Simulations of Quantum Protocol in Distributed Networks using ns-3

Rasmus Hjøllund Davidsen
Computer Engineering, Group 1048, 2025

Master Thesis

This report is typeset using the Typst system.

AALBORG UNIVERSITY
STUDENT REPORT

Department of Electronic Systems
Aalborg University
https://www.es.aau.dk

Title:
Simulations of Quantum Protocol in Distributed Networks using ns-3

Theme:

Project Period:
Spring Semester 2025

Project Group:
Group 1048

Participants:
Rasmus Hjøllund Davidsen

Supervisors:
Petar Popovski
Kristian Skafte Jensen

Copies: 1

Number of Pages: 34

Date of Completion:
03/06-2025

Abstract:

This thesis investigates a new protocol proposed in 2025 for sequential entanglement swapping in Ethernet LANs. The protocol is analysed in conjunction with necessary modifications to classical communication protocols e.q. a new protocol Q-STP to factor quantum links into routing decisions. A custom implementation was simulated using the ns-3 network simulator. The results demonstrate that the protocol functions effectively in practice and highlight its performance under varying conditions and topologies, particularly with respect to latency. Findings indicate that reducing the number of intermediate switches improves performance; however, this advantage must be weighed against the problems associated with longer quantum links.

Preface

This thesis was written as part of the Master's program in Computer Engineering, Networks and Distributed systems at Aalborg University. The research was conducted from February to June of 2025.

I would like to express a special thanks to Kun Chen Hu, whose insights and assistance has helped immeasurably during the making of this thesis.

Table of Contents

Introduction

The field of quantum communication has gained significant attention in recent years due to its promise of fundamentally secure information exchange and the potential to revolutionize distributed computing and networking. At the core of quantum communication lies the phenomenon of entanglement, a non-classical correlation between quantum particles that allows instantaneous state correlations, regardless of the distance between them [1]. Entanglement enables a range of powerful protocols, such as quantum teleportation, quantum key distribution (QKD), and distributed quantum computation.

However, directly distributing entangled pairs over long distances is limited by photon loss and noise in quantum channels. To overcome this limitation, entanglement swapping has emerged as a key technique. Entanglement swapping allows two particles that have never interacted to become entangled through a series of intermediate measurements involving other entangled pairs [2]. This concept forms the basis for quantum repeaters, which are essential for extending quantum communication across large-scale networks [3].

A particularly important variant of this technique is sequential entanglement swapping, where entanglement is extended step-by-step across multiple intermediate nodes. This sequential approach is crucial for constructing scalable and modular quantum networks, especially in real-world scenarios where each swap may involve coordination with classical communication channels.

As the vision of a quantum internet begins to materialize, there is a growing need to understand how these quantum protocols could operate within or alongside classical networking infrastructures. In particular, Ethernet LANs, which are widely used in local area networking, present an interesting environment to evaluate the performance and practicality of quantum communication protocols. Exploring the behavior of sequential entanglement swapping within such networks is a vital step toward bridging the gap between theoretical quantum communication and practical implementation.

This thesis builds upon the design proposed in [4], which outlines a protocol for performing sequential entanglement swapping in Ethernet LANs. While the original work focused on the theoretical design and communication flow, this thesis aims to extend the contribution by implementing and simulating the protocol using ns-3 to evaluate its performance under various network conditions.

The remainder of this thesis is organized as follows. Chapter 1 provides an overview of the relevant quantum and classical data link layer concepts, along with a detailed description of the protocol proposed in [4]. Chapter 2 describes the simulation setup in ns-3, including the network configuration and the specific aspects of the protocol to be tested. Chapter 3 presents and analyzes the simulation results, with a focus on latency measurements. Chapter 4 discusses the implications of the results, their limitations, and potential improvements to the protocol. Chapter 5 concludes the thesis and outlines directions for future work.

Chapter 1

Analysis

This chapter provides the theoretical background necessary to understand the protocol evaluated in this thesis. It begins with a brief overview of the fundamental concepts in quantum communication, focusing on those relevant to networked systems. Next, it introduces classical link-layer principles, which form the basis for protocol design and implementation. Finally, the chapter presents a high-level description of the protocol studied in this work, outlining its key mechanisms and expected behavior. This foundation sets the stage for the simulations performed in the following chapters.

1.1 Quantum Concepts Overview

This section provides a brief overview of the quantum mechanical principles that underpin the protocol studied in this thesis. The topics are quantum entanglement, entanglement swapping, and sequential entanglement swapping—all of which form the theoretical foundation for distributed quantum communication and are essential to understanding the protocol's design and coordination requirements.

1.1.1 Entanglement

Entanglement is an important part of quantum information. Putting it simply when two quantum systems are entangled it means that there exists some correlation between them (in a non-classical sense). An example of entanglement is the so called Bell states (also called EPR pairs), the first of which is shown in Equation (1.1) [5]:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle. \tag{1.1}$$

In this state if the first qubit is measured in the computational basis then the second qubit will collapse to that value i.e. if the first qubit is measured as 0 then the second qubit will collapse and the state becomes $|00\rangle$ and vice versa.

1.1.2 Entanglement Swapping

Entanglement swapping is a method for creating entanglement between two qubits that never interacted by performing a joint measurement on their respective entangled partners [6]. This is useful since direct sharing of entanglement over long distances is likely to fail due to decoherence and loss [7]. If Alice shares a maximally entangled state $|\Phi^+\rangle_{\mathrm{AC}}$ with Charlie, and Bob does the same with David then:

$$|\Phi^+\rangle_{\mathrm{AC}} \otimes |\Phi^+\rangle_{\mathrm{BD}} \tag{1.2}$$

If Charlie and Bob then perform a joint measurement in the Bell basis and communicate the results to Alice and David via a classical channel, then Alice and David can perform local operations to obtain the entangled state $|\Phi^+\rangle_{AD}$ [8]. This way entanglement can be transferred to the particles of Alice and David without the particles ever *meeting* physically.

1.1.3 Sequential Entanglement Swapping

The concept of entanglement swapping just introduced can be performed multiple times to create entanglement between particles with greater distance. Figure 1.1 shows can example of a method of performing sequential entanglement swapping from [4]. At the start all switches are entangled with their neighbors. Then the switches closest to the Alice and Bob (Switch 1 and 3) perform entanglement swapping using the qubits they posses. This creates entanglement between Alice's qubit and Switch 2′s first qubit, and between Bob's qubit and Switch 2′s second qubit. Finally Switch 2 can perform the entanglement swapping on the two qubits that are entangled with Alice and Bob, which, if successful, entangles Alice's and Bob's qubits.

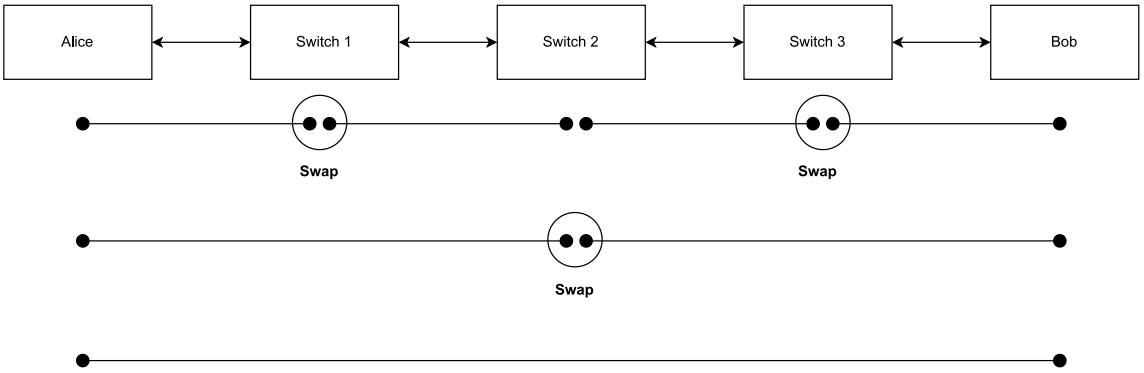


Figure 1.1: Example of Sequential Entanglement Swapping procedure.

1.2 Classical Layer 2 Networking Concepts

This section outlines the relevant aspects of Layer 2 in the OSI model—the Data Link Layer [9]. Key topics include Medium Access Control (MAC) mechanisms in technologies such as Ethernet and Wi-Fi, the structure and use of MAC addressing, and the format of common Layer 2 frame types. Additionally, the role of the Layer 2 network device known as a *bridge* will be introduced, with a focus on how they forward frames and manage traffic within Local Area Networks (LANs). Bridges are sometimes called switches depending on the context, and both are used from this point to mean the same thing.

1.2.1 Medium Access Control

An important part of the data link layer is the MAC sub-layer, which has many responsibilities such as determining when a device can transmit, how to deal with collisions and contention, and how to ensure fairness and efficiency of the shared channel. Many MAC mechanisms have been proposed throughout the years and different mechanisms are used in different scenarios. Here the focus will be on the mechanisms used in the Ethernet and Wi-Fi protocols: CSMA/CD and CSMA/CA.

In early versions of Ethernet a shared-bus was used as the medium, thus a mechanism for sharing the medium was required. The mechanism used is called Carrier Sense Multiple Access (CSMA)

with Collision Detection. In basic CSMA, stations listen to the channel before starting a transmission. While transmitting the frame the station is listening to the channel to detect if another station is also transmitting at the same time i.e. a collision. If a collision is detected the station stops the transmission. Networks using this MAC mechanism use no acknowledgement messages, but instead rely on the transmitting station to detect collisions. If a station does not detect any collision it assumes the frame was received without errors, otherwise it tries to retransmit it again after a timeout period (often exponential back-off) [10].

The MAC mechanism used in Wi-Fi is called CSMA with Collision Avoidance and differs from the CSMA/CD of Ethernet, in that it uses acknowledgements frames in response to correctly received frames. CSMA/CA also uses more delay times to more precisely determine when a station can transmit. Since Wi-Fi is wireless it also has to consider problems like the Hidden Station problem. It handles this by the use of two control frames: Request to Send (RTS) and Clear to Send (CTS). These two control frames can be used by a station to reserve the medium for a period of time to avoid collisions [10].

1.2.2 Addressing

In the MAC sub-layer all stations on a network have a MAC address. A MAC address is a unique identifier that is used for communication within a network segment and it is used in most IEEE 802 networking technologies, including Ethernet and Wi-Fi. An address of this kind consists of 48 bits and is typically represented by six groups of two hexadecimal digits. A set of two hexadecimal digits is also called an octet since it represents 8 bits.

MAC addresses can either be administered universally or locally, distinguished by the so called U/L bit which is the second least-significant bit of the first octet of the address [11]. If the address is administered universally the first three octets are a Organizationally Unique Identifier (OUI) assigned by IEEE, while the last three octets are chosen by the organization in question.

The least-significant bit of the first octet is referred to as the *Individual/Group* bit. If the bit is set to 0 it means the frame is meant for only 1 receiving device. This type of address is called a *unicast* address and frames with an address of this type are generally ignored by all other stations. If the bit is set to 1, stations will accept or ignore the frame based on other criteria than matching their own address. This is called *multicast*. There exists some pre-defined multicast addresses such as the *broadcast* address *FF:FF:FF:FF:FF:FF* and stations can be configured to accept frames with specific multicast addresses.

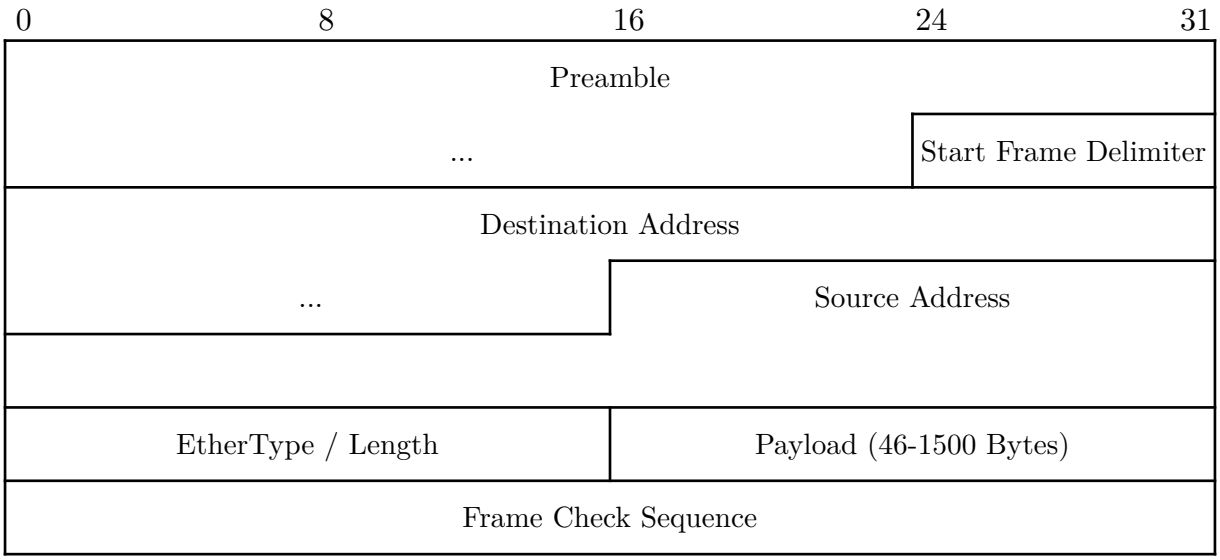


Figure 1.2: Ethernet Frame Structure.

1.2.3 Ethernet Frames

The IEEE 802.3 Ethernet standard defines the structure of Ethernet frames seen on Figure 1.2. The frame encapsulates higher layer data referred to as the payload. A frame starts of with the *preamble* which is 7 bytes of 0s and 1s that allow the receiving circuitry to synchronize with the frame's timing. Then comes the Start Frame Delimiter (SFD) which is the bit sequence 10101011 signalling to the receiver that the preamble has ended and that MAC frame is starting. The next two fields are the *destination* and *source* addresses of 48 bits each. The next 2 bytes are either the EtherType or the payload length depending on the value of the bytes. The payload can be anywhere from 46 to 1500 bytes, which might be followed by some padding. The final field is the Frame Check Sequence (FCS) containing a 32-bit Cyclic Redundancy Check (CRC) value used for error detection. The CRC value is computed using the address fields, the type/ length field, and the payload [11].

1.2.4 Bridges

A bridge is a type of layer 2 network device used to connect LANs together. A bridge must have at least two ports that receive and transmit frames on LAN segments [12]. The objective of a bridge is to forward frames between LAN segments and this requires the following functionalities [12]:

- The Forwarding Process.
- The Learning Process.
- The Filtering Database (will be called MAC table from now on).

The main objective of the *forwarding process* is to forward frames received on a port on other ports based on different criteria such as topology restrictions and the MAC table. To enforce topology restrictions means frames are only retransmitted if, and only if:

- The port the frame was received on is in the *Forwarding* state.
- The port considered for retransmission is also in the forwarding state.

- The port considered for transmission is not the same as the port the frame was received on.

If this does not apply for a port, the frame is discarded from that port. The frame then goes through a filtering process based on the destination MAC address and the information in the MAC table [12].

The *learning process* is used to learn the MAC addresses from incoming frames on each port. When a frame is received the MAC address and the port are being added to MAC table on the conditions that the port is in a state that allows for learning and that the address is an individual address.

The MAC table is a database that can be queried by the forwarding process to help determine which ports a frame should be forwarded on. The MAC table can contain both static and dynamic entries, but the latter is the most interesting for this project. Both static and dynamic entries contain [12]:

1. An individual MAC address.
2. A Port map that specifies what port, frames with that destination MAC address should be forwarded on.

Dynamic entries are automatically removed from the MAC table, if the *Ageing time* has elapsed since the entry was added or last updated, while static entries are never removed based on ageing time. This ageing out of dynamic entries is done in case end stations move to a different LAN segment or if the active topology changes which could cause the new path to the end station to be through a different port. The default ageing time is 300 s [12]. An example of a MAC table with the associated time of learning and expiration time is shown in Table 1.1.

MAC Address	Port	Learned Time	Expiry Time
00:11:22:33:44:55	1	12.5 s	312.5 s
AA:BB:CC:DD:EE:FF	2	14.5 s	314.5 s

Table 1.1: Example MAC Table.

1.3 Spanning Tree Protocol and its Extension for Quantum Networks

In this section, the classical Spanning Tree Protocol (STP) is first introduced, focusing on its role in managing redundancy and preventing loops in Local Area Networks (LANs). Following this, it is examined how STP can be adapted to support quantum networking scenarios. The proposed extension introduces modifications that enable selective routing based on the nature of the traffic, distinguishing between quantum-related coordination messages and standard classical data traffic.

1.3.1 STP

The Spanning Tree Protocol (STP) is a layer-2 protocol, part of IEEE 802.1D [12], used to build loop-free logical topologies in Ethernet networks. STP creates a *spanning tree* of the nodes in the network and it disables links that are not part of the spanning tree, so there is only one active path between two nodes. This is done to avoid switching loops that can results in problems like broadcast storms, since Ethernet frames do not contain a Time to Live (TTL) field and

broadcast packets can circulate the network infinitely and thereby degrading the performance of the network.

The first step in STP is to decide on a *root bridge* or *root switch*, which as the name suggests is the root of the spanning tree. The root bridge is determined based on a *Bridge ID* (consisting of a bridge priority and the MAC address of the bridge) by first comparing the bridge priorities and if they are equal using the MAC address as the tiebreaker.

Since each bridge only can use the information it has available, a special type of data frame called Bridge Protocol Data Unit (BPDU) is used to propagate information such as bridge IDs and root path costs between the switches. Configuration BPDUs are sent out by the root bridge in an interval based on the *Hello Time* and non-root bridges forward received BPDUs according to the same interval. Topology Change Notification (TCN) BPDUs are sent in the opposite direction i.e. towards the root bridge to notify of changes to the topology. The frame structure of a configuration BPDU can be seen in Table 1.2. BPDUs are sent with the destination MAC address 01:80:C2:00:00:00, which means only switches configured to use STP will read the packets.

Field	Size (Bytes)	Description
Protocol ID	2	Always 0.
Version	1	Always 0.
BPDU Type	1	Indicates the type of BPDU: 0x00 for Configuration and 0x80 for TCN.
Flags	1	Flags to indicate if the network topology has changed.
Root ID	8	The ID of the current root bridge.
Root Path Cost	4	The cumulative cost of all links to the root bridge.
Bridge ID	8	The ID of the bridge sending the BPDU.
Port ID	2	The ID of the port sending the BPDU.
Message Age	2	The age of the information was that used to create the BPDU i.e. the information from the root bridge. Is often increased by 1 in each bridge.
Max Age	2	Indicates the maximum time that a BPDU is saved.
Hello Time	2	Indicates how often BPDUs should be sent.
Forward Delay	2	Indicates the time spent in Learning and Listening states.

Table 1.2: BPDU Configuration fields.

If more than one path exists between two switches, the cost of each path is calculated based on the bandwidth of the link and the path with the lowest cost (i.e. highest bandwidth) is chosen. The cost of a path is obtained by a look-up table. Examples of cost values can be seen in Table 1.3 with cost values from 1998 [12] in the second column and cost values from 2004's Rapid Spanning Tree Protocol (RSTP) [13] in the third column.

Data Rate	STP Cost	RSTP Cost
10 Mb/s	100	2,000,000
100 Mb/s	19	200,000
1 Gb/s	4	20,000
10 Gb/s	2	2,000
1 Tb/s	N/A	20

Table 1.3: Path Cost of Different Data Rates in STP.

Ports can have a few different roles in STP based on the calculations performed by the switches: root port, designated port and blocked port. The *root port* is the port on a switch that is closest to the root bridge in terms of cost. All non-root switches have exactly one root port, while the root bridge itself has zero. The root ports can be seen as leading towards the root bridge. A *designated port* is a port that is allowed to forward traffic. They are found on a per-segment basis. This is done by comparing the ports at each end of the segment and their port cost and the total cost calculated by STP for that port to get back to the root bridge. If one end of a segment is a root port, then the other end is a designated port. All the ports on the root bridge are designated ports. A *blocked port* is a port that does not forward any Ethernet frames (including BPDUs) and only listens for BPDUs to stay up-to-date on the network topology.

The ports can also be in a handful of different port states with some overlap with the port roles: blocking, listening, learning, forwarding, and disabled. In the *blocking* state the port is receiving BPDUs but not forwarding traffic to prevent loops. In the *listening* state the port is listening for BPDUs and awaits information that would cause it to return to blocking. MAC tables are not populated and no traffic is forwarded in this state. The main difference between the blocking and listening state is that if a port is in the blocking state it does not send out BPDUs. In the *learning* state the port does still not forward traffic, but learns MAC addresses and add them to its MAC table. In the *forwarding* state the port is in normal operation i.e. receiving and forwarding frames while still listening for BPDUs that indicate it should go to the blocking state. The disabled state is for ports that have been disabled by the network administrator. The RSTP protocol previously mentioned uses some different port roles and state to facilitate faster responses to link failures.

Figure 1.3 shows an example of four switches in a ring topology with the cost of each link shown next to it. The red square shows the root bridge and the green lines signify the links that will be used for communication between the switches after STP has created the loop-free topology. The black line with an x between S1 and S4 means that S4 has blocked its port connecting to that segment.

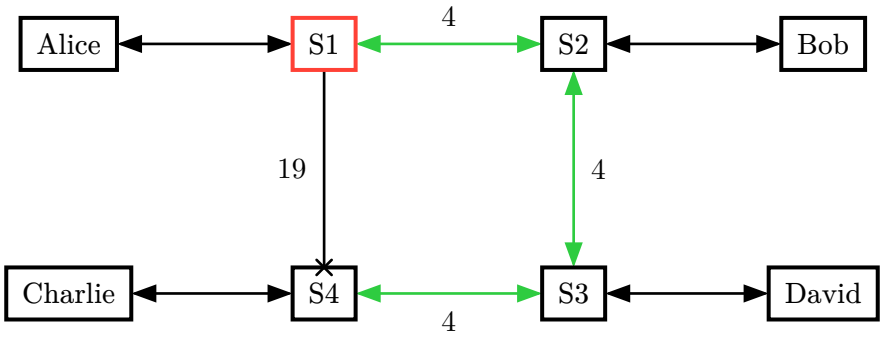


Figure 1.3: Example of STP.

1.3.1.1 Topology Changes

When a switch determines that a link failure has occurred (either due to not receiving a BPDU for the Max Age time or if it detects a port going down) it will generate a Topology Change Notification (TCN). A TCN (shown in Figure 1.4) is a very simple frame consisting only of the Protocol ID, Version and BPDU Type from the BPDU frame structure. The TCN is transmitted on the root port i.e. towards the root bridge. The upstream switch that receives the TCN sets the Topology Change Acknowledgment (TCA) field in its next configuration BPDU to notify the downstream switch that the topology change is acknowledged and retransmits the TCN on its root port. Once the root bridge receives the TCN it sets the Topology Change (TC) flag in its BPDUs for a period. This instructs the downstream switches to reduce their MAC address table ageing times to flush out old addresses and learn the new paths [12].

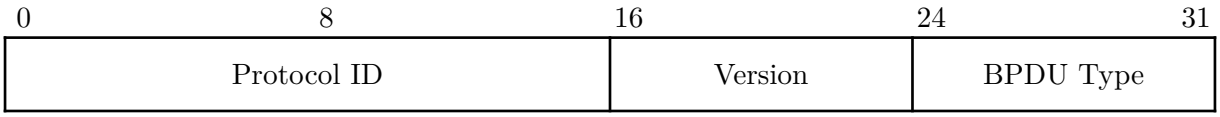


Figure 1.4: BPDU TCN Frame.

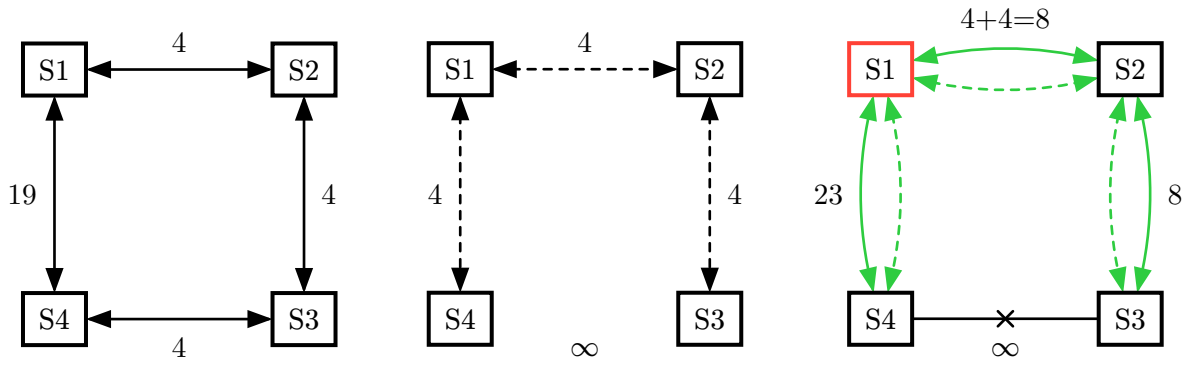
1.3.2 Q-STP

The Quantum-Spanning Tree Protocol (Q-STP) proposed in [4] is analogous to STP, except with the use of a different cost function that takes both classical links and quantum links into account. In the paper the cost of the l-th link is obtained by:

$$C_l = C_{c,l} + C_{q,l} \ , \quad \forall l \in \mathbb{L}_c \cap \mathbb{L}_q, \tag{1.3}$$

where $C_{c,l}$ is the cost of the l-th classical link and $C_{q,l}$ of the l-th quantum link. Since the cost of the classical communication channel is an integer value based on the bandwidth of the link, a similar concept could be used to obtain the cost of the quantum channel based on the quantum transmission rate in qbps (Qubits per second). Some balance between the cost of the two channels would have be achieved to ensure that the classical channel is good enough to facilitate the quantum protocol. An example of the behavior of Q-STP can be seen on Figure 1.5, where Figure 6a is the same classical network used above and Figure 6b is a network of quantum links represented by dashed lines. Since there is no link between S3 and S4 the cost is considered to be infinite. In this example by adding the cost of the classical and quantum links the network shown in Figure 6c is achieved, where the link between S3 and S4 is disabled. To achieve this in reality the cost of the quantum links could just be added to the classical cost already being transmitted in the BPDU.

An optimization that could be done to the protocol is to enable it to maintain different spanning trees for purely classical communication and the classical communication that is related to quantum communications. As shown on Figure 1.5 the protocol disables an otherwise better classical path because of a missing quantum link. Doing this on a real network would impair performance since it would force all traffic to use a slower link. Therefore, it could be beneficial to maintain two separate spanning trees so only the quantum related communication is using the worse path. A way to do this could be by adding a new 4 byte field to the previously defined BPDU to transmit the quantum cost along with the classical cost. Thus the Q-STP protocol simply needs to maintain two port states per port instead of one and perform all the STP calculations twice (once with the classical cost and once with the quantum cost).



(a) Classical Network. (b) Quantum Network. (c) Q-STP Network.

Figure 1.6: Example of Q-STP.

1.4 Quantum Protocol

This section presents the design of a quantum communication protocol tailored for operation within classical Ethernet LANs outlined in [4]. The protocol relies on classical control messages to coordinate the generation, transmission, and manipulation of qubits across the network.

The protocol stack includes several components: a classical signaling mechanism for qubit transmission, a lightweight discovery protocol for identifying quantum-capable nodes, and a path establishment procedure to ensure that entanglement swapping operations consistently traverse the same intermediate bridges. Finally, the protocol defines the sequence of classical messages required to coordinate sequential entanglement swapping, ensuring proper timing and synchronization across all involved nodes.

1.4.1 Transmission of Qubits

In the transmission of a qubit from one switch to another the classical channel has to be used for control messages (acknowledgements for example). In [4] a reference case is considered where the classical channel is used to send a *Quantum Header* whereafter the qubit is transmitted over the quantum channel. The header has to be sent over the classical channel, as it cannot be embedded in the qubit itself. Due to the no-cloning theorem [1], extracting or reading a header from a quantum packet would disturb the quantum state, which makes such an approach unfeasible. Therefore the quantum header has to be sent on the classical channel.

If both the header and qubit arrives successfully at the target receiver an acknowledgement is sent back. Using this scheme if an error occurs to any of the packets the complete sequence must be restarted, which is not ideal if error probabilities are high. Therefore [4] proposes a new scheme where a handshake is performed first before transmission of the qubit. The handshake consists of the first switch sending a *transmission request* (that includes the quantum header), which the second switch answer with an acknowledgement. At this point the qubit can be transmitted like in the reference case. This transmission scheme requires more messages to be transmitted to get a higher resistance against errors, since less messages would have to be retransmitted in case of an error. A visual example of the handshake procedure can be seen in Figure 1.7.

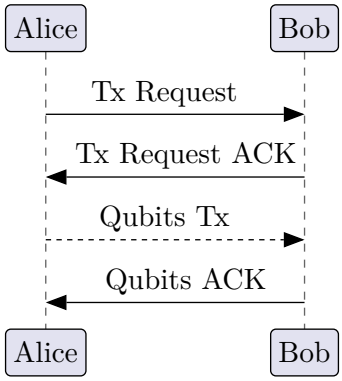


Figure 1.7: Example of the handshake and transmission of qubits.

1.4.2 Quantum Protocol Header

As mentioned previously transmission of a quantum header is required in the classical channel. In [4] a 20 byte long header is presented with the structure shown in Figure 1.8. The header consists of 4 byte sequence and acknowledgment sequence numbers to help with identification and sorting of packets, a 7 bit message type, a 1 bit ACK flag, a 8 byte E2E Entanglement Identifier to identify each entanglement between edge users, a 2 byte Level Number to track progress of ongoing entanglement-swapping, and Token ID to transfer said ID after successful entanglement-swapping.

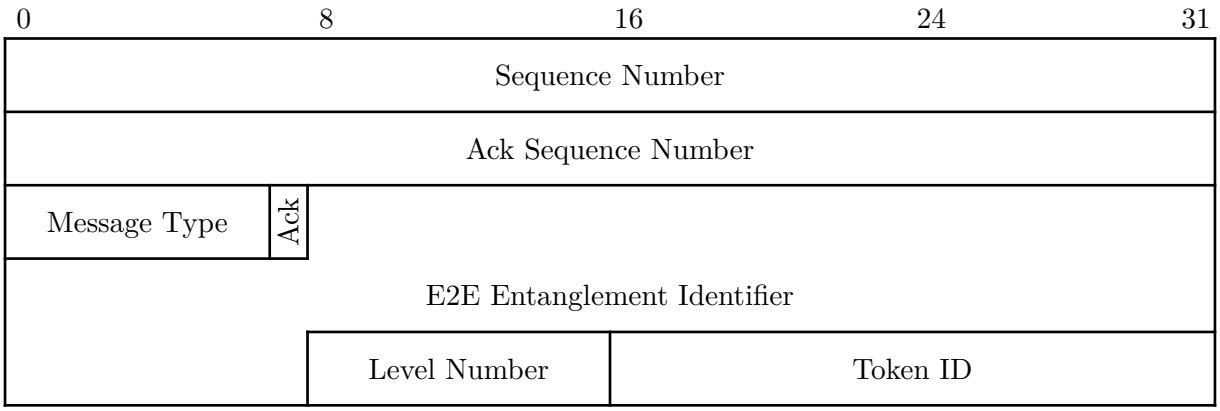


Figure 1.8: Quantum Protocol Header.

Examples of possible message types will be given and explained in the following sections. The level number and token ID will also be explained later on.

1.4.3 Discovery Protocol

When a node wants to start a transmission it should assure that the destination exists and is available. This is done via the Discovery Protocol, outlined in [4], which works similarly to protocols like the Internet Control Message Protocol (ICMP). The paper assumes that the transmitting node knows the MAC address of the receiver. The node then sends a *Discovery Request* using an Ethernet port which is associated with a quantum port. If the receiving switch has no information about the final destination it broadcasts the requests through all its Ethernet port that have corresponding quantum ports. When the request arrives at the destination a *Discovery Reply* is sent back as a reply. This way all the switches can update their MAC tables ensuring efficient delivery of frames in the next protocols. The Discovery protocol is shown in Figure 1.9.

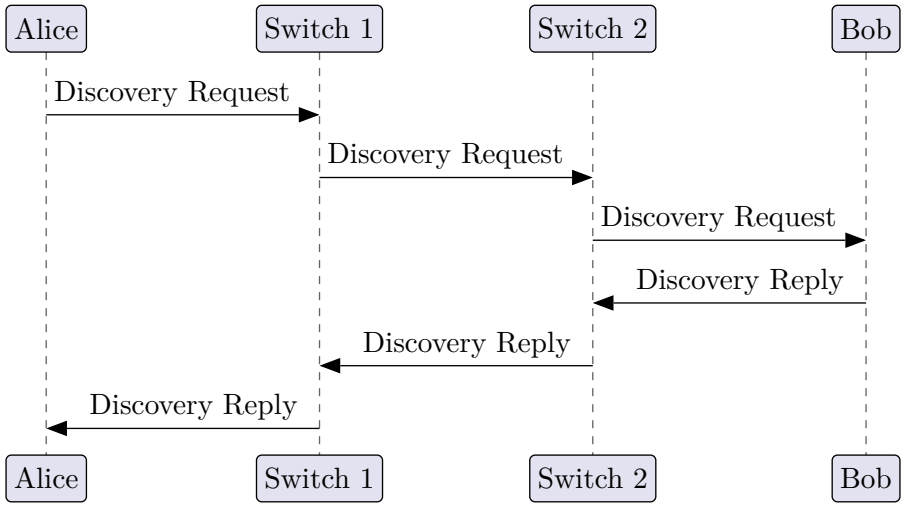


Figure 1.9: Example of the Discovery Protocol.

1.4.4 Path Establishment Protocol

Since the entanglement-swapping procedure requires a exact knowledge of the switches involved, a path establishment protocol is outlined in [4]. The Path Establishment Protocol creates a virtual circuit between the two users and allocates the necessary classical and quantum resources required for the entanglement.

When a node wants to perform entanglement with another node it sends an *Establishment Request* consisting of an *entanglement ID* and its own MAC address. Switches on the path to the destination node then overwrite the MAC address field and retransmits the request to the next switch. When the request arrives at the destination an *Establishment Reply* is sent back with the same entanglement ID and using the same manipulation of the MAC address field. This allows the switches to record which interfaces to use for a specific entanglement ID in an *Entanglement Table*. An example of this can be seen in Table 1.4, with arbitrarily chosen MAC addresses for origin and destination and generic port names for the ports.

The links in the Entanglement Table have to be periodically checked using control messages. If a switch thinks the established path has been broken it transmits an *Establishment Interrupted* message to the users, so the path establishment can be restarted.

ID	Origin	Destination	Origin Port	Destination Port
27	00:00:00:00:00:09	00:00:00:00:00:0b	eth-1 qeth-1	eth-2 qeth-2
49	00:00:00:00:00:0c	00:00:00:00:00:0b	eth-3 qeth-3	eth-2 qeth-2

Table 1.4: Entanglement Table of a Switch.

1.4.5 Swapping Protocol

The sequential swapping protocol outlined in [4] has two important concepts, *level* and *token*. These were mentioned in the header in Section 1.4.2. For one instance of end-to-end swapping two tokens are generated, one in each of the switches closest to the edge users. The tokens are then sent inwards in the virtual circuit after a successful swapping occurs. If an errors occurs in the swapping the tokens must either be sent back to the original switches or restarted there. If a switch has both tokens then it knows that it has to perform the final swapping.

In some scenarios with an equal number of intermediate switches, a situation could arise where two switches adjacent to one another have the two tokens. In this case the switch closest to the origin performs the next swap. The *level* is a number that shows how many sequential swappings have occurred in each direction of the circuit. The highest *level* number (V) can be calculated by [4]:

$$V = \left\lfloor \frac{S}{2} \right\rfloor + 1 \tag{1.4}$$

where S is the number of switches in the virtual circuit. The *level* (denoted by v) is used in case of swapping failure to notify the next switch and the previous v switches.

Once a virtual circuit has been established between two edge nodes the entanglement swapping protocol can begin. Each switch has to generate pairs of entangled qubits and share them with its neighbors using the scheme outlined in Section 1.4.1. Once all the qubits have been shared the two switches possessing the tokens (the switches closest to the edge users) begin the swapping procedure by sending a *Swapping Request* to the next switch which is responded to with a *Swapping Reply*, hereby informing the next switch that the swapping will occur. If the swapping is successful the token is transferred to the next switch using *Token Transfer* and *Token Ack* messages. When the last switch in the chain has completed the swapping it sends a *Swapping Complete* message to the neighboring switches that respond with a *Complete Ack* message before forwarding the *Swapping Complete* message towards the next switch. A visual example of the swapping protocol when no swapping failures occur can be seen in Figure 1.10.

If an error occurs in the swapping process the switch responsible for the swapping transmits a *Swapping Error* to the relevant switches according to the current *level*, since the qubits will have be regenerated, shared, and the swapping process has to start over.

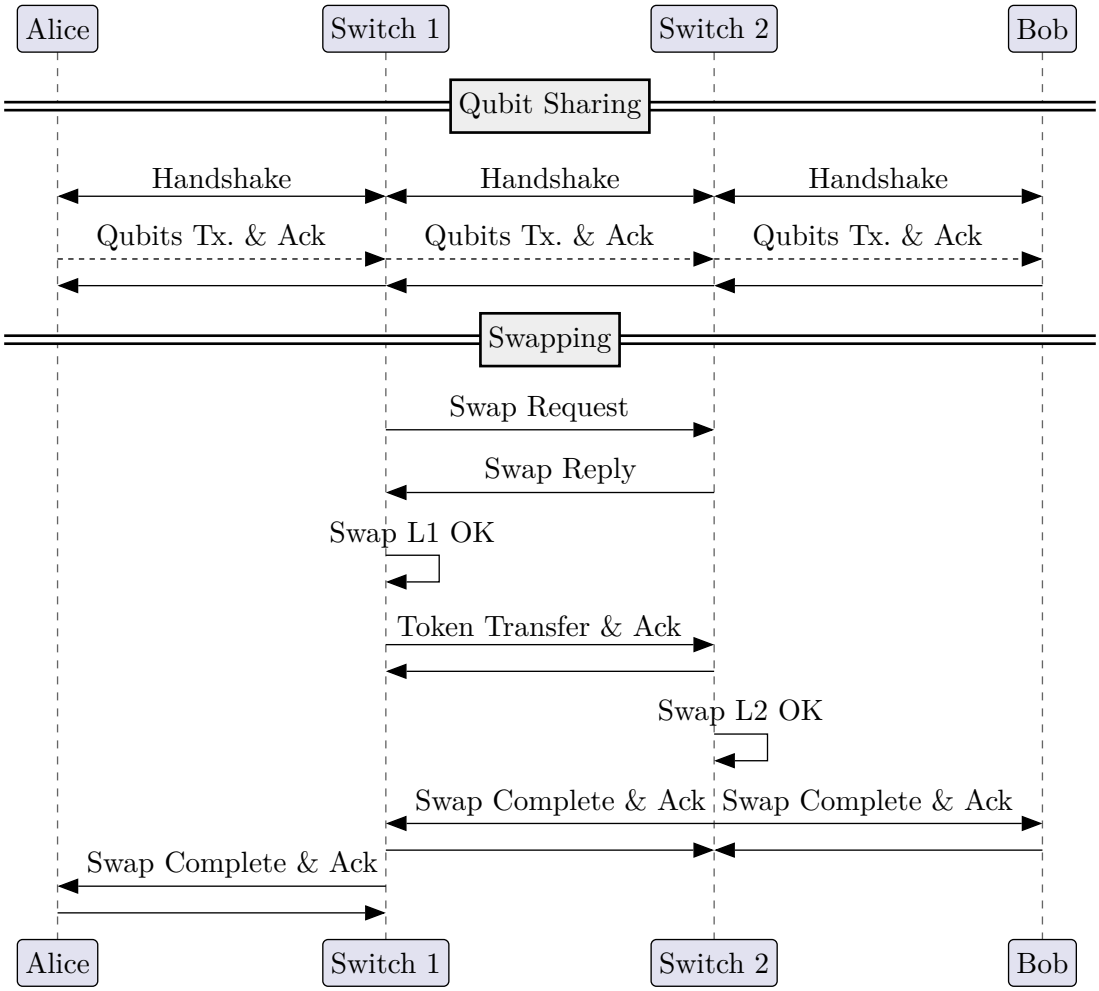


Figure 1.10: Example of successful end-to-end swapping between Alice and Bob with 2 intermediate switches using the Swapping Protocol.

Chapter 2

Simulation

In this chapter the setup of the simulation environment and different test cases will be explained. The simulation tool that will be used is the discrete-event network simulator Network Simulator 3 (ns-3), since it is open-source and widely used in research of network protocols and systems. It contains many different protocols like Ethernet and Wi-Fi making it a good option for this project [14].

2.1 Simulation Setup

In this section a high level overview is given of the different ns-3 components used. This includes the setup of classical and quantum links and the custom Q-STP, and Quantum Protocol applications.

One concept within ns-3, known as a node, can be thought of as a basic computing device, that things like applications, protocol stacks and peripheral cards can be added to [15]. All switches and user nodes are represented by this type with different configurations depending on the use. Nodes can be created by using the NodeContainer:

```
1 NodeContainer nodes;
2 nodes.Create(2);
```

Applications can be implemented in ns-3 by the use of the *Application* class [15]. Q-STP and the Quantum protocol handlers in switches and user nodes are implemented by specializing this Application class.

2.1.1 Link Setup

For the simulation two different types of links are needed: Classical links and Quantum links. In ns-3 terms this is called a *Channel* represented by a class of the same name. There exist many different specialized channel types in ns-3 such as [15]:

- CsmaChannel
- PointToPointChannel
- WifiChannel

The most interesting protocol for connecting switches is Ethernet as explained in the previous chapter, which uses Carrier Sense Multiple Access (CSMA). Therefore the *CsmaChannel* is used for classical connections between switches. The CsmaChannel will also be used to connect user nodes to switches for simplicity, but this could in the future be swapped with the WiFiChannel.

Since Quantum networks are not natively supported in ns-3 and the quantum links themselves are not the focus of the report, the quantum links are emulated using the PointToPoint channel. Small packets will be sent over the channel to emulate the transmission of qubits.

Links are created using *Helper* objects in ns-3 and the helper is also used to configure the links :

```
1 CsmaHelper csma;
2 csma.SetChannelAttribute("DataRate", StringValue("1Gbps"));
3 csma.SetChannelAttribute("Delay", TimeValue(MilliSeconds(1)));
```

To connect a node to a channel a so-called *NetDevice* has to be added to the node. This NetDevice represents a real-life network card and its drivers, and a node can therefore be equipped with multiple NetDevices to connect it to multiple channels. There exists different NetDevice types used to connect to different types of channels like CsmaNetDevice and PointToPointNetDevice [15]. To create the correct NetDevices for the Helper has to be used again. The following example shows how to connect 2 nodes using the CSMA Helper:

```
1 NetDeviceContainer devices;
2 devices = csma.Install(nodes.Get(0), nodes.Get(1));
```

The created NetDevices for that link are then stored in the NetDeviceContainer and they can later be retrieved into other containers to create per node NetDeviceContainers.

2.1.2 Q-STP

In ns-3 there is an implementation of a *Bridge Network Device* which aggregates multiple NetDevices and implements the forwarding and learning process from 802.11D explained in Section 1.2.4. However, the BridgeNetDevice does not implement the Spanning Tree Protocol (STP) part of 802.11D [16], which makes it unsuitable as is for this project.

Therefore a custom application has been made to implement the Quantum-Spanning Tree Protocol (Q-STP). This application takes a BridgeNetDevice and overwrites its function for receiving packets. The Q-STP application contains all the relevant data to perform the Q-STP protocol, such as tracking costs, root etc for both classical traffic and quantum traffic as explained in the previous chapter. To distribute the required data empty packets are generated according to STP principles and sent with the Ethernet header and a custom BPDU header with the BPDU fields explained in Section 1.3 and Section 1.3.2.

To properly use the two spanning trees it has also been necessary to maintain two Medium Access Control (MAC) tables and functions have been made to learn MAC addresses upon receiving frames and to get the correct output port for any incoming address based on the type of frame. Using this frames can be forwarded on the correct ports for both quantum related traffic and classical traffic.

To use the custom application in ns-3 one first has to create a BridgeNetDevice on the node. Then the application can be initialized by:

```
1 Ptr<QSTP> qstpApp = CreateObject<QSTP>();
```

Then the BridgeNetDevice needs to passed to the application using a set function, along with the quantum ports for the switch stored in a NetDeviceContainer and a boolean value indicating whether the application should proceed in Q-STP mode or in regular STP mode:

```
1 qstpApp>SetBridge(bridgeDev, quantumPorts , isQuantum);
```

Then the Q-STP application can be added to the node and start and stop times can be set.

2.1.3 Quantum Protocol Handlers

To handle packets of of the Quantum protocol a special handler has been created in the form of an application. Since the functions that have to be carried out by the switches and user nodes are different two distinct applications have been created called: *QP-Switch* for the switches and *QP-App* for the user nodes. Both apps work with frames with the EtherType 0x0477 and a custom packet header like described in Section 1.4.2.

2.1.3.1 QP-Switch Application

When the Q-STP application receives a frame intended for it, the frame is sent up the stack to this handler, where the QP header is unpacked. From the header the application reads the message type and acts according to it. The application is also responsible for controlling sequence numbers, acknowledgements and retransmission of packets if needed.

The QP-Switch application contains an *std::vector* of a struct called *entanglement_data*, which stores the relevant ports and addresses obtained from the path establishment protocol. The application also maintains connection states for each connection to track sequence numbers and acknowledgements. Retransmisions are handled by storing unacknowledged frames in a buffer and retransmitting them based on a timeout value.

To track entanglement swapping progress a struct is used to store relevant information like tokens, level, if neighbors have a qubit entangled with the switch and if a swap is currently occurring. This is done to make sure switches do not attempt to perform swapping under incorrect conditions that do not match required swapping conditions. This QP-Switch application is also responsible to performing the entanglement swaps, which has been simplified to being a success or failure by drawing from a uniform distribution and comparing the value to a preset *swap failure probability*.

The QP-Switch application is added to switches in the same fashion as the Q-STP application, and requires the same BridgeNetDevice and *quantumPorts*:

```
1 Ptr<QpSwitch> qp_switch = CreateObject<QpSwitch>();
2 qp_switch>Setup(bridgeDev, quantumPorts);
3 switches.Get(i)->AddApplication(qp_switch);
```

2.1.3.2 QP-App Application

The QP-App that is used for the user nodes is quite similar to the QP-Switch application. It takes as setup parameters the NetDevices for the nodes classical and quantum ports, and if the node is intended to start an entanglement it also requires the MAC address of the intended target of the entanglement and a boolean value set to True that indicates that it should start the entanglement. The code to use the application follows the same structure as for the QP-Switch application:

```
1 Ptr<QpApp> app = CreateObject<QpApp>();
2 app->Setup(classicalPort, quantumPort, macAddress, true);
3 nodes.Get(0)->AddApplication app;
```

2.2 Topologies

Ensuring that a protocol works on different network topologies is extremely important in evaluating it, therefore a set of topologies that are commonly used in research will be used for testing. Some commonly used topologies are outlined in [17], which, along with additional topologies, will be used in this report.

Line Topology

Probably the most simple topology is the line topology. It consists as the name suggests of N switches set up in a line. This means that the edge switches have one link to another switch while the inner switches have two links each. If a frame arrives at the first switch destined for a node connected to the last switch it must traverse the whole line to reach it. This type of topology however provides no redundancy since a single link failure essentially cuts the network in two. A line topology can be seen on Figure 2.11.

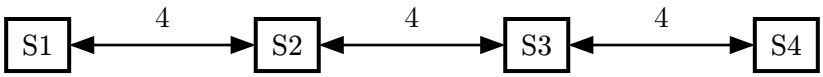


Figure 2.11: Line Topology with 4 Switches.

Ring Topology

A ring topology consists of N nodes connected in a circular manner, i.e. each node has exactly two connections to other nodes. This is similar to the line topology except that the two edge switches are connected to each other. The circular structure of this topology means that there is a loop, which also provides a level of redundancy incase of link failure. A ring topology can be seen on Figure 2.12.

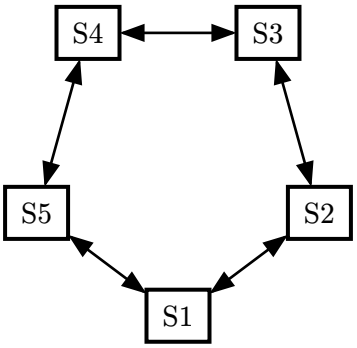


Figure 2.12: Ring Topology with 5 Switches.

Star Topology

The star topology consists of a central hub that each of the *leaf* switches are connected to. Any traffic going from one leaf switch to another therefore has to go through the central hub. Multiple star topology networks can also be connected by adding a link between their central hubs thereby creating a more complex network. A simple star topology can be seen on Figure 2.13.

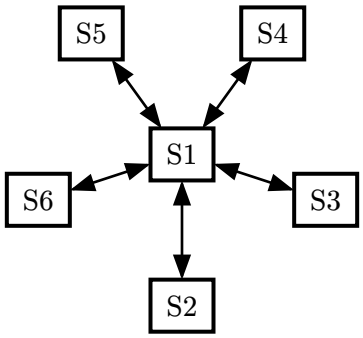


Figure 2.13: Star Topology with 5 Leaf Switches and a Central Hub Switch.

Grid Topology

A grid topology is a topology where switches are connected in either a 2D or 3D grid structure. The edges of the network can be connected to each other creating a torus. A network of this types contains multiple loops that will have to be handled by STP. This topology lends itself nicely to the quantum topology being comprised of a subset of the links in the classical topology. This can be used to demonstrate the effect Q-STP could have. An example of this that will be used later can be seen on Figure 2.14, where there are no vertical quantum links between the middle switches.

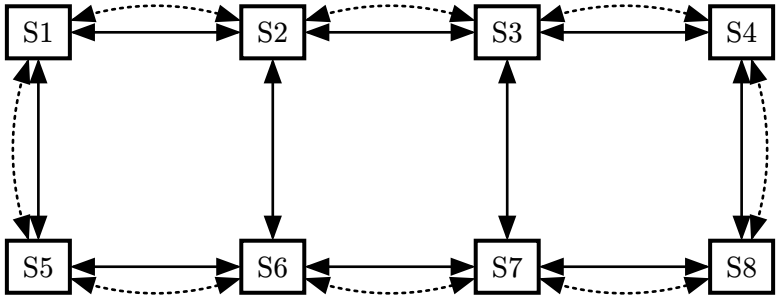


Figure 2.14: Grid Topology: 2 Rows × 4 Columns. Classical links as solid lines and quantum links as dotted lines.

2.3 Network Congestion

Any real quantum network would have many users transmitting both classical data and quantum data concurrently. Therefore it would be interesting to see how the implemented protocol would perform in a network with congestion. To simulate congestion a grid network like described in Section 2.2 with 2 rows and N columns will be used. A user node will be connected to each of the edge switches. The user nodes will then send UDP traffic to each of the other nodes using the ns-3 built-in OnOff application. The parameters for the OnOff application can be seen in Table 2.5. The On (how much time is spent sending) and Off (how much time is spent idling) times are drawn from uniform distributions.

Parameter	Value
Data Rate	1 Mbps
Packet Size	1024 Bytes
On time	$U(0\text{ ms}, 200\text{ ms})$
Off time	$U(100\text{ ms}, 500\text{ ms})$

Table 2.5: OnOff Application Parameters.

2.4 Simulation Timeline

In this section timelines of simulations will be presented to give a better understanding of how simulations unfold. All simulations start off with the Q-STP application starting on the switches, which keeps running until the simulation finishes. At some predetermined time t_{start} (e.g. at 25 s) the Quantum Protocol application starts on the switches and the nodes. One of the nodes is initialized to start the QP process by sending a *Discovery Request* at some time after the start of the application. After the node that sent the first *Discovery Request* receives the *Reply* signalling that the other node is running, it schedules the transmission of a *Establishment Request* either immediately or with a delay. When a switch or node receives a correct *Establishment Reply* it will send a *P2P Request* to its right neighbor in order to distribute qubits between all the participants.

There are two options for the starting of the swapping protocol. Option 1 is that all the switches start the swapping at some predetermined time t_{swap}. This guarantees that all switches start the swapping at the same time, which is nice for measuring the latency of the protocol but it might be unrealistic to have such tight synchronization in a distributed network. Option 2 is for the switches to start the swapping process individually based on the reception of qubits and *Qubit ACK* frames, i.e. when an edge switch has qubits entangled with both neighbors it can start the swapping, which requires no synchronization with the other switches. This could cause one edge switch to start swapping earlier than the other edge switch.

Figure 2.15 shows an example timeline of a simulation using Option 2 as described above. The width of the bars i.e. the time spent in that protocol stage is arbitrary, since it might differ depending on topology, link speed and number of switches. From here on out Option 2 is what is used.

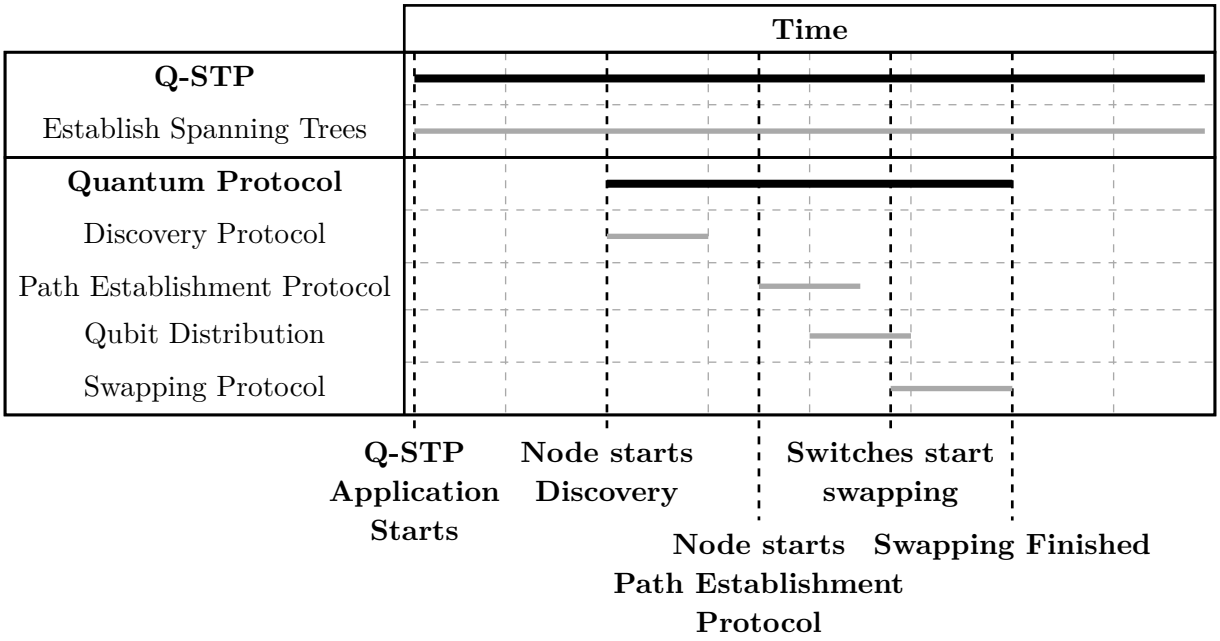


Figure 2.15: Example Simulation Overview Timeline with Option 2.

2.5 Performance Indicators

To evaluate the performance of the implemented protocol(s) some metric has to be used. In this project it will be latency. To make it clear what is meant by latency it will be explained now. There are three interesting latencies in this context, namely the latency of each stage of the Quantum Protocol:

- Discovery Latency
- Path Establishment Latency
- Swapping Latency

The *Discovery Latency* is the latency of the discovery protocol. It is defined as the time it takes from the transmission of a *Discovery Request* until the reception of *Discovery Reply*, and it is therefore easy to measure. The *Path Establishment Latency* is equivalent to the *Discovery Latency* just for the *Establishment Request* and *Reply* instead. Both these are measured at the node starting the protocols by using the ns-3 *Simulator::Now()* function to obtain the simulation time just before transmission and just after receiving.

The *Swapping Latency* is a bit more tricky since it involves multiple switches. For this project it will be defined as the time from when the first switch is ready to perform swapping until the last *Complete Ack* frame is received. To measure this in the simulation all switches will log when they are ready to perform a swap, i.e. ready to send *Swap Request* in case with >1 switches. In the case with only 1 switch it will be calculated from when the switch knows it has both tokens and is ready to perform the swap. This is used to mark the start of the protocol. The protocol then ends when the last *Complete Ack* is received. Calculating the difference between these recorded timestamps gives the *Swap Latency*.

2.6 Test Scenarios for Quantum Protocol

To evaluate the performance of the implemented quantum protocol, a series of simulation tests have been designed. Each test targets a specific aspect of the protocol's behavior, with a particular focus on latency under varying network conditions. The scenarios differ in terms of network load, and protocol parameters in order to assess the protocol's robustness and responsiveness. The following subsections describe the individual test cases and the rationale behind each configuration.

To ensure statistically meaningful results, each simulation scenario is executed multiple times. This is done by incrementing the run number set with ns-3's SeedManager function *SetRun()*. The reported latency values represent the average over 100 independent runs. The tests will be denoted by *1.a*, *1.b* etc. for each stage of the protocol so it is easy to refer back to in the following chapters. In tests where the number of intermediate switches is used it means the number of intermediate switches between the two users performing the end-to-end swapping.

The tests in this thesis are divided into two distinct categories: Fixed Link Delay and Fixed Total Distance. *Fixed Link Delay* refers to the case where each link in the network has a fixed delay i.e. 1 ms in this report. Adding more switches and therefore more links in this case should increase the total latency of the protocols. On the other hand *Fixed Total Distance* means that the total delay from user 1 to user 2 will be fixed. Adding more links then decreases the delay of the individual links. To work with the ns-3 links as described previously a delay value is needed for the links. In this case a distance is converted to delay, assuming that all the links are fiber-optic, the speed of light is approximately $2 \cdot 10^8$ ms/s [18] which is used to approximate the delay for a certain distance. All the latency tests will be performed on a grid topology as shown in Figure 2.14. The Discovery/Path Establishment protocols and the Swapping protocol are tested separately, since they are do not have an effect on each other.

It has to be mentioned that no ns-3, by default, does not consider processing time of an application i.e. any task performed takes 0 simulation time. This is important to keep in mind as it will have an effect on some of the test results.

2.6.1 Discovery Protocol and Path Establishment Protocol

The first test (1a) that is performed for the Discovery Protocol is the Fixed Link Delay test where the discovery protocol is the only traffic (not counting BPDUs) on the network. The expected latency in this case assuming the hardware of the switches, the back-off time of transmissions and the length of the links is identical is according to [4]:

$$T_{\text{disc}} = 2LT_{\text{req}} \tag{2.5}$$

where L is the total number of links, $2(\#\text{ of switches} + 1)$, and T_{req} is the time delay for transmitting a request (or ACK) frame. The expectation for this test is that the the latency of the Discovery Protocol in simulations will be the the same as the theoretical latency. The latency of the Path Establishment Protocol is expected to be approximately the same as that of the Discovery Protocol, since it involves the same number of messages.

The second test (1b) is with the same setup as the first (Test 1.a) with the exception that competing classical traffic has been added. How the competing traffic is modelled is described in Section 2.3. It is expected that the competing traffic will cause the latency of the Discovery and Path Establishment protocols to increase. This is because the competing traffic will occupy the same queues as the QP traffic.

The third test (1c) is of Fixed Total Distance type. The test is conducted for predefined distances of 200 m, 400 m, 600 m, and 1000 m. In this case it is expected that the latency remains mostly unchanged with different number of switches as the link delay should be the major contributor to the latency. For this test no external traffic will be generated.

2.6.2 Swapping Protocol

The swapping protocol is also tested on the same two types of tests as the previous case i.e. Fixed Link Speed and Total Distance. However a new parameter is defined for the swapping protocol. Namely the probability of swapping failures occurring when performing the swap. As mentioned in Section 1.4.5 this causes the protocol to have to redistribute qubits and redo swaps. Tests will be performed where this probability is set to 0% i.e. no swapping failures will occur and where it is set to values >0%.

The first test (2a) of the swapping protocol is on Fixed Link Speeds with no swapping failures. The purpose of this test if to see how the latency evolves with more links. This test is also performed with and without external traffic to see how it effects the protocol latency. It is expected that the latency for both with and without external traffic will increase as more switches are added.

The second test (2b) is to test how the latency evolves when the total distance is static and no swapping failures occur. This is to test if having more switches and therefore needing to perform more swaps effects the latency in a meaningful way. The expectation is that more intermediate switches will slightly increase the latency. It is not expected to be a huge difference since the swapping starts from both sides around the same time and swaps therefore occur concurrently. This test will be performed for multiple total distances like in *Test 1c*.

The last test (2c) is also with the total distance fixed as in the previous test. However, in this test the swapping failures will be enabled. This test will be conducted on the 1000 meter fixed distance with different values for the failure probability: 20%, 40%, 60% and 80%. The purpose of this test is to show how the swapping failures effect the latency. It is expected that a higher number of intermediate switches will increase the latency for all failure probabilities above 0%, since needing to do more swaps increases the probability that one of them will fail. It is also expected that the increase with more switches will be more severe for higher error probabilities.

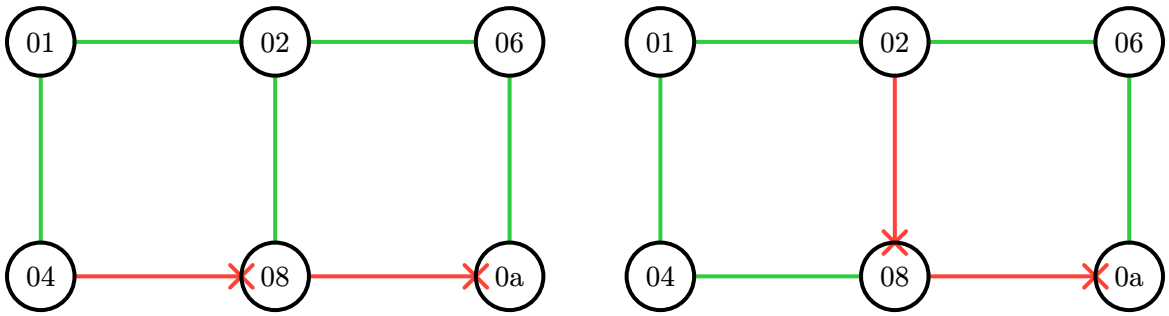
Chapter 3

Results

In this chapter the results of different simulations intended to show the performance of the different protocols will be presented. It will start of with showing the spanning trees generated by the Quantum-Spanning Tree Protocol (Q-STP) protocol for different topologies, followed by the latency of the *Discovery Protocol* and *Path Establishment Protocol* in different scenarios explained in the previous section. Lastly the latency of the *Swapping Protocol* in scenarios with and without swap failures will be presented.

3.1 Q-STP

To demonstrate that Q-STP works and performs the desired actions in simulations, some scenarios will be presented where Q-STP works differently than Spanning Tree Protocol (STP). The first one is the grid topology shown on Figure 2.14, with no vertical quantum link between the middle switches. For simplicity a 2x3 Grid will be used here. Figure 3.16 shows both the classical spanning tree and the quantum spanning tree computed by the protocol. The switches are denoted by the last 2 hexadecimal values of their Medium Access Control (MAC) address. The links are colored either green for active or red for "blocked" meaning one of the ports on that link is in the discarding state. All classical links have the same data rate and delay. Figure 17a shows the tree generated by the classical STP procedure which is used for normal traffic, while Figure 17b shows the new quantum tree that is found by Q-STP in addition to the classical tree. The quantum tree shows that the protocol can adjust the tree based on the missing quantum link as expected.

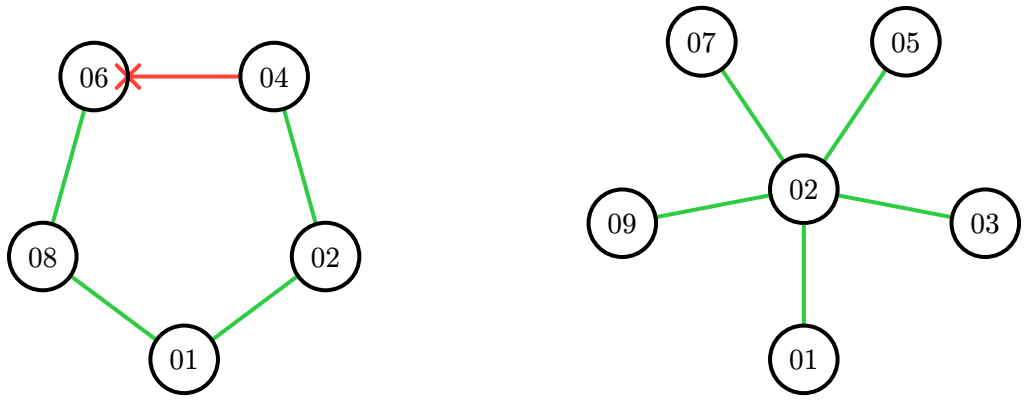


(a) Classical Spanning Tree.

(b) Quantum Spanning Tree.

Figure 3.17: Simulation of Q STP on a 2x3 Grid.

Figure 19a and Figure 19b show the output of Q-STP on ring and star topologies in the simulations. As explained earlier the ring topology contains one loop which is correctly blocked by the protocol, while the star topology contains no loops and therefore all links are active and forwarding. Tests on the line topology show the same where no links are blocked.



(a) Ring Topology. (b) Star Topology.

Figure 3.19: Simulation of Q-STP on ring and star topologies.

3.2 Discovery Protocol and Path Establishment Protocol

The results of the three tests for the Discovery Protocol and the Path Establishment Protocol will now be presented. The first is *Test 1a*, i.e. fixed link delay with no external traffic. The result of that test can be seen on Figure 3.20, where the average measured latency and the theoretical latency from Equation (2.5) are plotted together. The latency of the Path Establishment Protocol has not been plotted since it was the same as that of the Discovery Protocol. The plot shows that the measured latency is equal to the theoretical latency as was expected.

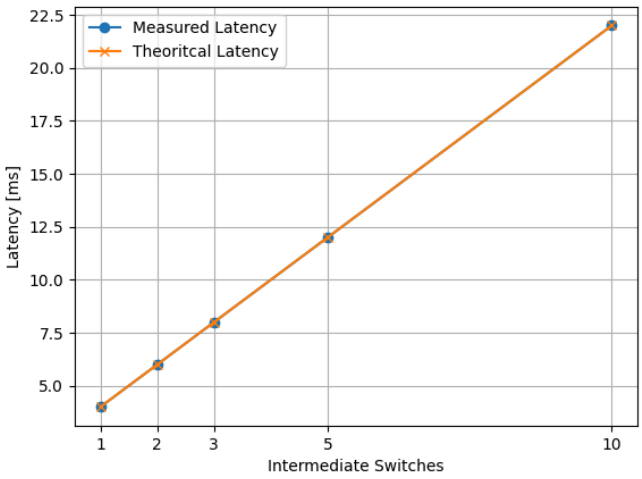


Figure 3.20: Latency vs Number of Switches for the Discovery Protocol.

The next test, *Test 1b*, was to see how the latency evolves when external traffic is introduced in the grid network. The result of the test can be viewed on Figure 3.21, where the latency of the protocols has again been plotted with the theoretical latency.

Here it can be seen that the competing traffic causes the latency of both protocols to increase. It also shows that the difference between the measured latency and the theoretical latency increases as more switches are added.

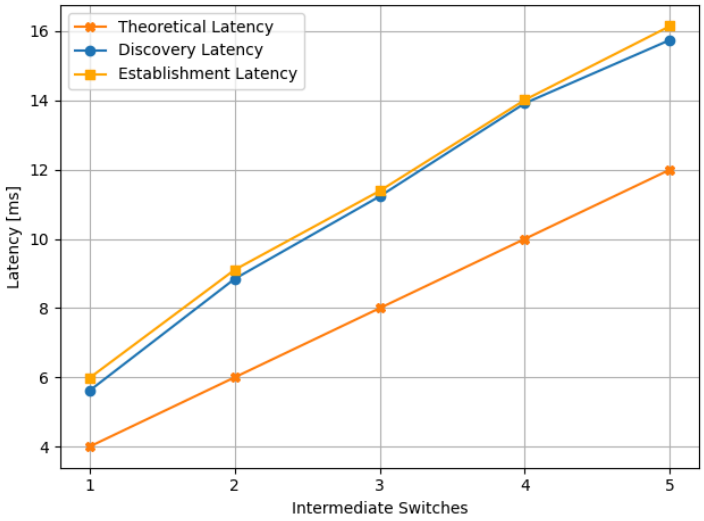


Figure 3.21: Latency vs Number of Switches for the Discovery Protocol with competing traffic.

For the third test namely *Test 1c* with the total distance between the users fixed, the results can be seen on Figure 3.22. As expected when the total distance is larger then the latency of the protocol is also larger as the frames have to travel further. The latency also remains stable when adding more switches showing that it is the link delay dominating the latency in the simulations. This is due to the way Network Simulator 3 (ns-3) considers all the code to be run in a single time instant i.e. forwarding the frames takes no simulation time.

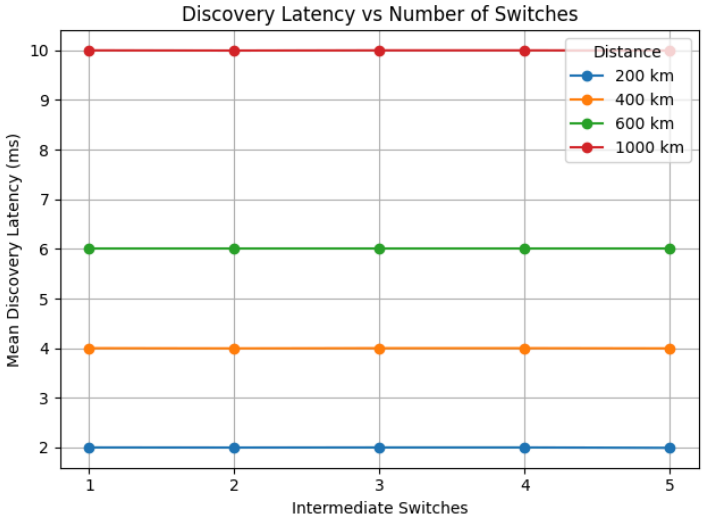


Figure 3.22: Latency vs Number of Switches for the Discovery Protocol with competing traffic. No simulated processing time in switches.

3.3 Swapping Protocol

In this section the results of the simulations of the three tests for the Swapping Protocol will be presented. The first of the tests, *Test 2a* is the case with fixed link delay and varying number of switches with 0% swap failure probability. Figure 3.23 shows the result of the simulation for both the case with no external traffic (Baseline) and the case with external traffic (Congested). The plot shows that for the baseline there is a large increase in latency when going from 1 switch to 2 switches, likely due to contention of the channel between the 2 switches. With more than 2 switches the latency increases slightly as expected. In the case with congestion the same pattern emerges of a large increase from 1 switch to 2 switches whereafter the latency increase steadily after.

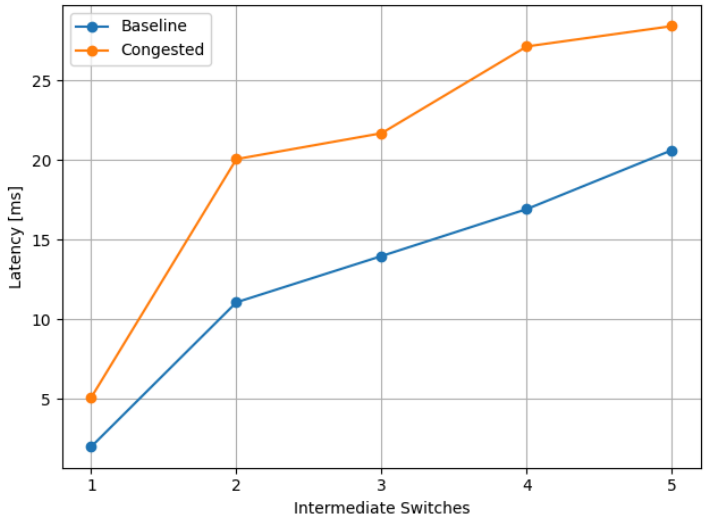


Figure 3.23: Latency of the Swapping Protocol with varying number of intermediate switches.

The second test, *Test 2b* was with the fixed total distance and no swapping failures. As explained previously this was tested for multiple total distance values. Figure 3.24 shows the result of the simulations. It shows (like in the case of the Discovery Protocol) that a larger total distance increases the latency. For each of the distance values tested there is also a significant increase in latency when going from 1 switch to 2 switches like in Test 2a. Adding more switches beyond two does not seem to cause any significant changes to the latency. This could be due to the behavior of ns-3, where the processing of frames in switches occur in 0 simulation time, resulting in extremely fast retransmits of frames. The latency at 2 intermediate switches seems to be slightly higher than for more than 2 switches, which is most likely due to contention of the shared channel between the two switches if they both attempt to perform the swapping simultaneously.

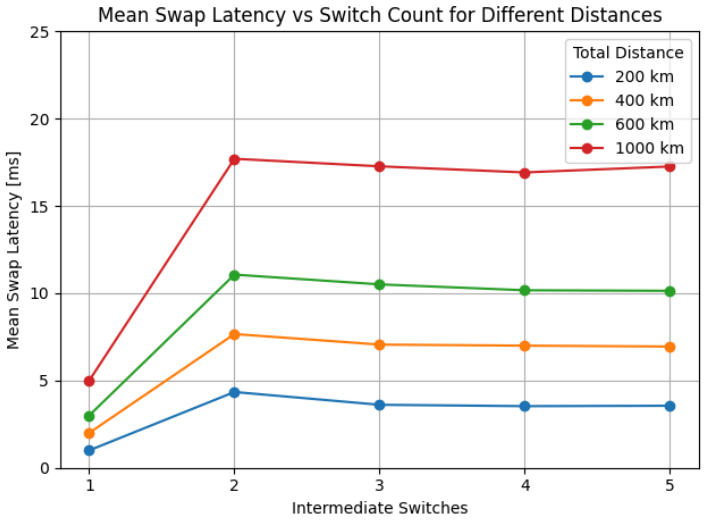


Figure 3.24: Swapping Protocol.

The third and final test for the swapping protocol, *Test 2c* involves adding swapping failures to the total distance test. As explained this has been done with various levels of swapping failure probability. The results can be seen on Figure 3.25, which has been plotted with a logarithmic Y-axis, since latencies got very large for the 80% swap failure probability test. Compared to the test with no swapping failures we shows that having more switches increases the latency as was expected. It also shows that a higher failure probability leads to a higher latency increasing when considering more switches.

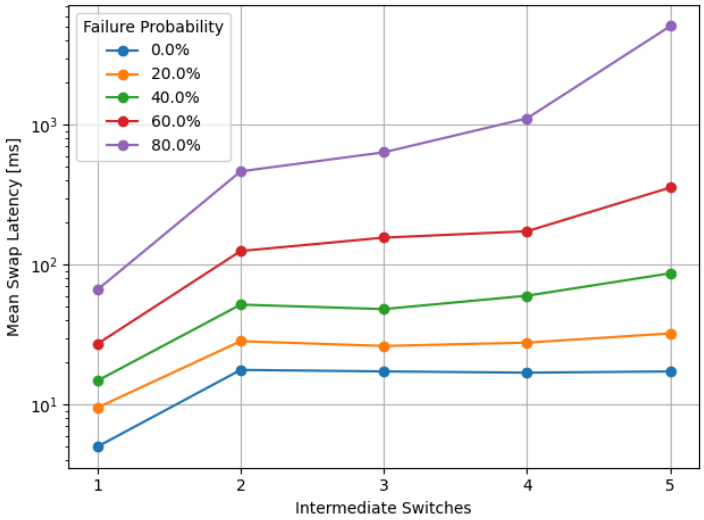


Figure 3.25: Latency of Swapping with different swap failure probabilities for 1000 km total distance.

Chapter 4

Discussion

This chapter reflects on the results obtained from the simulation-based evaluation of the protocol, with a focus on interpreting their significance and examining the limitations of the simulation model. By analyzing both the strengths and constraints of the simulation, this chapter provides a critical perspective on the protocol's practical viability and outlines directions for future improvement.

4.1 Limitations of the Simulation

While the simulations conducted in this project provide useful insights into the behavior of the implemented protocol, several limitations must be acknowledged. These stem primarily from simplifications made to ensure feasibility within the ns-3 simulation environment, as well as from the inherent abstraction of complex quantum phenomena. The following subsections outline key limitations related to the modeling of link types, quantum operations, and the scope of the simulation.

4.1.1 Use of Ethernet Links

In the simulations performed in this project, all classical links were modeled as Ethernet/ CSMA links, and all quantum links as Point-to-Point links. This choice was made for simplicity and practicality, allowing for more controlled conditions and clearer isolation of the protocol's behavior. However, in real-world deployments, networks are unlikely to be limited to such idealized, wired links. Wireless communication is often necessary, especially in distributed or mobile systems, and it introduces several challenges that are not captured in this simulation model. Unlike wired links, wireless links, such as Wi-Fi, are subject to variable latency, interference, signal attenuation, and higher error rates. These factors can lead to increased packet loss, jitter, and retransmissions, which may affect the timing and overall performance of the protocol. As such, the results obtained in this simulation may represent a best-case scenario, and further studies incorporating wireless link models would be necessary to fully evaluate the protocol's robustness in practical environments. This could be by the use of ns-3′s Wi-Fi links and modules as mentioned earlier.

4.1.2 Simplicity of Quantum Links and Operations

As just mentioned the quantum links were modelled using the Point-to-Point links of ns-3. This was done as a straightforward way to represent high-fidelity connections between the nodes for quantum communication. However, it has to be noted that this is an abstraction and does not model actual quantum operations. As a result, the simulations do not account

for the delays, failure probabilities, or noise that would be introduced by real-world quantum hardware. Instead, the quantum link was treated as an idealized channel with deterministic behavior, allowing the focus to remain on the classical control protocol layered on top. Like the quantum links, as mentioned in Section 2.1.3.1, the entanglement swapping operation was simplified using a probabilistic approach. This abstraction allowed for the inclusion of imperfect operations in the protocol without requiring a detailed physical model of quantum behavior. This abstraction allowed testing of how the protocol handles failures of the entanglement swap and how it affects the latency of the protocol. Further extensions of the work could involve integration of some quantum network simulator or development of a hybrid simulation approach to better model the behavior of the quantum communication.

4.2 Evaluation of Protocol Performance

The primary objective of this project was to evaluate how well the protocol performs when implemented and tested in a simulation environment (i.e. ns-3), as the original paper did not provide any simulation-based results. This work therefore serves as a first attempt to validate the protocol's expected behavior and latency performance under controlled conditions.

The simulation results demonstrate that the discovery protocol behaved as expected. The swapping protocol also performed well under ideal conditions. However, performance degradation was observed at higher swap failure probabilities, particularly in scenarios involving multiple intermediate switches. In general, the results suggest that using fewer switches leads to lower latency. However, caution should be taken in interpreting this outcome, as it may be influenced by the limitations in how quantum links and swap operations were modeled in the simulation environment.

It is important to note that these findings are based on idealized conditions. The performance benefits observed from using fewer switches may be amplified by the simplification of the quantum links as deterministic and failure rates as fixed rather than dependent on distance. In a more realistic model, where factors such as decoherence and loss are accounted for, the relative performance of different topologies could differ significantly. As such, the current results should be viewed as an initial, optimistic evaluation of the protocol's behavior under best-case conditions.

4.3 Extending STP for Quantum Networks

As discussed in Section 1.3.2, the classical STP was adapted in this work to support quantum links by introducing a new header field to represent the cost associated with a quantum link. The resulting Q-STP protocol maintains two separate spanning trees to more effectively route classical-only and quantum-related traffic, accounting for the differing requirements of each. This was shown to work in different network topologies in the previous chapter.

A key aspect of the original STP that was not implemented is its ability to dynamically react to topology changes. This functionality which is critical in classical networks for maintaining resilience through redundant links remains unaddressed in this implementation of Q-STP. Future work into Q-STP could be to explore how topology change detection, notification and reconfiguration could be extended to accommodate the quantum links along with the classical links. Additionally, alternative metrics beyond the one used here (qubits per second) could be explored as link quality indicators, potentially leading to more nuanced and effective routing decisions.

Conclusion

This thesis set out to evaluate the performance of a recently proposed protocol for facilitating sequential entanglement swapping in local quantum networks (LANs) [4], which had not previously been tested through simulation. The protocol is multi-stage and includes a Discovery Protocol for identifying active quantum nodes, a Path Establishment Protocol to create a virtual circuit between users, and a Swapping Protocol to manage sequential entanglement swapping. The paper also proposed a modified version of the Spanning Tree Protocol (STP) that takes into account both classical and quantum links.

The protocol and the modified STP referred to as Q-STP were implemented in the ns-3 simulator. As the focus of this thesis was primarily on the classical control aspects, quantum links and operations were modeled in a simplified manner. A series of simulations were performed across varying topologies and network conditions to test the different stages of the protocol.

The results confirmed that the protocol behaved as expected under idealized conditions and provided insight into how performance changes with the number of intermediate nodes and increasing swap failure probabilities. The simulations suggest that minimizing the number of switches can help reduce latency. However, this conclusion should be interpreted with caution due to the simplifications in the simulation model, particularly the abstraction of quantum links and operations.

Future work could expand the simulation framework to include dynamic topology changes in Q-STP, incorporate more realistic models of quantum link behavior, and explore additional performance metrics beyond latency. Integration with a dedicated quantum network simulator would also allow for a more accurate assessment of quantum specific behaviors, such as decoherence, entanglement quality, and probabilistic success rates.

Acronyms

ARP	Address Resolution Protocol
BPDU	Bridge Protocol Data Unit
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Multiple Access
CTS	Clear to Send
FCS	Frame Check Sequence
ICMP	Internet Control Message Protocol
LAN	Local Area Network
MAC	Medium Access Control
OUI	Organizationally Unique Identifier
Q-STP	Quantum-Spanning Tree Protocol
RSTP	Rapid Spanning Tree Protocol
RTS	Request to Send
SFD	Start Frame Delimiter
STP	Spanning Tree Protocol
TC	Topology Change
TCA	Topology Change Acknowledgment
TCN	Topology Change Notification
TTL	Time to Live
ns-3	Network Simulator 3

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th Anniversary Edition. Cambridge University Press, 2010.

[2] M. Żukowski, A. Zeilinger, M. A. Horne, and A. K. Ekert, “"Event-ready-detectors" Bell experiment via entanglement swapping,” *Physical Review Letters*, vol. 71, no. 26, pp. 4287–4290, 1993, doi: 10.1103/PhysRevLett.71.4287.

[3] 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*, vol. 81, no. 26, pp. 5932–5935, 1998, doi: 10.1103/PhysRevLett.81.5932.

[4] K. Chen Hu, K. S. Jensen, and P. Popovski, “Sequential Entanglement-Swapping assisted by Quantum Protocol over Ethernet Networks,” in *International Conference on Quantum Communications, Networking and Computing*, Feb. 2025.

[5] “Basics of Quantum Information.” [Online]. Available: https://learning.quantum.ibm.com/course/basics-of-quantum-information

[6] J.-W. Pan, D. Bouwmeester, H. Weinfurter, and A. Zeilinger, “Experimental Entanglement Swapping: Entangling Photons That Never Interacted,” *Phys. Rev. Lett.*, vol. 80, no. 18, pp. 3891–3894, May 1998, doi: 10.1103/PhysRevLett.80.3891.

[7] “Entanglement Distribution Techniques in Quantum Networks.” [Online]. Available: https://postquantum.com/quantum-networks/entanglement-distribution/?utm_source=chatgpt.com

[8] R. Horodecki, P. Horodecki, M. Horodecki, and K. Horodecki, “Quantum entanglement,” *Reviews of Modern Physics*, vol. 81, no. 2, pp. 865–942, Jun. 2009, doi: 10.1103/revmodphys.81.865.

[9] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*, 5th ed. Upper Saddle River, NJ: Prentice Hall, 2010.

[10] O. Bonaventure, *Computer Networking: Principles, Protocols and Practice*. 2011.

[11] “IEEE Standard for Ethernet,” *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, vol. 0, no. , pp. 1–7025, 2022, doi: 10.1109/IEEESTD.2022.9844436.

[12] “IEEE Standard for Local Area Network MAC (Media Access Control) Bridges,” *ANSI/IEEE Std 802.1D, 1998 Edition*, vol. 0, no. , pp. 1–373, 1998, doi: 10.1109/IEEESTD.1998.95619.

[13] “IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges,” *IEEE Std 802.1D-2004 (Revision of IEEE Std 802.1D-1998)*, vol. 0, no. , pp. 1–281, 2004, doi: 10.1109/IEEESTD.2004.94569.

[14] nsnam, “ns-3.” [Online]. Available: https://www.nsnam.org/

[15] "5. Conceptual Overview \ifmmode— \else —\fi Tutorial." [Online]. Available: https://www.nsnam.org/docs/tutorial/html/conceptual-overview.html

[16] "ns-3: Bridge Network Device." [Online]. Available: https://www.nsnam.org/docs/release/3.19/doxygen/group__bridge.html

[17] International Journal of Emerging Technologies and Innovative Research, "A Review Paper on Networking Topologies," *International Journal of Emerging Technologies and Innovative Research*, vol. 5, no. 9, pp. 324–330, Sep. 2018, [Online]. Available: http://www.jetir.org/papers/JETIRFH06055.pdf

[18] I. Timbercon, "Time Delay of Light in Fiber Calculator." [Online]. Available: https://www.timbercon.com/resources/calculators/time-delay-of-light-in-fiber%20calculator/