reducing the number of required entanglement distillations.

Figure 5.4 shows the achievable rate for various combinations of communication qubit count and storage qubit count in the network nodes over a three node repeater chain while figure 5.5 shows these results for a four node chain. While the achievable rate for the three node chain increases significantly when moving from one communication qubit to two, there seems to be no affect when increasing to four communication qubits. On the other hand, the four node chain sees an additional benefit when moving from two communication qubits to four communication qubits.

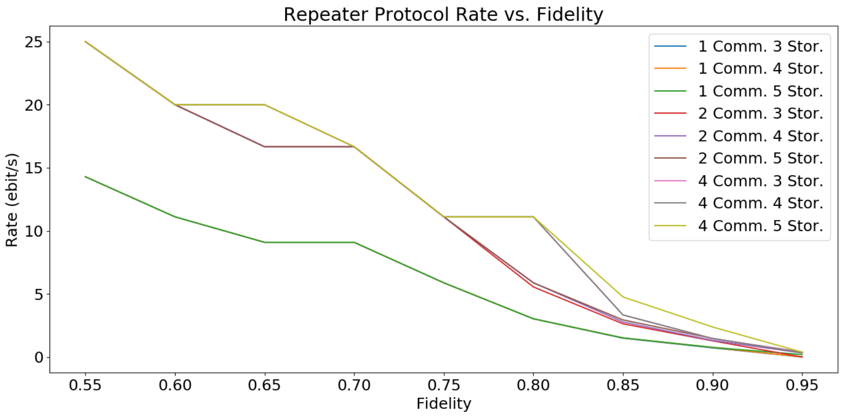


Figure 5.4: Plot of the achievable rate vs. minimum fidelity for different numbers of node resources over a repeater chain of four nodes and elementary link length of 5 km.

When the chain length is lower there are fewer entanglement swaps performed and fewer entanglement distillations are needed to meet the minimum end-to-end fidelity. This reduces the number of required elementary links.

Interestingly, figure 5.4 shows that increasing the communication qubits allows for a three node repeater protocol that achieves a higher rate than those seen for 5 km distance in figures 5.2 and 5.3. When the number of communication qubits increases, the latency of the repeater protocol can be reduced if the elementary links are generated using link capabilities with higher rate but lower elementary link fidelity. With more communication qubits, the lower fidelity links can be generated in parallel and

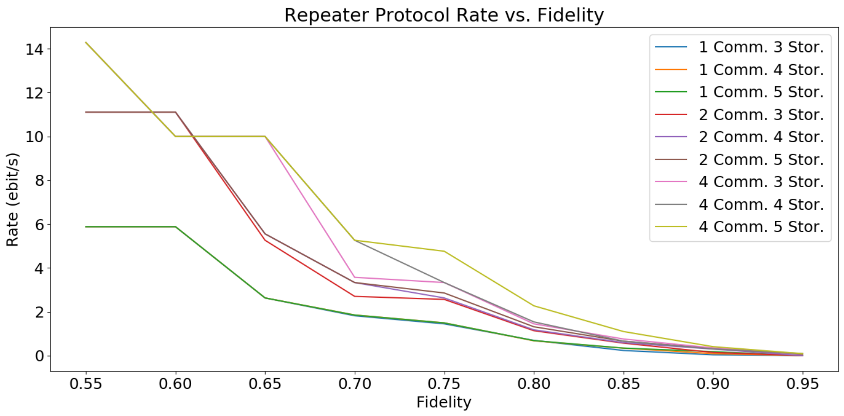


Figure 5.5: Plot of the achievable rate vs. minimum fidelity for different numbers of node resources over a repeater chain of four nodes and elementary link length of 5 km.

distilled into a higher fidelity link faster than a single higher fidelity link may be generated.

5.2. Scheduling Technique Comparison

With an idea of how various network parameters influence repeater protocol latency we may now proceed to evaluate techniques for constructing the TDMA schedule. In this section we compare the several scheduling strategies discussed in chapter 4 to see how different Quality of Service metrics may be affected. By showing how the schedulers differ in these ways a proper strategy may be used to meet the desired behavior of the network. Specifically, we show how the protocols behave when node resources are under heavy congestion and observe how the different scheduling methods affect overall network throughput as well as the worst case response time and inter-delivery jitter.

Network Topologies To evaluate our schedulers, we used three different network topologies as shown in figure 5.6 along with the network parameters specified in table 5.1. Our choice for these topologies is motivated by our desire to observe scheduler performance under network congestion. In the case of the star topology of figure 5.6a, the central node is a repeater while the all other nodes are end nodes. In this situation any repeater protocol connecting two nodes must cross a common repeater, resulting in high resource contention. The H topology in figure 5.6b reduces congestion when demands connect nodes vertically and the line topology experiences congestion when the repeater protocols overlap.

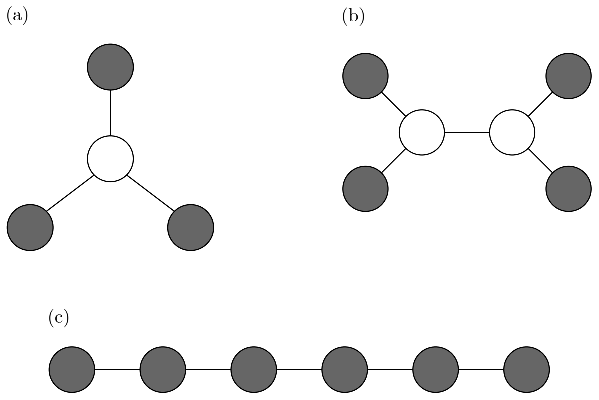


Figure 5.6: Network topologies used for evaluating TDMA scheduling techniques. Grey nodes represent end nodes and white nodes are repeater nodes. Edges represented directly connected nodes. a) star topology. b) H topology. c) Line topology. Repeater protocols connecting end nodes that share nodes between their repeater chains are said to ”cross” at these shared nodes.

Link Length	5 km
# Communication Qubits	1
# Storage Qubits	3
Time Slot Size	10 ms

Table 5.1: Simulation network parameters.

Demand Generation Our procedure for generating network demands is as follows. We first fix the fidelity requirement of a set of demands to one specified in table 5.2 and then randomly select end nodes in the network as source and destination pairs. Once a protocol has been constructed that meets the fidelity requirement, we randomly assign a rate from the set shown in table 5.2 such that the rate at which the protocol can be executed is larger than the demand rate. If the rate of the demand was higher than the protocol could achieve, it would not be possible to satisfy the demand regardless of the scheduling decisions. We select these rates in order to have a smaller hyperperiod for the constructed task sets. With these parameters the longest schedule is 512 slots or 5.12 s using a slot size of 10 ms.

Fidelity	$0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9$
Rate ($\frac{ebit}{s}$)	12.5, 6.25, 3.125, 1.5625, 0.78125, 0.390625, 0.1953125

Table 5.2: Possible QoS requirements for demands used in simulations.

For each fixed fidelity and topology we generated 100 sets of demands such that each set of demands has a summed resource utilization of 1.5 for the communication qubit resources at the end nodes. This choice was made to make sure the full set of demands and their protocols could not be scheduled. This allows us to highlight noticeable differences between the schedulers when not all demands can be satisfied.

We will first showcase the results of the NPRP TDMA scheduling techniques before using them as a baseline for the LPRP TDMA scheduling techniques.

5.2.1. NPRP TDMA Scheduling

For the non-preemptive repeater protocol TDMA scheduling problem we simulated schedule construction using five different scheduling techniques. Two of these scheduling techniques are from the periodic task scheduling formulation and use the non-preemptive EDF heuristic (labeled PTS-NP-EDF) as well as the CEDF heuristic (labeled PTS-CEDF) in [1]. The other three schedulers are from the RCPSP formulation and use the original project construction (labeled RCPSP-NP-EDF), the full protocol reservation transformation (labeled RCPSP-FPR), and a CEDF heuristics for the RCPSP problem (RCPSP-CEDF).

Since the full set of demands is not scheduleable due to overutilization of resources we try to fulfill as many as possible by attempting to add them to a running set of satisfied demands one-by-one. More concretely, we start by attempting to the schedule the first demand alone, then add the second if they may be scheduled together or reject it if it's not possible. We then attempt to add as many demands as possible to the TDMA schedule.

Network Throughput

We begin by first presenting a comparison of the average network throughput over 100 simulations for each demand fidelity and topology. Achieving high network throughput is desirable in quantum networks as this allows supporting applications that demand high entanglement rates. It additionally allows supporting more users with limited network resources. From our simulations we extract the average achieved throughput for each of our topologies and also compute the average change in the throughput relative to a baseline.

The plot in figure 5.7 shows the average network throughput of the TDMA schedules for the star topology in figure 5.6a. The error bars in this plot correspond to the standard deviation of the achieved network throughput across the simulations. We see that the achieved throughput of demands with higher fidelity requirement observe lower variance than those with lower fidelity requirement. This behavior may be attributed to the higher latency of repeater protocols when meeting higher minimum fidelity. Since the minimum rates are fixed, the resource utilization of higher fidelity repeater protocols will be higher which reduces the number of protocols that can be scheduled together compared to lower fidelity requirements.

The average network throughput is not expected to change much due to the fact that all repeater protocols cross the central repeater, resulting in a congestion of the resources. For the periodic task schedulers, attempting to split the set of demands into independent task sets has no effect as they all overlap at the repeater. It appears that the the more considerate decision making provided in the CEDF schedulers is able to improve the throughput of the network.

Using PTS-NP-EDF and RCPSP-NP-EDF as baselines for the periodic task schedulers and RCPSP schedulers respectively, we also plot the average change in network throughput in figure 5.8. This quantity is computed by finding the amount of throughput change per simulation and averaging the change over all simulations. We can see that for lower fidelity requirements the RCPSP-CEDF and RCPSP-NP-FPR heuristics observe an increase in the average throughput change while a decrease is observed for higher fidelity requirement compared to the baseline. The PTS-CEDF scheduler performs just as well if not better than the PTS-NP-EDF scheduler.

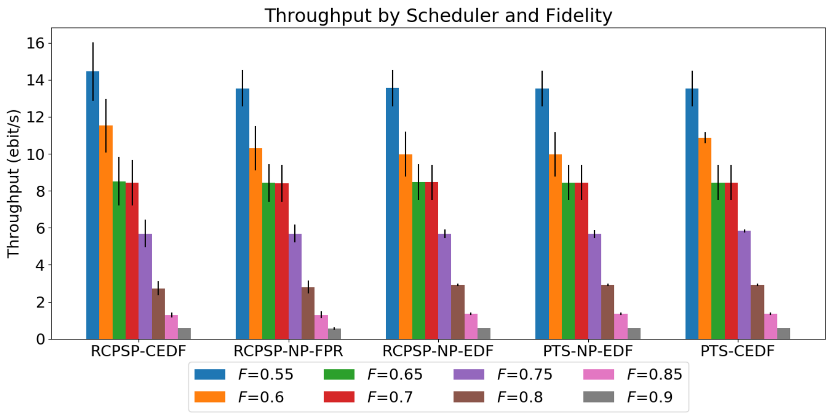


Figure 5.7: Average network throughput (over 100 simulations) of NPRP scheduling techniques on star topology.

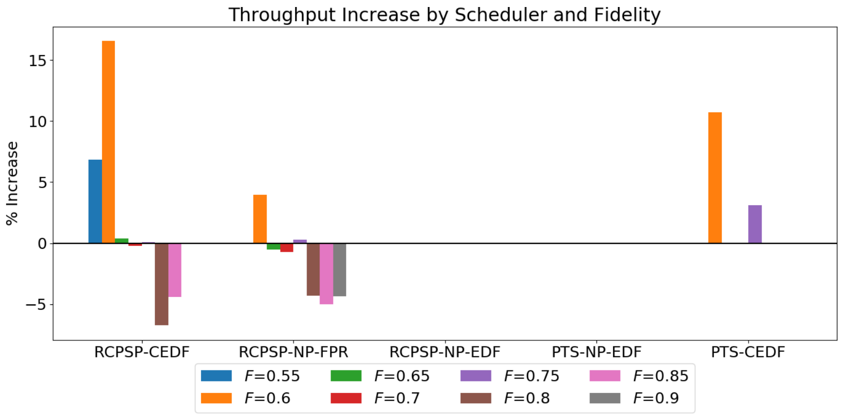


Figure 5.8: Average network throughput increase (over 100 simulations) of NPRP scheduling techniques on star topology. PTS-NP-EDF is the baseline for PTS-* schedulers while RCPSP-NP-EDF is the baseline for RCPSP-* schedulers.

We contrast these results with those in figures 5.9 and 5.10 for the H topology. In the H topology, the repeater protocols used to connect nodes vertically do not use both repeater nodes and can be executed in parallel. As a result, there is a notable difference between the achievable throughput when using an RCPSP scheduler over a periodic task scheduler for lower fidelity requirements. For higher fidelity requirements this difference is less noticeable as for the same reasons we described previously.

Similarly to before, we see that the RCPSP-NP-EDF scheduler observes a higher network throughput for higher fidelity requirements, though here we see that the increase for lower fidelity requirements is much greater than before. Specifically, we saw a reduction in average throughput change for fidelity requirements of 0.65 in the star topology while we now see an increase of approximately 22% and 8% for the RCPSP-CEDF and RCPSP-NP-FPR schedulers respectively. A fidelity requirement of 0.6 observes a similar change in network throughput while a requirement of 0.55 was greatly increased. We also see that PTS-CEDF achieves a higher network throughput compared to PTS-NP-EDF more consistently for various fidelity requirements.

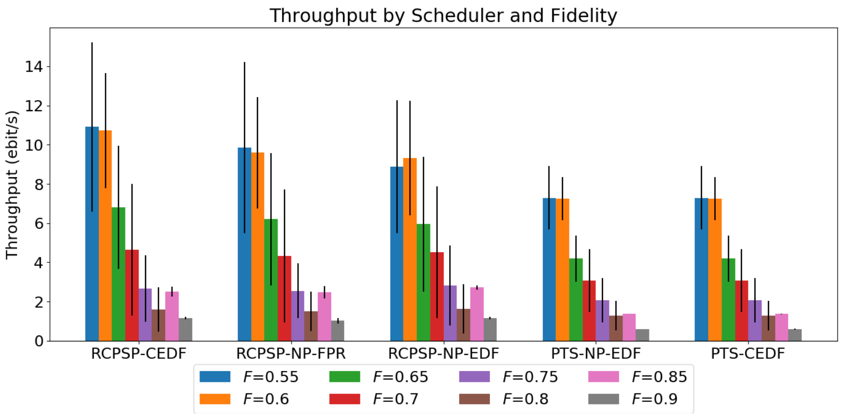


Figure 5.9: Average network throughput (over 100 simulations) of NPRP scheduling techniques on H topology.

Even though there are opportunities to execute repeater protocols in parallel we see that the average network throughput is lower than that in the star topology. Furthermore, the error bars show that there is higher variance in achieved throughput. This may be attributed to repeater

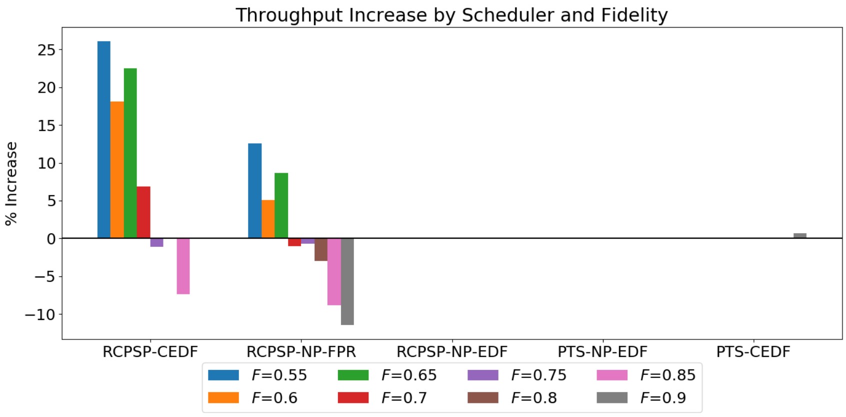


Figure 5.10: Average network throughput increase (over 100 simulations) of NPRP scheduling techniques on H topology. PTS-NP-EDF is the baseline for PTS-* schedulers while RCPSP-NP-EDF is the baseline for RCPSP-* schedulers.

protocols that utilize both repeater nodes. As we saw before, increasing the repeater chain length increases the latency of a protocol. This causes the repeater resources to be occupied for longer periods of time, reducing the throughput of the network even when there are protocols that may be executed in parallel.

A comparison of the network throughput for the NPRP TDMA schedulers on the line topology can be seen in 5.11. In this topology, many demands may be independent from one another or conflict with one another depending on the choice of end nodes. For example, a demand between end nodes at either end of the line will conflict with any other demand in the network, whereas demands between directly connected nodes only overlap with demands that use either node in their repeater chain. We see here that PTS-CEDF improves upon PTS-NP-EDF for all fidelity requirements whereas none of the RCPSP schedulers outperform one another for all fidelities. For example, the average throughput for $F \in \{0.55, 0.65, 0.7, 0.75, 0.8\}$ are higher for the RCPSP-NP-EDF scheduler whereas RCPSP-CEDF performs best for $F \in \{0.6, 0.9\}$ and RCPSP-NP-FPR performs best for $F = 0.85$.

When comparing the average network throughput for each topology in figures 5.7, 5.9, and 5.11 we see that the size of the error bars decreases as we introduce additional repeater protocols that can be performed in parallel. This suggests that highly congested components of a network observe lower variance in network throughput.

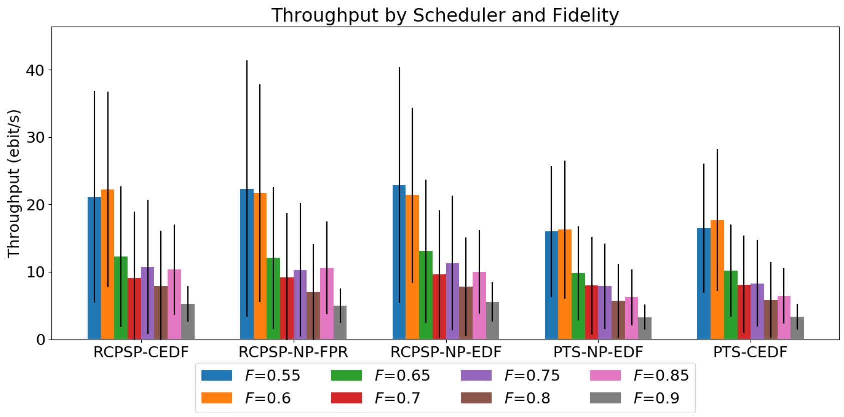


Figure 5.11: Average network throughput (over 100 simulations) of NPRP scheduling techniques on line topology.

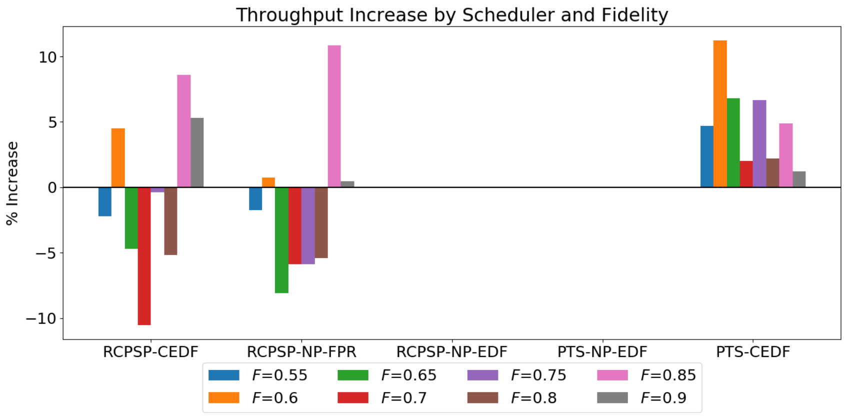


Figure 5.12: Average network throughput increase (over 100 simulations) of NPRP scheduling techniques on line topology. PTS-NP-EDF is the baseline for PTS-* schedulers while RCPSP-NP-EDF is the baseline for RCPSP-* schedulers.

5.2.2. Worst-case Response Time

Next, we analyze the observed worst-case response time under several NPRP scheduling heuristics. Recall from chapter 2 that the worst-case response time is the largest amount of time that passes from the release time of a task to the moment it finishes. The average worst-case response time has an impact on the observed inter-delivery jitter if some instances of a repeater protocol in the schedule have a large time-gap between one another. This has implications for applications that require a consistent rate of entanglement delivery as well as specific applications that require multiple entangled links to be present at their device simultaneously.

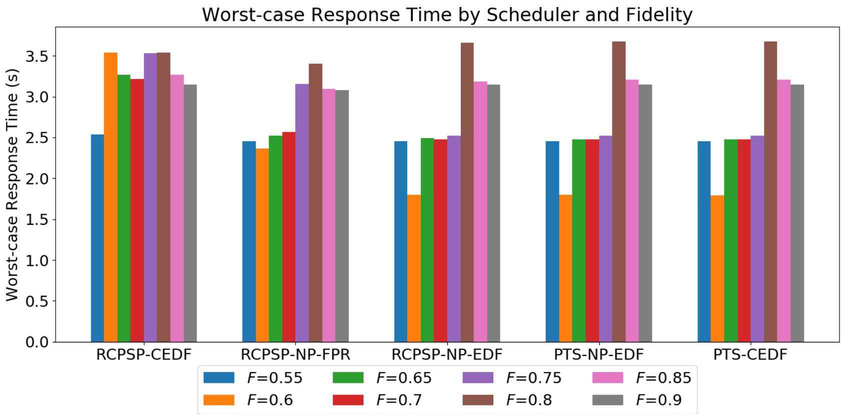


Figure 5.13: Average network throughput (over 100 simulations) of NPRP scheduling techniques on star topology.

Figure 5.13 shows a plot of the average worst-case response time for the star topology. Since the hyperperiod of the TDMA schedule is upper bounded by 5.12 s we see that the worst-case response time of the schedulers does not exceed this quantity. A common trend among some of the schedulers is that the average worst-case response time increases as the fidelity requirement increases. Since the latency of a repeater prootcol increases with the demanded fidelity we can expected this type of behavior especially in the case of high resource contention. We see that the periodic task schedulers and RCPSP-NP-EDF observe the same average worst-case response time for all fidelity requirements whereas RCPSP-NP-FPR and RCPSP-CEDF observe an increase in average WCRT for lower fidelities compared to the other schedulers as seen in figure 5.14. This suggests that the average worst-case response is not directly affected by the achieved network throughput.

Figure 5.15 shows the average worst-case response time on the H topology. Here we do not see the same increase in worst-case response

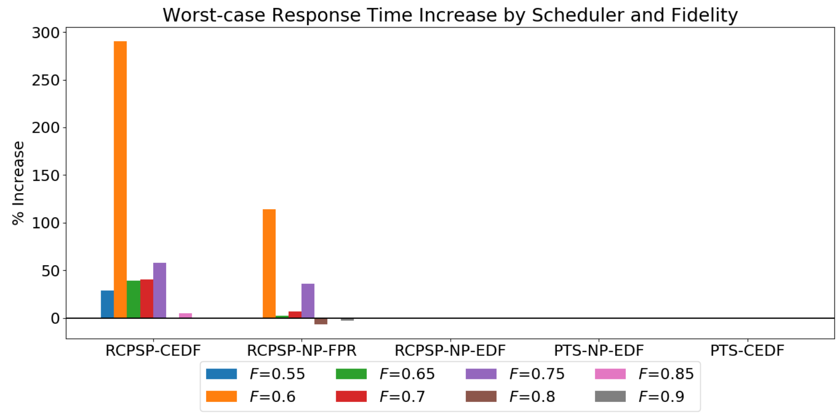


Figure 5.14: Average network throughput increase (over 100 simulations) of NPRP scheduling techniques on the star topology. PTS-NP-EDF is the baseline for PTS-* schedulers while RCPSP-NP-EDF is the baseline for RCPSP-* schedulers.

time for higher fidelity demands and in fact we see that for RCPSP-CEDF that the average worst-case response time was higher for lower fidelity requirements. In contrast, figure 5.17 shows a notable increase in worst-case response time when the fidelity requirement is increased like in the star topology.

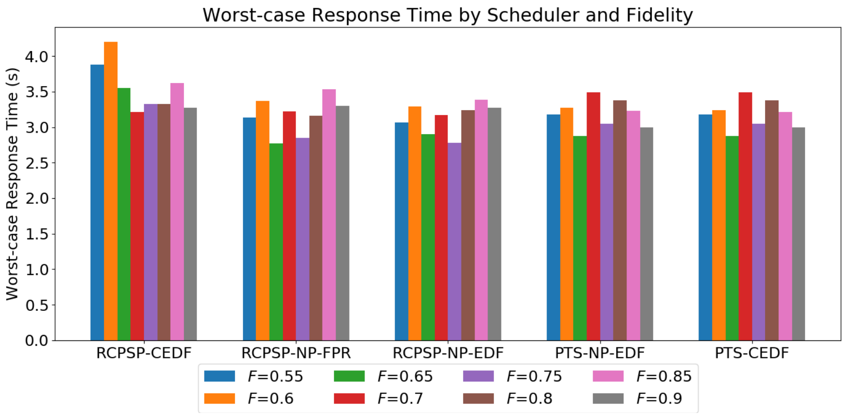


Figure 5.15: Average network throughput (over 100 simulations) of NPRP scheduling techniques on H topology.

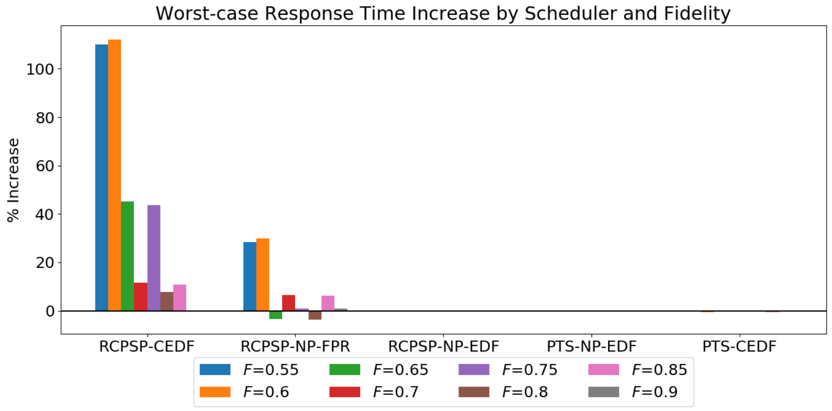


Figure 5.16: Average network throughput increase (over 100 simulations) of NPRP scheduling techniques on the H topology. PTS-NP-EDF is the baseline for PTS-* schedulers while RCPSP-NP-EDF is the baseline for RCPSP-* schedulers.

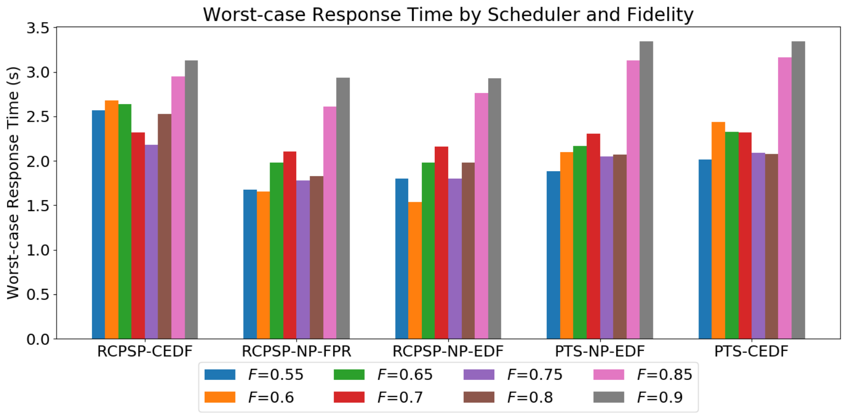


Figure 5.17: Average network throughput (over 100 simulations) of NPRP scheduling techniques on line topology.

We can see that moving from the star topology with high congestion to the line topology shows larger differences in the average change of worst-case response time. In the star topology, the RCPSP-CEDF scheduler observed average worst-case response time change less than 50% for many fidelity requirements while on the line topology the majority of fidelity requirements exceeded a 50% change for RCPSP-CEDF.

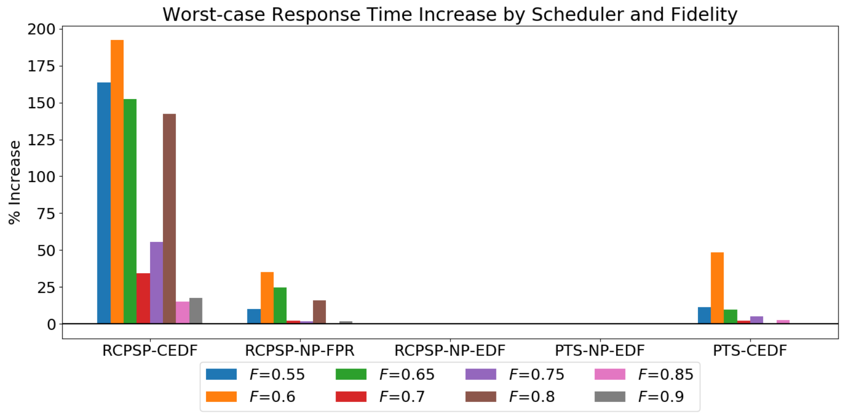


Figure 5.18: Average network throughput increase (over 100 simulations) of NPRP scheduling techniques on the line topology. PTS-NP-EDF is the baseline for PTS-* schedulers while RCPSP-NP-EDF is the baseline for RCPSP-* schedulers.

Jitter

We conclude the comparison of NPRP TDMA scheduling techniques by comparing their observed jitter in delivery. Jitter is the variance in inter-delivery times of entanglement and corresponds to the amount of time that spans between two consecutive executions of a repeater protocol. Similarly to worst-case response time, low jitter is important to achieving regular delivery of entanglement to applications. The difference between these two quantities is that low jitter can be observed if the worst-case response time is consistently high while high jitter is observed when the response time of repeater protocols changes frequently. While our method for TDMA schedule construction satisfies the rate requirements for the whole duration of the schedule, there are no inter-delivery constraints imposed in our methods. Thus it is useful to see what type of behavior the schedulers demonstrate without these explicit constraints.

Figure 5.19 shows the average jitter observed on the star topology. While the average throughput for these scheduling techniques was comparable, we see that periodic task schedulers observe lower jitter than the RCPSP schedulers. Most notably, we see that the RCPSP-CEDF and RCPSP-NP-FRP schedulers observe much higher jitter.

The high jitter seen in RCPSP-CEDF may be due to its decisions to delay protocols in an effort to satisfy deadline constraints. When high frequency demands are mixed with low frequency demands we may expect greater variation in this jitter as the high frequency demands may or may not be delayed by RCPSP-CEDF's decisions. This will cause the inter-delivery times of entanglement to vary. As for RCPSP-NP-FPR, the over-reservation of network resources in the case when they are not needed may cause additional delays of protocols as compared to RCPSP-NP-EDF which only reserves resources for the actions they are required.

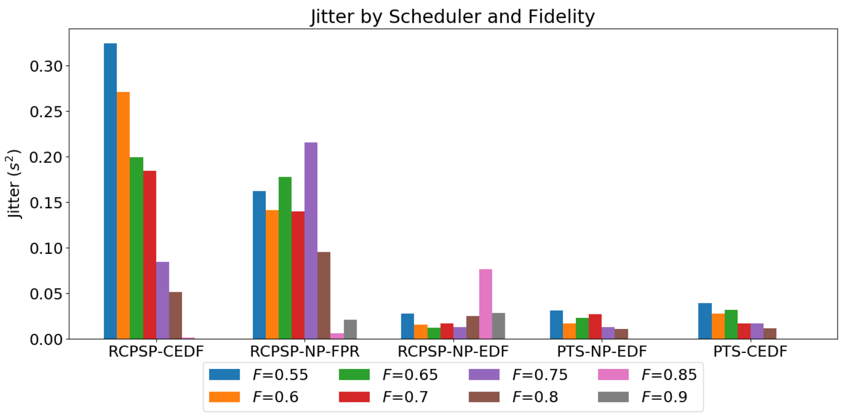


Figure 5.19: Average jitter (over 100 simulations) of NPRP scheduling techniques on the star topology.

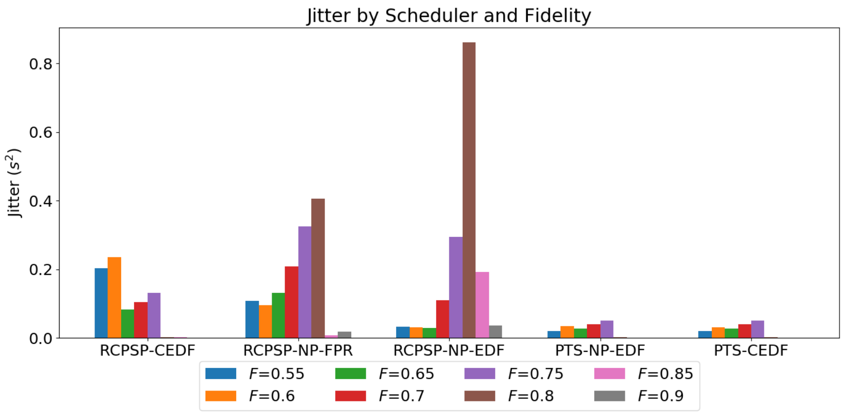


Figure 5.20: Average jitter (over 100 simulations) of NPRP scheduling techniques on the H topology.

We now contrast these results with those seen for the H topology in figure 5.20. We see that the jitter observed in RCPSP-NP-EDF for fidelity requirements of 0.8, 0.85, and 0.9 is much greater than what is seen for RCPSP-NP-FPR or RCPSP-CEDF. As we saw from the achievable throughput, the set of satisfied demands between these two schedulers differ and the set of demands may impact the behavior of inter-delivery times in the constructed schedules. Compared to the star topology, RCPSP-CEDF appears to experience less jitter when there is less contention on the repeater resources. The jitter observed in the periodic task schedulers remains low between these two topologies.

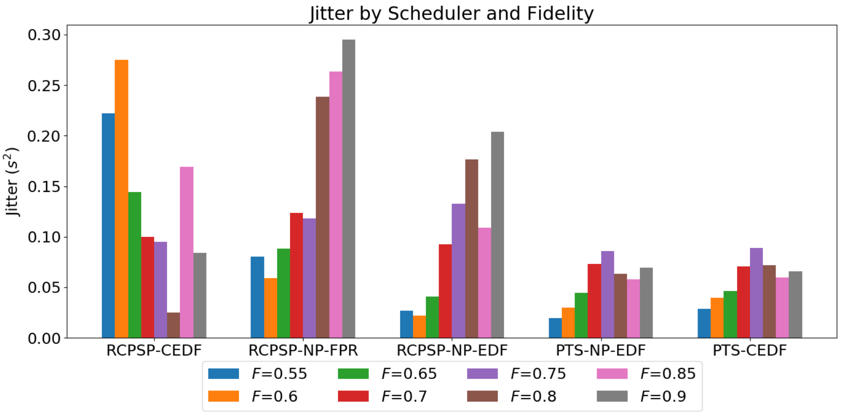


Figure 5.21: Average jitter (over 100 simulations) of NPRP scheduling techniques on the line topology.

We finish the discussion of jitter comparison by observing the behavior of the schedulers on the line topology as shown in figure 5.21. We see that compared to the star and H topologies that the periodic task schedulers observe an increase in their observed jitter. This may be explained by the fact that separating demands based on the path-vertex intersection graph of their protocols has no effect on the set of protocols to schedule together. As a result, a protocol that could be executed in parallel is delayed and observes greater variance in inter-delivery times. For the RCPSP schedulers we see that for RCPSP-CEDF there does not appear to be any observable trend between jitter and the required fidelity, while the RCPSP-NP-FPR and RCPSP-NP-EDF see an increase in jitter as the required fidelity increases. We may expect this given that the latency of repeater protocols increases as the fidelity requirement increases.

This can cause protocols to experience longer amounts of delay, possibly increasing the inter-delivery jitter.

5.2.3. LPRP TDMA Scheduling

Now that we have presented the results for the NPRP TDMA scheduling techniques we may move on to analyzing the performance of the LPRP TDMA scheduling techniques. We remark that any scheduler used for the NPRP TDMA schedule construction can be used for LPRP schedule construction. The difference is that the NPRP schedulers do not take advantage of the permitted delay in between protocol actions. We will first show how the amount of permitted delay impacts the performance of the LPRP techniques before comparing them to the NPRP scheduling techniques presented previously.

Effects of Preemption Budget

In order to observe the effects of the permitted preemption delay on the performance of LPRP TDMA schedulers we simulated three schedulers on the line topology in figure 5.6c using the same procedure of simulating 100 demand sets as described at the start of the section. The three schedulers we analyze here are the EDF-LBF periodic task scheduler (PTS-PB), the RCPSP LPRP TDMA scheduler (RCPSP-PB) and the segmented protocol RCPSP scheduler (RCPSP-PBS).

For each set of demands we altered the permitted preemption delay to be 10 ms, 100 ms, and 1 s to see if increasing the permitted delay improves the behavior of these schedulers. We would expect that increasing the permitted delay allows repeater protocols to start earlier and resume when network resources are less congested. We compare these to the PTS-NP-EDF and RCPSP-NP-EDF techniques from the NPRP TDMA scheduling analysis to have a general idea of how preemption delays improve performance.

Figures 5.22, 5.23, and 5.24 show the results of increasing the permitted preemption delay on scheduling performance metrics. Here, we use PTS-NP-EDF as a baseline for PTS-PB while we use RCPSP-NP-EDF as a baseline for RCPSP-PB and RCPSP-PBS.

For PTS-PB we see a notable increase in the throughput as we increase the permitted delay in the protocols. The increase is even more pronounced for higher fidelity requirements which may be attributed to the increased latency of the repeater protocols needed to satisfy them. In order to meet fidelity demands, many more actions are involved in the protocol which allows additional preemption to take place when scheduling. Increasing the permitted delay appears to also increase the resulting worst-case response time for PTS-PB while having little effect on the average jitter.

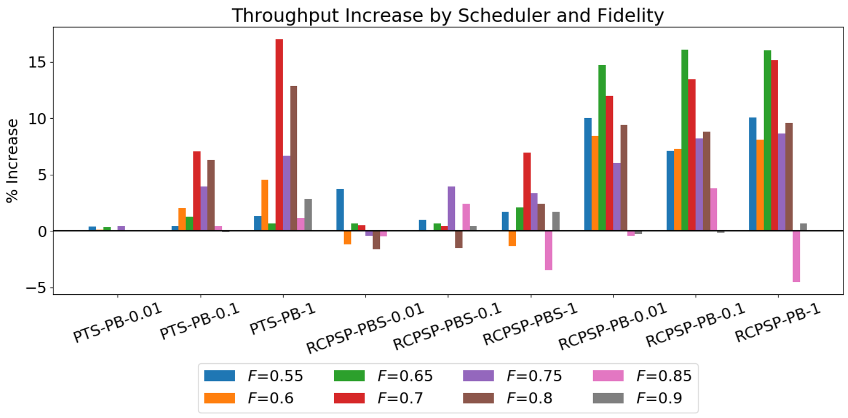


Figure 5.22: Average increase in throughput (over 100 simulations) when varying the permitted amount of delay in repeater protocols.

RCPSP-PBS observes a less dramatic effect on the throughput when increasing the permitted delay as opposed to RCPSP-PB which appears to observe a consistent increase over RCPSP-NP-EDF even in the case when 10 ms of delay is permitted. For both of these schedulers there appears to be a few cases where permitting delay in the repeater protocol reduces the achievable throughput when compared to RCPSP-NP-EDF. This agrees with our expectations that starting protocols earlier allows a reduction in worst-case response time.

In contrast to PTS-PB, we see that the worst case response time dramatically decreases with these techniques as the protocols are able to begin earlier in time when delay is permitted. This is most notable for higher fidelity requirements where there are many points of preemption during a repeater protocol. We believe that this is due to the fact that no parallelism can be extracted when many repeater protocols overlap in the network. The use of EDF-LBF allows preemption to occur in order to meet deadlines of tasks in the system but this causes longer delays in completing protocols as seen when increasing the permitted delay.

From figure 5.24 we see that the LPRP schedulers observe higher amounts of jitter compared to the NPRP schedulers on the line topology in figure 5.21. Since permitting delay in repeater protocols allows some instances of the repeater protocol to start earlier there is greater variation in the response times of individual instances.

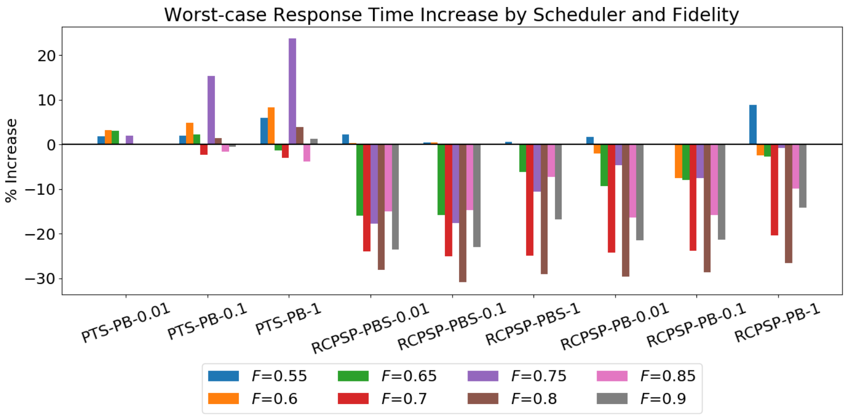


Figure 5.23: Average increase in worst-case response time (over 100 simulations) when varying the permitted amount of delay in repeater protocols.

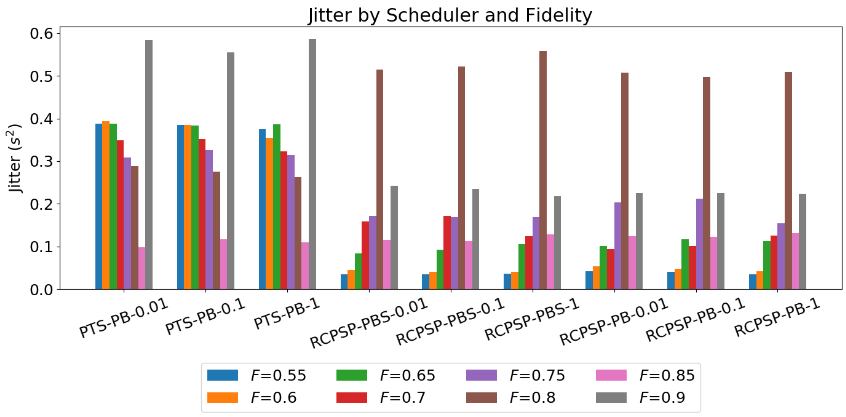


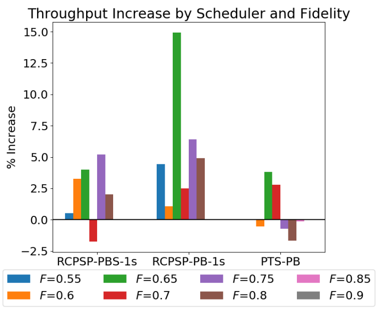
Figure 5.24: Average jitter (over 100 simulations) observed when varying the permitted amount of delay in repeater protocols.

We now revisit the experiments from the NPRP TDMA scheduler analysis and study the LPRP TDMA schedulers to see how performance metrics vary when using LPRP TDMA schedulers. For these comparisons we fix permitted delay of repeater protocols to be 1s for the PTS-PB, RCPSP-PB, and RCPSP-PBS schedulers. We omit the results of the star topology as we found that due to the limitation of a single repeater node there was little observed difference in the schedulers.

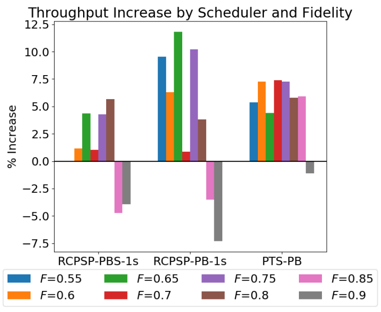
Network Throughput

Figures 5.25a and 5.25b show the average increase in network throughput for the LPRP TDMA schedulers. In contrast to what was seen in figure 5.9, the RCPSP-PB* schedulers achieve an increase in throughput for fidelity requirements of 0.75 and 0.8 as opposed to a decrease seen in RCPSP-CEDF and RCPSP-FPR while the throughput increase is not as pronounced for lower fidelity requirements.

However, the line topology results in figure 5.25b show an even more notable difference for the RCPSP-PB* schedulers as throughput for lower fidelity requirements increased compared to the NPRP schedulers on the same demand sets. This shows us that even using a project transformation to segment the protocol can outperform RCPSP-NP-EDF thanks to permitted delay even though the segments still over-allocate the resources used.



(a)



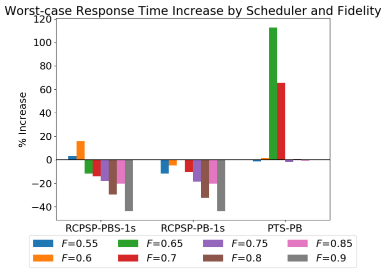
(b)

Figure 5.25: Average network throughput increase (over 100 simulations) of LPRP scheduling techniques on the a) H topology and b) line topology. PTS-NP-EDF is the baseline for PTS-PB while RCPSP-NP-EDF is the baseline for RCPSP-PB* schedulers.

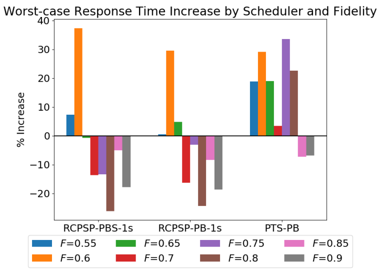
The results for the PTS-PB scheduler show an increase in network throughput similar to the PTS-CEDF scheduler for these topologies. In this case it may be preferable to use the PTS-CEDF scheduler as similar levels of throughput are achievable without delaying protocol actions.

Worst-Case Response Time

The increase in worst-case response times of the LPRP TDMA schedulers for the H topology and line topology can be seen in figure 5.26a and 5.26b. As we saw in the analysis of effects of preemption budget, the RCPSP-PB* schedulers show a decrease in worst-case response time for higher fidelity requirements compared to RCPSP-NP-EDF. The PTS-PB scheduler appears to show an increase with respect to PTS-NP-EDF for the majority of fidelity requirements while still reducing the response time for higher fidelity requirements.



(a)



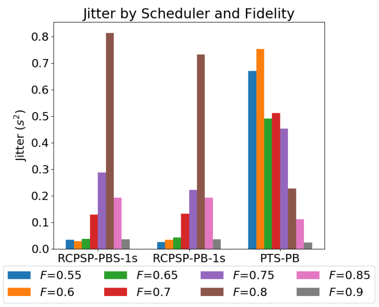
(b)

Figure 5.26: Average increase in worst-case response time (over 100 simulations) of LPRP scheduling techniques on a) H topology and b) line topology. PTS-NP-EDF is the baseline for PTS-* schedulers while RCPSP-NP-EDF is the baseline for RCPSP-* schedulers.

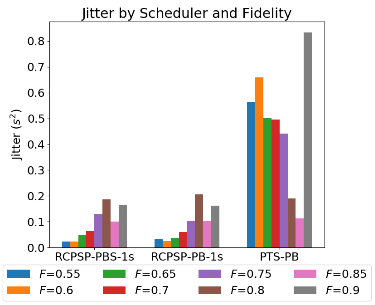
Jitter

We finish the comparison of LPRP TDMA scheduling techniques to NPRP by examining the average jitter observed for the H and line topologies. The results of the H topology in figure 5.27a for the RCPSP-PB* schedulers resemble those found for the RCPSP-NP-EDF scheduler shown previously while the PTS-PB scheduler shows a significant increase in jitter over the PTS-NP-EDF scheduler and PTS-CEDF schedulers. The line topology results in figure 5.27b show that the jitter of the RCPSP-PB* schedulers is comparable to that seen for RCPSP-NP-EDF while PTS-PB shows significant increase as for the H topology.

These results seem to suggest that the LPRP scheduling techniques are able to provide improvement in network throughput for lower fidelity requirements while decreasing the worst-case response time and maintaining a similar jitter behavior to what is observed for the RCPSP-NP-EDF scheduler.



(a)



(b)

Figure 5.27: Average jitter (over 100 simulations) of LPRP scheduling techniques on the a) H topology and b) line topology. PTS-NP-EDF is the baseline for PTS-* schedulers while RCPSP-NP-EDF is the baseline for RCPSP-* schedulers.

5.3. Resource Allocation

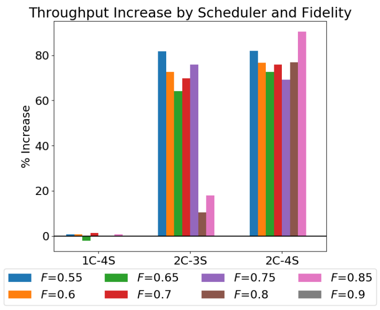
We now showcase the effects of resource allocation in the network to observe the impact on scheduling performance metrics. Specifically, we first show improvement in scheduler performance by adding additional resources to repeater nodes that are heavily congested and then show how allocating subsets of network resources to repeater protocols affects the behavior of the schedulers.

5.3.1. Increasing Repeater Resources

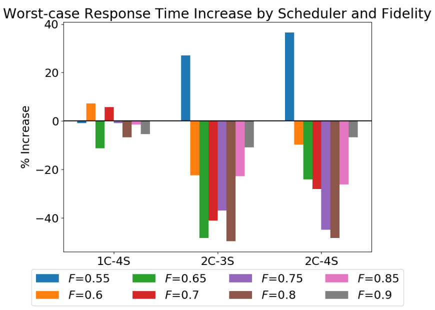
As we saw in the case of the star topology when analyzing NPRP scheduling techniques, the achievable network throughput was limited by contention on the central repeater. We also saw from the concrete protocol analysis that increasing node resources is able to reduce the latency of repeater protocols as more elementary link generations can be performed in parallel, thus providing a higher rate.

To observe the effects of increasing network resources we used the star topology as before and increased the number of communication qubits and storage qubits at the central repeater. We used the one communication qubit and three storage qubit case as a baseline and investigated increasing each of these quantities by an additional resource. In doing so, the end nodes consistently have a single communication qubit and three storage qubits as before. This type of network configuration may reflect how nodes internal to the network that experience heavy congestion should be built with additional network resources in order to accompany demand.

For each of the repeater node configurations we simulated 100 demand sets for each demand fidelity and randomized the rates as for the previous experiments. Here we investigate the results of using the RCPSP-NP-EDF scheduler for all simulations and we use the one communication qubit and three storage qubit configuration from the NPRP analysis as a baseline for comparison.



(a)



(b)

Figure 5.28: a) Average network throughput increase (over 100 simulations) and b) average worst-case response time increase (over 100 simulations) of RCPSP-NP-EDF on the star topology when increasing repeater resources. ”1C-4S” corresponds to one communication qubit and four storage qubits, ”2C-3S” corresponds to two communication qubits and three storage qubits, and ”2C-4S” corresponds to two communication qubits and four storage qubits at the central repeater node.

Figure 5.28a shows the average increase in throughput for the investigated configurations. We see that simply adding a single storage qubit to the central repeater has little effect on the average change in throughput. In contrast, adding a single communication qubit is able to significantly improve the throughput of lower fidelity requirements. By adding both we see that the throughput of higher fidelity requirements also increases significantly.

The lack of increase in throughput for adding a single storage qubit to the repeater node may be due to the fact that adding storage resources does not allow any additional links to be built in parallel. The reason that we see a significant increase for higher fidelity requirements when both of these resources are increased may be attributed to the fact that repeater protocols that use higher levels of nested entanglement distillation can be realized as there are more resources to hold previously generated links. For lower fidelity requirements we see that this additional storage qubit has little effect as it may be unnecessary to use these storage qubits to achieve lower latency for repeater protocols with lower fidelity requirements.

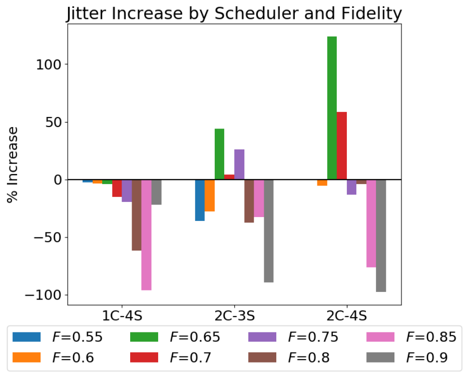


Figure 5.29: Average increase in jitter (over 100 simulations) of RCPSP-NP-EDF on the star topology when increasing repeater resources.

The worst-case response time is also greatly reduced when adding communication resources to the central repeater. Figure 5.28b shows that there is an average increase in worst-case response time for a fidelity requirement of 0.55 whereas all higher fidelity requirements reduced by 5-40%. It is not clear why a fidelity requirement of 0.55 experiences such an increase in worst-case response time. One may think that by increasing the network throughput there are more repeater protocols in the system which may delay one another, though we would expect this type of behavior for all fidelity requirements.

While the network throughput and worst-case response time were not significantly affected by the addtion of a single storage qubit to the central repeater, we see that this can greatly reduce the amount of jitter experienced by higher fidelity requirements as seen in figure 5.29. Increasing the communication qubit resources seems to have mixed results with higher jitter experienced for fidelity requirements of 0.65, 0.7, and 0.75 while other fidelity requirements experienced reduced average jitter.

These results show that allocating additional resources to congested repeater nodes allows one to achieve higher overall network throughput while reducing worst-case response time and jitter for some fidelity requirements. It is thus not necessarily the case that end nodes need to have large numbers of resources in order to accommodate them into the network as the resources they use in a repeater protocol are only used for connecting to the directly connected repeater and are not shared among other protocols. Repeater nodes, on the other hand, must have higher counts of resources to support link establishment with both connected nodes along several repeater chains that may cross one another. Should end nodes act as repeater nodes then it becomes necessary to improve the resource capabilities of these nodes to accommodate more repeater protocols.

5.3.2. Reducing Protocol Resources

Up to this point we have constructed protocols for the network such that they maximally use network resources where available. From the previous section we saw that increasing the resources of a highly congested repeater node has significant improvement on the network performance. As we saw from figures 5.4 and 5.5 in the protocol analysis, increasing the node resources results in a decrease in latency of a repeater protocol which corresponds to a higher execution rate.

The resulting speedup from increasing node resources is less than two times that for the reduced number of resources. This observation appears to suggest that if we construct protocols in the network using a subset of the network resources, we can achieve a higher network throughput as the vacant resources can be used for other protocols in parallel.

To investigate this further, we simulated our RCPSP schedulers on a star topology with two communication qubits and four storage qubits. For each fixed demand fidelity we generated 100 demand sets and performed two separate simulations where in one case the constructed repeater protocols for each demand used the full set of resources along its repeater chain while the second case used repeater protocols that only used half of the resources (one communication qubit and two storage qubits). We then obtained the throughput and worst-case response time of the simulated schedules and plotted the average increase of these metrics using repeater protocols that use the full set of resources as a baseline.

Figure 5.30 shows the average change in network throughput for the RCPSP schedulers. From these results it is clear that lower fidelity requirements experience an increase in network throughput regardless of the chosen scheduling technique while higher fidelity requirements suffer a decrease in throughput. As mentioned in the previous sections, the number of communication qubits and storage qubits has a significant effect on the latency of repeater protocols that have higher fidelity

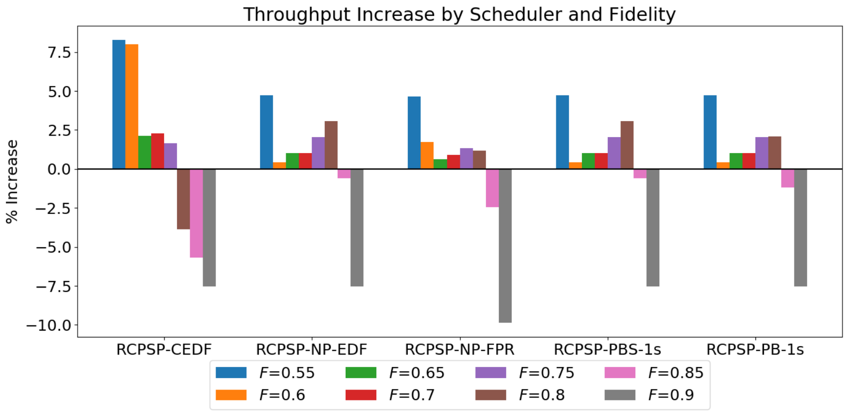


Figure 5.30: Average change in throughput (over 100 simulations) when reducing the allocated resources for repeater protocols.

requirement. This is because higher fidelity requirement repeater protocols generate significantly many more elementary links in order to satisfy their fidelity requirement. By increasing the number of communication resources there is a significant reduction in latency of the protocol thereby increasing the achievable rate.

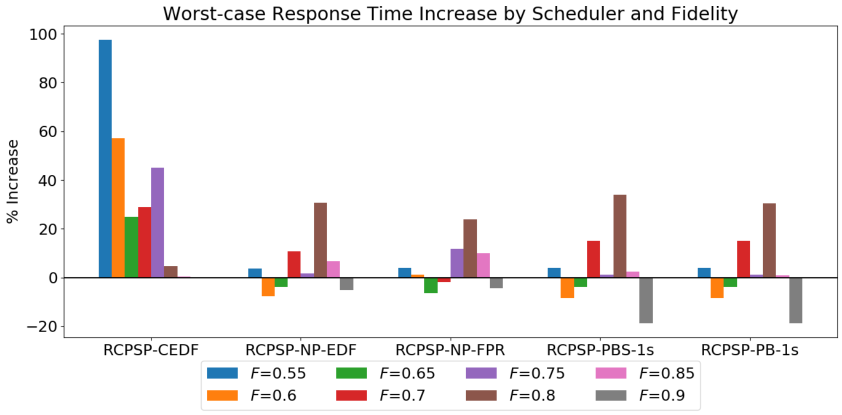


Figure 5.31: Average change in worst-case response time (over 100 simulations) when reducing the allocated resources for repeater protocols.

The worst-case response time on the other hand sees no clear trend in the increase of the fidelity requirement, though all schedulers other than RCPSP-CEDF appear to perform similarly for the same fidelity requirement. In some cases such as $F = 0.8$, the increase is as high as 20% while $F = 0.9$ shows a decrease as low as 20% for the LPRP schedulers.

These results suggest that appropriate selection of used resources for repeater protocols may have a significant effect when demand sets contain mixed fidelity requirements. Lower fidelity requirements may be allocated a smaller subset of resources whereas higher fidelity requirements may be allocated larger number of resources in order to increase the overall network throughput.

References

[1] C. Ekelin, *Clairvoyant non-preemptive edf scheduling,* in *18th Euromicro Conference on Real-Time Systems (ECRTS'06)* (IEEE, 2006) pp. 7–pp.

6 Conclusion

Here we conclude the thesis with a discussion of the results of this work and practical considerations. We additionally discuss the prospect of future work that may extend upon what has been presented and an outlook of this work's applicability.

6.1. Summary

In this thesis we have proposed a novel method of multiplexing network resources for repeater protocols in networks of quantum processors. Our proposal uses dynamic time-division multiple access channel methods that encode repeater protocols between end nodes in the network. This method allows supporting flexible Quality of Service requirements for different applications among multiple users of the network. We additionally examine the step of constructing the TDMA schedule based on demands and their concrete repeater protocols and show two different TDMA scheduling problems that apply to current state-of-the-art devices and those in the near-term future. For each of these problems we propose two novel formulations of the scheduling problem using periodic task scheduling and resource-constrained project scheduling as a way of constructing the TDMA schedules.

Our analysis and results have shown how various network parameters influence the behavior of repeater protocols in the network and how different scheduling techniques can be used to achieve different trade-offs between the quality of the produced TDMA schedule and the complexity of the scheduler. To address the LPRP TDMA scheduling problem in the context of periodic task scheduling, we have introduced a new limited preemption task scheduling problem known as preemption budget task scheduling and provided a description of proposed heuristic for scheduling tasks under this setting. Our results have shown that our heuristic can outperform non-preemptive scheduling methods in the setting of constructing TDMA schedules.

We have additionally investigated the effects of resource allocation to congested repeaters nodes as well as repeater protocols that operate in the network to see the effects on various performance metrics.

6.2. Discussion

The results of comparing the two different scheduling formulations have shown that using RCPSP methods for constructing the TDMA schedule demonstrate great improvement in network throughput over periodic task scheduling methods. This improvement comes at the cost of increased worst-case response time and jitter in the inter-delivery of entanglement to end nodes in the network. This data may be useful to those building networks of quantum processors that need suitable average behavior from the network to support applications deployed on end nodes.

Our comparison of the NPRP and LPRP TDMA scheduling problems show that using limited preemption techniques can provide significant increase in achievable network throughput while decreasing worst-case response time. This motivates the development of networked quantum processors that tolerate longer storage times of entangled links.

We have additionally investigated the effects of resource allocation to congested repeaters nodes as well as repeater protocols to see the effects on various performance metrics. Our results show that increasing the amount of resources at congested repeater nodes in the network has a great impact on the achievable throughput without increasing the network resources of end nodes. Furthermore, allocating a subset of network resources to low fidelity requirement repeater protocols can greatly improve the achievable network throughput without significant effect on the worst-case response time of the repeater protocols. These results may be useful to those researching routing and protocol selection for sets of demands that observed mixed fidelity requirements.

6.3. Future Work and Outlook

While we believe that our approach presents a method for operating near-term quantum networks built on NISQ devices, there are many questions and alternative designs that should be investigated as the quality of devices progress. Here, we will summarize several avenues to further the research in traffic engineering of quantum networks.

- This work managed routing, protocol selection, and TDMA scheduling using a centralized controller in the network that had full knowledge of network details. Using a central controller to manage ever-growing networks may not scale well and an alternative design could be one that performs these actions in a distributed manner across the network much like the distributed approach taken in [1].
- In the case of using a central controller, knowledge of the constructed TDMA schedule may be used as feedback to routing and protocol construction stages to better utilize the remaining resources in the network. A more interesting approach to coordinating repeater protocols may be to perform routing, protocol selection, and scheduling jointly similarly to what is done in [2].
- Our TDMA schedule construction encodes entire repeater protocols between end nodes, dictating which elementary links are used to establish a session between end nodes. One might instead wonder if it is possible to construct a TDMA schedule for the network that allows a dynamic choice of the links to be used for a protocol.
- The focus of this thesis was on near-term NISQ devices used for quantum networks. One could investigate CDMA [3] multiplexing methods for future devices that tolerate large latency in repeater protocols. Using CDMA may involve dynamically requesting directly connected nodes for generating entanglement between one another rather than scheduling an agreed time in advance.
- Our method for constructing concrete repeater protocols used a constant entanglement flow based heuristic. This heuristic may not find protocols that have maximal rate over all possible repeater protocols between end nodes in the network. It would be interesting to see how the complexity of other protocol generation methods may influence the achievable network throughput.

- Many simplifications have been made to our network model in order to focus on the effects of scheduling techniques. Decoherence of stored links and gate noise were ignored in the effect on fidelity of repeater protocols and we assumed that qubit connectivity in a network node is fully connected. NISQ devices suffer from considerable noise and often have limited qubit connectivity at a node. Being able to use any communication qubit to generate entanglement with directly connected nodes may be difficult to achieve in practice and may impose further limitations on network performance.

- In addition to noise, operations like entanglement swapping or entanglement distillation may be probabilistic depending on the processor architecture. Failures of these operations can significantly reduce the achievable network throughput if they cause the entire repeater protocol to fail. Advanced methods of protocol construction like that in [4] may provide backup links to be generated in the case when steps of a protocol fail.

- In this work we have considered the probabilistic heralded entanglement generation method based on NV centres in diamond. Alternative platforms for quantum processors such as atomic ensembles [5, 6] offer a near unity chance of successful entanglement generation and may have significant impact on the achievable rates and TDMA schedule encoding. This is due to the fact that near unity success rates will require shorter amounts of time to generate links.

- Our methods for TDMA schedule construction has used heuristic methods based on periodic task scheduling and resource-constrained project scheduling. Other methods for constructing a schedule may involve branch-and-bound methods that have higher complexity but may produce schedules with higher throughput. An analysis of how these methods compare to the heuristic methods may provide additional information on the trade-offs.

- Our scheduling heuristics were only made to satisfy rate requirements of demands without imposing other constraints on worst-case response time or jitter. Without taking these into consideration the provided TDMA schedule may not satisfy the specific needs of deployed applications. Other QoS parameters such as tolerable jitter and soft rate requirements can be investigated to observe the impact on achievable network throughput and the metrics investigated here.

- One significant obstacle in engineering our proposed method is the requirement of tight timing synchronization across nodes in the network. In practice such timing synchronization may only be achievable up to a certain geographical limitation and it is worth investigating how the approach taken in this thesis scales with the size of the network.

These alternative designs and research questions introduce a whole field of quantum network traffic engineering that can be useful for the architecture of future quantum networks. There is still much to be learned from classical networks in the construction and management of quantum networks and especially those at a global scale.

References

[1] K. Chakraborty, F. Rozpedek, A. Dahlberg, and S. Wehner, *Distributed routing in a quantum internet,* arXiv preprint arXiv:1907.11630 (2019).

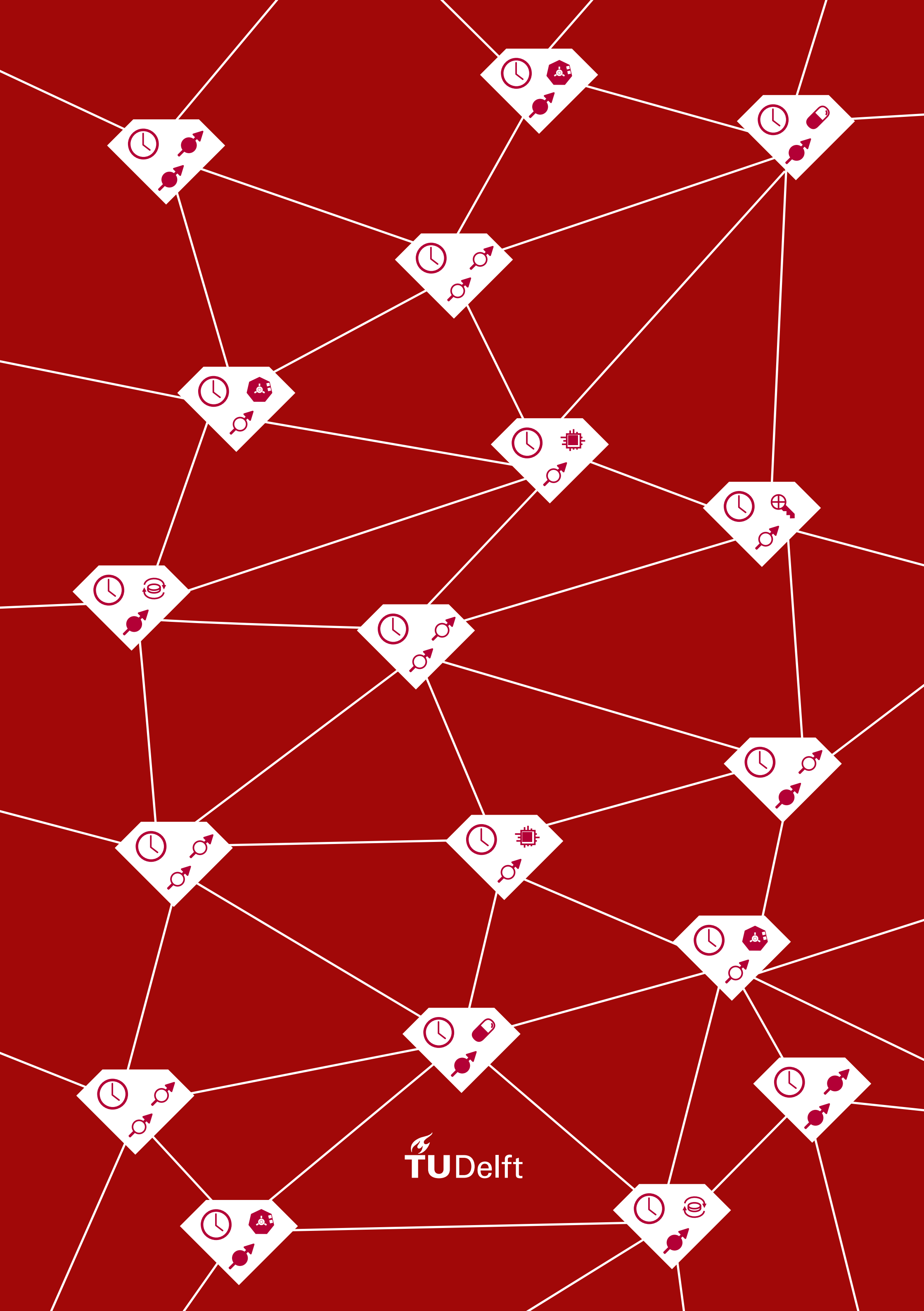
[2] K. Chakraborty, D. Elkouss, B. Rijsman, and S. Wehner, *Entanglement distribution in a quantum network, a multi-commodity flow-based approach,* arXiv preprint arXiv:2005.14304 (2020).

[3] A. K. Jagannatham, *Principles of modern wireless communication systems* (McGraw-Hill Education, 2015).

[4] S. Shi and C. Qian, *Concurrent entanglement routing for quantum networks: Model and designs,* in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication* (2020) pp. 62–75.

[5] L.-M. Duan, M. D. Lukin, J. I. Cirac, and P. Zoller, *Long-distance quantum communication with atomic ensembles and linear optics,* Nature **414**, 413 (2001).

[6] C.-W. Chou, H. de Riedmatten, D. Felinto, S. V. Polyakov, S. J. Van Enk, and H. J. Kimble, *Measurement-induced entanglement for excitation stored in remote atomic ensembles,* Nature **438**, 828 (2005).


TUDelft