INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA INFORMÁTICA

Telepatia sintética interação humano-máquina em aplicações de messaging decentralizadas

LUÍS SAMPAIO DA NÓVOA BARBOSA ARTEIRO

julho de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Decentralization in messaging applications with support for contactless interaction

Luís Arteiro

A dissertation submitted to obtain the degree of Master of Science, Specialisation Area of Systems and Multimedia

Supervisor: Dr. Paula Escudeiro
Co-Supervisor: Dr. Carlos Ferreira

Porto, July 6, 2020

Dedicatory

Everybody wanted to know what I would do if I didn't finish my thesis.

I guess we'll never know.

Abstract

Peer-to-peer communication has increasingly been gaining prevalence in people's daily lives, with its widespread adoption being catalysed by technological advances. Although there have been strides for the inclusion of disabled individuals to ease communication between peers, people who suffer arm/hand impairments have little to no support in regular mainstream applications to efficiently communicate with other individuals. Additionally, as centralized systems have come into scrutiny regarding privacy and security, the development of alternative, decentralized solutions has increased, a movement pioneered by Bitcoin that culminated in the blockchain technology and its variants.

Aiming towards expanding inclusivity in the messaging applications panorama, this project showcases an alternative on contactless human-computer interaction with support for disabled individuals with focus on the decentralized backend counterpart. Users of the application partake in a decentralized network based on a distributed hash table that is designed for secure communication (granted by a custom cryptographic messaging protocol) and exchange of data between peers. Such system is both resilient to tampering attacks and central points of failure (akin to blockchains), as well as having no long-term restrictions regarding scalability prospects, something that is a recurring issue in blockchain-based platforms.

The conducted experiments showcase a level of performance similar to mainstream centralized approaches, outperforming blockchain-based decentralized applications on the delay between sending and receiving messages.

Keywords: synthetic telepathy, inclusion, decentralization, software engineering, cryptography, distributed hash table

Resumo

A comunicação ponto-a-ponto tem cada vez mais ganhado prevalência na vida contemporânea de pessoas, tendo a sua adoção sido catalisada pelos avanços tecnológicos. Embora tenham havido desenvolvimentos relativamente à inclusão de indivíduos com deficiência para facilitar a comunicação entre pessoas, as que sofrem imparidades no braço/mão têm um suporte escasso em aplicações convencionais para comunicar de forma eficiente com outros sujeitos. Adicionalmente, à medida que sistemas centralizados têm atraído ceticismo relativamente à sua privacidade e segurança, o desenvolvimento de soluções descentralizadas e alternativas têm aumentado, um movimento iniciado pela Bitcoin que culminou na tecnologia de *blockchain* e as suas variantes.

Tendo como objectivo expandir a inclusão no panorama de aplicações de *messaging*, este projeto pretende demonstrar uma alternativa na interação humano-computador sem contacto direto físico e com suporte para indivíduos com deficiência, com foco no componente *backend* decentralizado. Utilizadores da aplicação são inseridos num sistema decentralizado baseado numa *hash table* distribuída que foi desenhado para comunicação segura (providenciado por um protocolo de *messaging* criptográfico customizado) e para troca de dados entre utilizadores. Tal sistema é tanto resiliente a ataques de adulteração de dados como também a pontos centrais de falha (presente em blockains), não tendo adicionalmente restrições ao nível de escabilidade a longo-prazo, algo que é um problem recorrente em plataformas baseadas em blockchain.

As avaliações e experiências realizadas neste projeto demonstram um nível de performance semelhante a abordagens centralizadas convencionais, tendo uma melhor prestação que aplicações descentralizadas baseadas em *blockchain* no que toca à diferença no tempo entre enviar e receber mensagens.

Acknowledgement

Acknowledges are long due and the development and implementation of this project fortunately went unscathed mostly due to the efforts of Fábio Lourenço, who I had the pleasure to have worked with during the past two years and throughout the project. Furthermore, I would also like to extend my utmost appreciation to Dr. Paula Escudeiro and Dr. Carlos Ferreira who respectively believed in the project and urged the group to push boundaries and go on step further, having had inclusively provided with resources and hardware throughout. Their guidance are greatly treasured, having gone beyond their duties by helping publish this work on conferences and further perfect my skills as a student and researcher. Additionally, I would like to extend this courtesy to the institution that is ISEP, where I grew up as a person and as a professional. Dr. Miguel Areias also deserves my utmost respect for guiding me towards learning more about the intricacies of decentralization.

It would be my pleasure to recognize Nadia Sluer and Michael Koenka's efforts on allowing me to better pitch, present and share this project to a much wider audience, a task that is certaintly not easy.

On an ending note, I would like to forward my appreciation to all my friends and family. They know who they are.

Contents

List of Figures

List of Tables

List of Algorithms

List of Source Code

List of Acronyms

AEAD	Authenticated Encryption with Associated Data
AHP	Analytic Hierarchy Process
API	Application Programming Interface
B2B	Business-to-Business
BCI	Brain-Computer Interface
BFT	Byzantine Fault-Tolerance
CAS	Content-addressable Storage
CI	Consistency Index
CPRNG	Cryptographically Secure Pseudorandom Number Generator
CR	Consistency Ratio
DAG	Directed Acyclic Graph
dApps	Decentralized Applications
DH	Diffie–Hellman
DHT	Distributed Hash Table
DLT	Distributed Ledger Technology
ECC	Elliptic Curve Cryptography
ECDH	Elliptic-curve Diffie–Hellman
ECDHE	Elliptic-curve Diffie–Hellman Ephemeral
EEG	Electroencephalography
EMG	Electromyography
EnM	Encrypt-and-MAC
EtM	Encrypt-then-MAC
EU	European Union
EVM	Ethereum Virtual Machine
FCT	Fundação para a Ciência e a Tecnologia
FE	Front End
FS	Forward Secrecy
FURPS	Software quality classifying model standing for Functionality, Usability, Reliability, Performance and Supportability
GDPR	General Data Protection Regulation
GILT	Games Interaction and Learning Technologies Research Center

hApp	Holochain Application
HCII	Human-Computer Interaction International Conference
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure-as-a-Service
IBC	Interblockchain Communication Protocol
IHCI	Interfaces and Human-Computer Interaction Conference
INESC-TEC	Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência
IRTF	Internet Research Task Force
ISEP	Instituto Superior de Engenharia do Porto
JSON	JavaScript Object Notation
KDF	Key Derivation Function
MAC	Message Authentication Code
MtE	MAC-then-Encrypt
NCD	New Concept Development
NPD	New Product Development
OOP	Object-oriented Programming
PCG	Permuted Congruential Generator
PD	Parkinson's Disease
PoS	Proof-of-Stake
PoW	Proof-of-Work
RPC	Remote Procedure Call
RSA	Rivest–Shamir–Adleman
SAM	Served Available Market
SDK	Software Development Kit
SSI	Silent Speech Interface
SSR	Silent Speech Recognition
TAM	Total Addressable Market
TCP/IP	Transmission Control Protocol/Internet Protocol
TDD	Test-Driven Development
TDMEI	Tese/Dissertação/Estágio
TLS	Transport Layer Security
TNC	TERENA Networking Conference
TPS	Transactions-per-second

UI	User Interface
UML	Unified Modeling Language
UUID	Universally Unique Identifier
VA	Value Analysis
WASM	WebAssembly

Chapter 1

Introduction

The project that is inherent to this thesis was developed as part of the Tese/Dissertação/Estágio (TDMEI) course. This chapter's main purpose is to provide an overview of the project and what is to be expected throughout this document. This section will start by presenting with a background context of this very report, proceeding with the context and respective problem in which the application is currently being developed. Afterwards, the approach is briefly synthesised, alongside the inherent main goals driving the development of the project and its restrictions. The Introduction chapter ends with the relevant contributions this project might have to the industry as well as the document's general structure.

1.1 Project Background

This report constitutes the cornerstone of the TDMEI curricular unit, as part of the Master's in Informatics Engineering of Graphics Systems and Multimedia in Instituto Superior de Engenharia do Porto (ISEP).

The application is divided into two components: its interface and the dedicated decentralized backend. This report focuses on the latter and was developed by the author of this document, whilst the former is the focus and main theme of Fábio Lourenço's thesis, who is part of the team developing the project.

The first assignment of the course aims to describe the project that is being developed, through a thorough analysis of the surrounding context, state-of-the-art and justify the approach that ultimately was chosen. To this, a value analysis is inherent which further gives more context to the problem at hand and better grasp the advantages of this solution. Additionally, each approach discussed will be compared with each other and a design for the implementation of the project will also be outlined. The second and final assessment encompasses the development and implementation of the project, as well as reaching to conclusions through experimentation. This report pertains the second assignment.

In light of the above, the project is being developed with Games Interaction and Learning Technologies Research Center (GILT), which greatly helped throughout the project by providing access to resources. This work can be summarized as a new approach to human-machine interaction solely by brain activity and silent speech, where the data for both is extracted through an Electroencephalography (EEG) headset and Electromyography (EMG) electrodes, respectively. This interaction occurs in an unorthodox messaging/social application that is decentralized and distributed, attempting a new approach on privacy and communication between peers in a trustless context and with anonymity, security and privacy as crucial cornerstones, roughly replicating blockchain concepts. The following chapters aim to explain the approach further, the problem it is addressing and clarify the surrounding context.

1.2 Context and Problem

The research project at hand ties directly with two state-of-the-art and rising concepts of different fields: contactless interaction with applications; and development of scalable decentralized applications with security and anonymity in mind. In layman's terms, it contains machine learning and decentralized systems powered with technologies that can be compared with blockchain (albeit not quite).

There has been significant progress addressing issues related with support for disabled people to make use of devices, mainly through EEG sensors capable of recording brain activity that, powered by machine learning, can be translated into actions and image reconstruction, as well as other types of outputs (Tatum, Diciaccio, and Yelvington 2016). This can prove to be helpful for people with specific motor disabilities (e.g. arm/hands impairment) to also make use of such applications. Similarly, EMG electrodes can measure activity from muscle responsible for speech, enabling Silent Speech Recognition (SSR), whose output can be used and applied to a myriad of fields, including the messaging and communication ones that are being addressed in this work.

With the advent of mobile messaging applications, there is an apparent lack of support for disabled individuals to communicate on mobile platforms and use such applications to interact efficiently with other people (Jackson 2012). This scarce support effects directly individuals with disabilities that effect the arm/hand motion, a requirement to utilize handheld or computer devices.

In addition to this, security/privacy of data in messaging platforms is a recurrent topic of debate (Hoyle et al. 2017) that has been catalyzed by the recent Cambridge Analytica scandal from Facebook. Such mainstream applications utilize a centralized deployment approach to their platforms, where the business logic and user's messages flow through dedicated servers. Such applications are, therefore, inherently vulnerable to central points of failure and specific attack and data breaches, occurrences that have happened in the past (Cadwalladr and Graham-Harrison 2018). Even though there has been development of decentralized applications based on blockchain, their performance are often lackluster as it scales (Thakkar, Nathan, and Viswanathan 2018).

All in all, the main problem that is being addressed by the current application this report refers is the inclusivity of individuals with specific disabilities and their possibility of communication with high security and privacy levels in an environment that is not susceptible to data breaches and tampering and is non-reliant on a central authority/entity.

1.3 Approach

As it was previously mentioned, this project deals with two rather distinctive fields that are, on their own, highly compatible, these being machine learning and decentralized systems.

The project (code-named as Meletea whilst being developed) deals with EEG - the monitoring of the brain's electrical activity. This is the medium in which the user effectively interacts with the application. Machine learning algorithms will translate these into actions and will, consecutively, make it possible for users to navigate through the application without any need for arm or hand movement. Additionally, considering that people with both arm/hand and voice impairments are the targets and experimental audience for this project, silent speech is employed and results in text input without any movement whatsoever. Accurate SSR and translation into text are possible through machine learning, although the accuracy levels might dwindle from subject to subject and session to session.

We have witnessed a rise in interest of decentralized applications which, by nature, guarantee immutability of data and records on a trustless environment (Nakamoto 2008). Pioneers Bitcoin and Ethereum are prime examples of the potential of blockchain (and its variants) use-cases and application. Current mainstream messaging applications neglect these aspects as these are usually centralized solutions with dedicated servers (Raval 2016). Albeit there are decentralized alternatives, there has yet been developed a decentralized backend tailored for inclusivity and people with the aforementioned disabilities and data input. Having a decentralized backend can also (depending with certain technologies) open up the potential for interoperability with other decentralized systems, thus making it possible for these people to interact on more fronts and expand the feature set of the project further (Kwon and Buchman 2019).

Therefore, the backend (which is the focus of this thesis) counterpart will engage in a decentralized approach in which the system is agent-centric instead of the data-centric approach found in classic blockchain systems. This is mostly because there are a myriad of benefits that blockchain-variant technologies provide, these being namely related with heightened levels of security, privacy of data and tamper-proof capabilities, which can also be found in the classic blockchain concept. The main characteristic that makes this choice enticing is its decentralization that, from the get-go, mitigates any tampering attack and any possible central points of failure, as the data is spread through the network. This reliability on the state of the system is adequate for communication between users who are sure the information circulating within the system is reliable and checked throughout. The security of data is made through recurring validation and based on a cryptographic messaging protocol that is implemented.

Although there are current systems that make use of a distributed hash table on their design (such as the case with Jami), they use it only for mapping out users' address to establish communication and use servers to store undelivered messages, which goes against the idea of full decentralization on all data this project seeks.

The outlined project can be thought not only as a different approach to the messaging panorama but also a new application of distributed ledger technology where privacy is at the forefront of the user's priority. Furthermore, it makes it so people with specific motor disabilities can interact with others, thus breaking down barriers of communication regardless of one's disability status.

1.4 Objectives

The main objective of this project is to develop a messaging application with a Distributed Ledger Technology (DLT)-based network for communication between peers that both supports and is inclusive for people with hand/arm movement impairments. It also aims for these type of individuals to be able to efficiently communicate with others, regardless of disability status. Considering this main objective, some minor major objectives can also be outlined that branch from the former and that directly affect the development of the project and how it intrinsically works and is designed.

- Promote a new approach on contactless human-machine interaction and navigation through no movement whatsoever.
- Showcase a different proposal on the text input paradigm through muscle activity measurement to perform SSR.
- Analyse a new application of distributed systems tailored to disabled people, thus inserting the latter in a trustless and decentralized ecosystem.
- The main problem this project addresses regards the privacy and secure message transfer between peers. It aims to boast a new approach to peer-to-peer messaging that is both resilient to attacks and maintains a high level of privacy whilst doing so, all within an ecosystem where every peer has the same amount of influence over the network and data.

These objectives translate, generally, the components that constitute the overall project. That is, the communication between the user and the application through an EEG headset/EMG electrodes and the decentralized backend, the two being considered the cornerstones of the application.

Notwithstanding, there are some key goals specific to the decentralized system counterpart. There are two main objectives for this, one being having the system that can scale linearly as more users join in the network and has no limit when it comes to long-term scalability; the other refers to the privacy and security on the exchange of data and messaging between the constituting peers of the network.

Delving further in the latter, it aims to make it impossible for data eavesdropping and for it to be compromised, as well as not holding any user's sensitive data. Having the application data and records cryptographically stored (with partial consensus) in a public and decentralized DLT makes it resilient against central points of failure. Furthermore, it is intended to instil a sense of community for those that are within the network and to make the project baseline design ready for possible future features and easily expandable.

Adding to this, this project aims to showcase a new implementation of the so-called Blockchain 3.0, a different approach to the concept of DLT and how these can be scalable on a long-term perspective. This project is merely experimental and fully research-oriented towards decentralization in messaging systems and accuracy from interaction through interfaces like Brain-Computer Interface (BCI) and Silent Speech Interface (SSI).

The security and resilience of the application is aimed to utilize well-known, tested and peer-reviewed cryptographic methods, most preferably resilient to quantum-computers' processing power, when users interact with each other, allowing them to do so securely.

1.5 Hypothesis

This project aims to showcase an application that utilizes a decentralized approach that is as safe as a blockchain-based platform but has higher performance, retaining the tamper-proof and central point of failure resilience that the latter inherently has. This project being roughly research-oriented, the main hypothesis this project aims to prove is:

Hypothesis $\mathcal{H}$ - Decentralized environments and resource sharing based on Distributed Hash Table (DHT) using a partial consensus protocol have better performance, scalability prospects and maintain a bottom line of data integrity akin to blockchains when compared with the latter with global consensus protocols.

This hypothesis stems from comparing three distinct approaches: a centralized messaging application, a classic blockchain-based one and a blockhain-variant platform, more specifically based on DHT. With this, we aim to prove that systems based on blockchain-variants (more specifically utilizing a DHT) can have a better performance/throughput than their pure blockchain counterpart. The hypothesis also relates to centralized solutions, as these are the comparison point in terms of performance in which our application strives for.

The experiments and overall performance of operations and messaging between users with the DHT should provide empirical evidence that the performance is indeed better when compared with a classic blockchain platform adapted to messaging.

1.6 Expected Outcome

Overall, the main objective and expected outcome of the application is to allow the user to navigate through the interface in a contactless manner and have the chance to use subvocalization to input text within it, even though it is prone to low levels of accuracy.

Regardless, taking into account this report focuses on the backend counterpart, the expected outcome of this project is to showcase a high level of performance. Metrics referring to such performance include the delay between the user sending a message/posting data to the network and the recipient receiving the message.

Additionally, it is expected that data within the network is tamper-proof against specific attacks to change pieces of data by malicious users. Communication and data exchange are expected to have a level of privacy and resilience against malicious third-party intents and be protected through cryptographic algorithms and follow peer-reviewed methods, in order to showcase reliability regarding the protection of the data exchange and communication within the system. The performance metric is to undertake experiments comparing different approaches and make use of statistical tests to validate the output samples.

Ultimately, the overall project should showcase a new concept when it comes to messaging, both on an interaction level, support for handicapped individuals and decentralization level, with the latter providing higher indices of performance when compared with blockchain approaches and competing with mainstream common centralized platforms.

1.7 Brief Value Analysis

The value analysis of this project is more deeply explained in Section 3. The value proposition of the project relates directly with the support for disabled individuals for private communication, targeting specific disabilities whose support is extremely scarce, more specifically upper-body limb loss or impairment of conventional hand use. Additionally, the heightened level of personal data security is an added value to the system, as well as its wide range of applicability, not only for the messaging panorama, for example.

Regarding the product development paradigm, the Front End Innovation refers to the first stage of product development and, using the New Product Development mode, there are activities and processes within the project that correlates with these and are included in the development cycle (Peter A Koen et al. 2002).

The opportunity identification part of the project stemmed from the increasing scepticism regarding privacy in message exchange and support for impaired individuals to do so. To further outline the previous aspect, opportunity analysis was made on the market and technology trends. The idea generation and enrichment aspect of the project stemmed from the idea of empowering the user on a more democratic and safe ecosystem, which grew and culminated in the development of the idea of a decentralized system. The idea selection for the approach for the latter was chosen according to criteria based on performance and scalability. Finally, the concept definition winded up from an evolutionary and incremental idea selection processes, that passed through centralized, EEG-based communication, culminating in a blockchain variant-based messaging application with BCI and SSI.

The Canvas Business Model of this application deals primarily with key partners, those being GILT and hospitals for the assessment from subjects of the performance and accuracy of the application. Key resources include everything related with the SSI and BCI, more specifically EEG and EMG sensors, partaking as well within the cost structure of the project. The customer segments refer to the aforementioned messaging app users that want more privacy and comfort of usage, and hand/arm disabled people.

The value network of the application used to analyze the relationship with the project and respective entities comprise mainly of working with public institutes to gather subjects for evaluation and to obtain feedback for study data and improvement on the feature set of the application.

Finally, the Analytic Hierarchy Process was applied in order to choose the best approach for the backend counterpart based on innovation factor, high throughput, the security of data, central points of failure and cost of operation and hosting. The outcome of the process and the corresponding solution was the DHT-based approach.

1.8 Planning

The project was organized in order to fulfill many needs, to be applicable to scientific conferences and to uphold the outlined objectives inherent to the application.

1.8.1 Tasks Planning

The planning for the project is showcased in the diagram in Figure 1.1, which was firstly created for the development of the project and to tend to the objectives that had been set.

The development cycle that refers to the design, implementation and testing of the application as a whole follows a Scrum methodology, with weekly *sprints* between the development members, with sporadic meetings with the supervisors to track the development of the application. The week objectives, tasks and priorities are discussed in these weekly meetings.

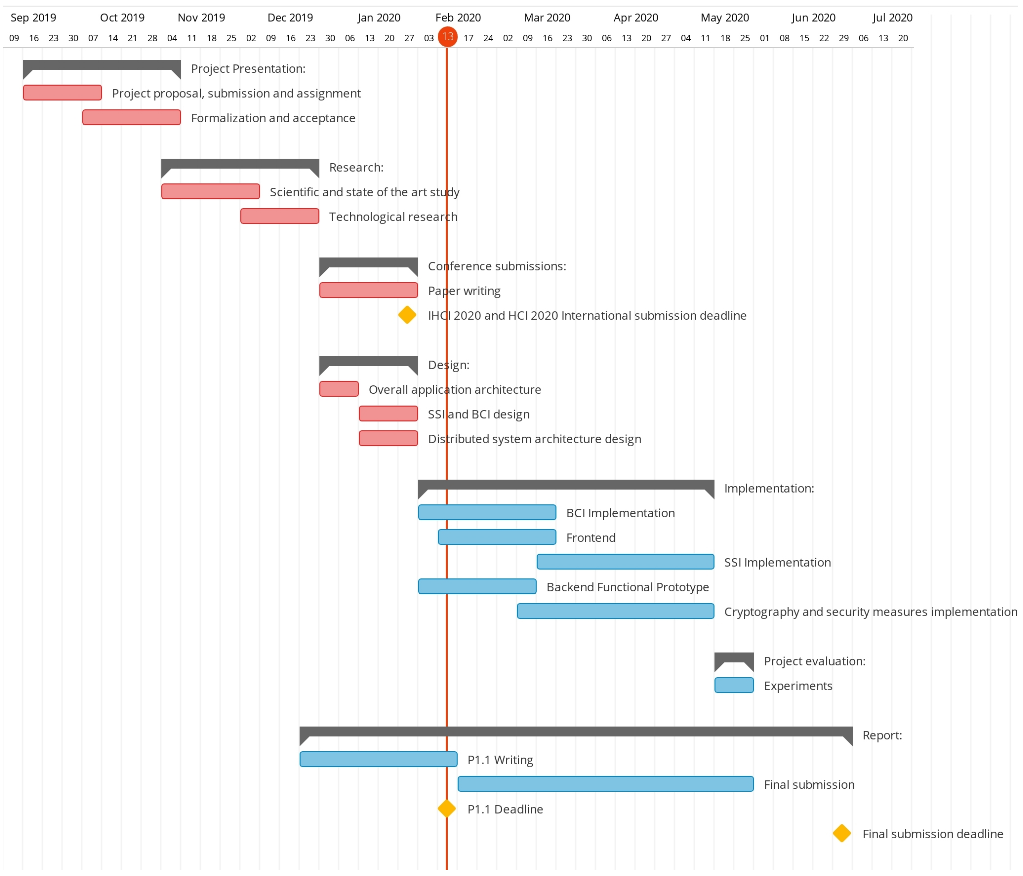


Figure 1.1: Project's Gantt diagram detailing the planning of the development cycle of the project.

In addition to this, the group utilized a Scrum board to withhold the tasks and assign these accordingly, having a back log for future features. To aid this management, Trello was used to assign tasks to assignees and track the progress of each one, doing so on a weekly basis.

As it is possible seeing in the diagram, the project planning encompasses the proposal, submission and acceptance stage, the research, both technological and scientific regarding the technologies and approaches for the project and the conferences work-in-progress paper's writing. The development of the application had an initial design phase for both counterparts, having the remaining months to dedicate exclusively for the development of the project, also including the evaluation and assessment phase with individuals from the target audience. Nonetheless, the report development is made throughout the year and tweaked in accordance to the changes in the project.

It is also outlined in Figure 1.1 that there was an experimental phase that would be dealt with target audience. Although initially planned, this was later not possible because of the current COVID-19 crisis, making it impossible to schedule meetings in hospitals. This information is deeply explained in Chapter 7.

1.8.2 Meetings

As this project looks for non-conventional ways to interact with computers using out-of-the-box approaches, the team felt the need to look for guidance in the different fields that include the main features of our project. Moreover, the team elected a supervisor, Dr. Paula Escudeiro, and a co-supervisor, Dr. Carlos Ferreira.

Throughout the lifetime of this project, a few meetings were arranged to discuss objectives and approaches with the supervisors, as well as with other experts in different fields that can guide ahead in some areas that our supervisors' expertise does not encompass, one of these being Dr. Miguel Areias, from Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC-TEC), that had experience within the area of embedded systems and decentralization. Appendix A displays a brief summary of each meeting, along with the participants and its respective date, pertaining only to the meetings that were held in person.

Besides in-person meetings, there are also common contacts via e-mail to receive feedback about the conference papers the group wrote (refer to Section 7.4), and informal and unplanned face-to-face consultations to answer simpler questions and keeping every party of the project in sync with the progress of the project - these are not displayed in this document because are frequent throughout, augmented by the inability to meet physically because of COVID-19 restrictions.

1.9 Work Contributions

The application, as a whole, not only contributes to the state-of-the-art of two different fields - silent speech recognition/machine learning and decentralized systems based on blockchain - but also provides a new approach and support for communication between disabled people and those who do not suffer impairments, merging the two in a single ecosystem. It effectively can be considered as an attempt to be a breakthrough of messaging communication agnostic of people's impairments.

Regarding the interaction between the user and the machine, the unorthodox way that is inherent to this application showcases the use of machine learning and brain wave activity measurement on a real-life, practical application. Furthermore, related to the non-audible speech (which is a focus of ongoing research) paradigm, this project provides a new approach relating to the rest of related work. Even though studies have been conducted on SSR systems, having a high degree of "correctness" and accuracy has yet to be breached and, therefore, the implementation within this project can outline a new approach that effectively goes beyond these numbers.

When it comes to the decentralized backend counterpart, this project showcases a new way of applying a distributed hash tabled-based DLT concept to a decentralized application. Considering the decentralized backend related to the application, the messaging protocols that are discussed and employed are useful to demonstrate the benefits of decentralization and doing so with privacy and anonymity of the user in mind, all the while being immutable and tamper-proof, concepts that are inherent to the common blockchain applications.

1.10 Document Structure

This document is divided into eight chapters referring to the content and development cycle of this project, proceeded with annexes and bibliographic references. Each chapter is related with one another and the way they are laid out are meant to be logical, chronological and represent a sequence of thought.

The second chapter describes the project's surrounding context, explaining the adjacent theoretical concepts related to the development of the project, its stakeholders and processes, as well as showcasing some limitations the project has. The main problem, its context and hypothesis will also be explained.

Proceeding to the third chapter, a value analysis regarding the application will be made, identifying the parties involved as well as crucial business processes inherent, as well as analysing and categorizing the market this project inserts in.

The fourth chapter deals with the current state of the art of DLT, blockchain, decentralized and centralized systems, as well as their evolution until the current state of the art. The current viable approaches and related technologies will be discussed and then assessed following a set of criteria, culminating in the decision of the approach taken, justifying it accordingly. similar applications that have undertaken a similar route to this project will also be outlined.

Within the fifth chapter, diagrams and the overall architecture of the application, including the conceptual diagram, will be demonstrated. The main goal of this chapter is to showcase the functional and non-functional requirements of the application and the design choices taken to address the former. The whole solution will be briefly demonstrated, focusing afterwards on the decentralized system counterpart. The logical sequence of the application and the interaction between the components that make up the backend will be differentiated. This chapter will be finalized with design alternatives that were considered but were not implemented, being promptly justified.

The sixth chapter will focus on the implementation of what was detailed in the application; how the Holochain framework is used and how the business concepts are structured and implemented to fulfill the use case goals. Moreover, the overall project structure, frameworks and technologies used will be explained, alongside snippets of code to reinforce some features that were implemented.

The following chapter of this document will explain the assessment criteria for the evaluation part of the application that, essentially, grants scientific value to the thesis and the project at hand. The way each criterion will be assessed, coupled with the corresponding statistical hypothesis and assessment evaluation will be outlined. In this chapter the completed functional and non-functional requirements will also be showcased.

Regarding the last chapter of this thesis, conclusions will be drawn regarding the overall project and which goals were not met. Also, a final appreciation for the project will be conducted, as well as future work prospects this project has.

Chapter 2

Problem and Context

Throughout the development of the application, there were a few concepts inherent to the problem that affected the development of the project. In fact, the application was tailored to some of these. These concepts were key to outline the most important features and, therefore, how to address the latter. Some of the domain-specific concepts are key to understanding how suitable the provided solution is for the target audience and how directly it affects the set of features the application provides and is developed. Moreover, the problem and the formulation of the hypothesis this report attempts to corroborate are also outlined. Ultimately, this chapter will focus on the problem the application is attempting to address, as well as the entities that are related to it, alongside some restrictions it has.

2.1 Problem and Hypothesis

Firstly, there should be a swift distinction how the term "centralized" is used. Throughout this report, centralization and decentralization are terms related to the deployment of the respective application and location of their data. For example, a messaging application can be architecturally decentralized (e.g. have a microservices-based architecture, where the business logic is split into different, self-containable services) but still considered centralized because its deployment and databases are located on dedicated servers. A blockchain, on the other hand, is decentralized because it effectively exists and is deployed in every peer within the network and the data is spread throughout all the agents that make up such network, not in a centralized nor targettable location.

Current common messaging applications, such as Facebook Messenger or WhatsApp, utilize a centralized approach on their applications where data goes through dedicated servers when users message each other. One of the issues this raises regards vulnerability to attacks or downtime prospects. Regarding the former, it directly means that third parties have a unique and known target to approach if they are to try and break user's message for malicious intentions. On the other hand, when it comes to the latter, even though there are current platforms that allow to direct traffic to other redundant servers in case of a server failure, the potential for central point of failure still stands. When compared to decentralised approaches based on blockchain, for example, this probability is practically reduced to zero, because there is always redundancy of data from other peers within the network.

However, even though the blockchain technology indeed provides a more democratic environment and reduces the need for a central authority (Witzig and Salomon 2018), there are performance issues that arise as more users join the network (this topic is further explained in

Chapter 4). Applying this to the messaging panorama, blockchain-based approaches usually require a fee for each message to be able to use it.

The focus of this report is on proving that blockchain variants can withhold a majority of its core attributes (i.e. decentralization of data spread across the network; uneditable data records and transparency of data flow within the blockchain) and provide a better performance and throughput, with the same level of security of private data and transparency of public data records and transactions that is granted by blockchains.

This proof-of-concept is applied to a messaging panorama that is tailored to disabled individuals, who make use of the application through a contactless interface, further encompassing a wider range of disabilities' support. Although not the focus of this report, the decentralized platform developed ought to tailor to this unique type of interaction, something that is null in the current messaging landscape and is part of the problem this project is trying to address: lack of support for disabled individuals, especially those who suffer from arm/hand and voice impairments or related that make physical contact with devices effectively impossible, and making it so securely and privately with access to the advantages decentralized systems provide.

Taken the above into account, and considering this project aims to be fairly research-oriented, an hypothesis arises, this one being: blockchain variants (more specifically, stemming from a DHT) can perform better (i.e. better throughput/message sending speed) than their blockchain counterparts, all the while maintaining data redundant, transparent and tamper-proof.

In addition to this, especially in the messaging panorama, its privacy is crucial to avoid data leakage or it being compromised by malicious third-parties. Therefore, not only normal public data within the network ought to be transparent, communications between users has to be confidential and employ peer-reviewed cryptographic algorithms that grants security of communication between peers.

2.2 Domain Concepts

As it was mentioned previously, this chapter will introduce and briefly describe the important concepts that are implicit within the application being developed, in which its proper definition is crucial to better understand the problem and adapt to the audience to, at later stages, obtain a scientifically valid solution. Considering the whole platform blends a contactless interface interaction procedure and a decentralized dedicated backend, domain concepts of both sides will be explained.

2.2.1 Hand/arm disabilities

This project focuses on assisting subjects with upper-body limb disabilities in private online communications. Therefore, this concept must be clarified by enumerating some of the impairments of which we refer to. The following sections will explain and contextualize these and detail the specific disabilities the application panders to.

Parkinson's Disease

Parkinson's Disease (PD) affects 1% of the population over the age of 60, going up to 5% after the age of 85, being the second most impactful neurodegenerative disease after Alzheimer's disease. It is caused by neuronal loss, mainly in the *substantia nigra* brain region, which is the main cause of the motor symptoms associated (Reeve, Simcox, and Turnbull 2014).

The main symptoms related to hand/arm impairments of PD are akinesia and bradykinesia, which translate to motor complications in terms of speed and amplitude, as well as difficulty in hand movements and grip (e.g. handwriting or holding a cup). These disabilities make the use of common solutions for online messaging challenging, as it becomes harder every day for these people to handle their devices properly and accurately.

Amputees

Another condition that affects the use of devices for messaging (and not solely) is the amputation of an upper-body limb. People can, for many reasons, lose a hand or even an arm, which makes the use of these platforms unpractical. However, phone usage for these individuals is possible as mobile touch keyboards nowadays support and ease the handling of the keyboard with only one hand by providing native features prepared for this type of usage (Welch 2016). Nonetheless, some people do not possess any arms/hands, for whomst it is impossible to interact physically with a device for messaging. Hence, there is a shortage of solutions for online communication for these subjects, even more so on a secure environment.

Paralysis

Comparable in terms of difficulty to the previously described conditions, paralysis is an impairment caused by many sources (e.g. spinal cord injury, traumatic brain injury, stroke, complications from surgery, amyotrophic lateral sclerosis, multiple sclerosis) (Armour et al. 2016). Although some of these causes do not provoke permanent paralysis, as some are able to recover after rehab and therapy (Dobkin 2005), this disorder has a huge impact on the everyday life of these people because it limits a considerable set of actions that require motion, either these being on legs or arms. Therefore, like the condition described in the previous section, these subjects suffer from the same problem while trying to communicate online with other peers.

2.2.2 Synthetic Telepathy

Synthetic telepathy can be described as the process of interacting with a computer through thought without the usage of movements or speech. This is usually nominated as "brain-computer interaction" (Millán et al. 2010). In our approach, we go further on this description and also include the interaction through silent speech, as subvocal muscle activity is still detectable while the subject is thinking and reading. Thus, synthetic telepathy can be achieved by measuring both brain activity and speech-related muscle activity while a user is thinking and humming, and translate this data to actions and inputs to a computer. Further in this document, the methods for this data gathering will be analyzed as well as different technologies used, regarding the invasive degree on the user body.

2.2.3 Privacy

The concept of privacy within the messaging panorama, although broad, can be narrowed down to the attempt of not leaking personal user information to third parties or to malicious entities in the communication between peers, without the chance of its data content being exposed. The security and exchange of data do not entirely boil down to this aspect, though. Many messaging applications claim to have implementations of secure protocols for message exchange, even though there are situations where that is not the case whatsoever (Hoyle et al. 2017).

Concerns for the privacy of data have also simultaneously been increasingly put into the spotlight due to recent events regarding social media, more recently with the Facebook and Cambridge Analytics data scandal, where personal data of millions of people's Facebook profiles were retrieved without consent and used for political campaigning purposes (Lapaire 2018).

Tighter regulation on the protection of data has seen a major increase of attention also because of the implementation of the General Data Protection Regulation (GDPR), an European Union (EU) regulation law which aims to provide control to users over their personal data, granting these a more fine-tuned control on what applications/companies can host of their user's data.

These events affected the way applications are designed in accordance to comply with such regulations and to provide a bottom-line of privacy regarding user content. For messaging applications, several aspects concern the user's privacy of data. One of these is data collection for advertising or selling other companies data for other purposes. Retrieving data for more than is required for the functioning of any messaging app is a current concern that deals with security of data in not only messaging applications.

Decentralization is also a catalyst for the privacy in data flow. Regular mainstream applications are centralized and their user's data are stored in servers, which make up a unique and single target for malicious attacks and data breaches. Decentralization addresses such problem by spreading all data distributively, making it impossible for data to be tampered with or compromised as the attacker needs to get a hold of over half of the entire network to control it, something that is not feasible practically.

Moreover, the presence or input of sensitive data (i.e. personal e-mail, full name, telephone number) in messaging applications is a subject of discussion for the privacy of data. The option to use the app anonymously matters because having to provide sensitive data to create a unique ID within the system can be used for malicious users to track information and specific users. This can be, however, remedied by hashing the data, which makes it unreadable to companies. This can be used to protect contact lists.

Additionally, when discussing the level of confidentiality of message exchange, the subject of cryptography is pertinent to be addressed. The spectrum in which messaging applications use end-to-end encryption in communication is wide and not always guaranteed, which is the case of Facebook's Messenger, for example (Jaeger 2014). Additionally, using dedicated frameworks to encrypt network traffic, such as Transport Layer Security (TLS) or noise, provide an important improvement of security on communication over the internet, since all of the data within is encrypted. Moreover, the existence of cryptographic primitives, such as specific key derivation [1], encryption or hashing algorithms, that are considered and peer-reviewed by cryptographers and having been target of research and attack resilience, are crucial if an application is to boast about higher levels of privacy, especially when their cryptographic protocol methods employ Forward Secrecy (FS) [2].

Therefore, the privacy, security and data integrity in peer-to-peer communication deals with cryptographic methods that are employed at surface-level and also at internet-protocol level, as is the case with TLS and with removing means to the user of supplying sensitive data that can eventually be exploited.

2.3 Stakeholders and Processes

The project inherent to this report counted with the participation of stakeholders and actors who have contributed to the development of the final solution. In this chapter, both the participants of the project will be outlined, as well as describing the actors that make use of the application.

2.3.1 Project Stakeholders and Processes

The development team that is charged with implementing the project, is made up of two developers, coupled with the main advisor and co-advisor. The common goal spanning all of the aforementioned elements is to create the application that showcases an alternative solution on user-machine interaction and decentralized systems and make it functional for regular use. The supervisors also provide the necessary equipment for the proper usage and development of the project, as well as contributing by applying the thesis to adequate scientific conferences, in order to attain further technical feedback and scientific validation from both the machine learning, decentralized systems and security fields.

Regarding the development team, their main goal was to draft out, sketch, design and implement the components that make up the application, referring directly to the technical background of the project, while supervisors also support scientific and technical support. Furthermore, individuals that suffer harm or hand impairments, in addition to patients with vocal problems, will be stakeholders of the application. Target audience groups were to be used, namely to provide feedback and evaluation on its accuracy and performance. Even though there was an assessment plan and questionnaire, albeit with an undefined institution, this was not possible due to reasons referred in Section 7.2.1.

[1]A Key Derivation Function (KDF) derives secret keys from a confidential asset (e.g. password) and can be used to stretch or shorten keys to output in a desired format (Kaliski 2000).

[2]Also known as Perfect Forward Secrecy, FS is a feature in key agreement-based protocols that ensures that an attacker does not have access to all messages even if they compromise one session key between the sender and the recipient (Ristic 2013)

2.3.2 System Stakeholders and Processes

Being a messaging platform, the system's stakeholders are simply and solely the end user, that is, the one that is making use of the application. There is no other actor in the system other than them, as the platform is broad and does not have different roles embedded, therefore not having any extra features for other stakeholders. With that said, the user has access to all messaging features which are, indeed, simple, as the base of the platform consists only of message exchange.

Thus, the definition of processes and stakeholders on a system perspective is not complex. Instead, the complexity of this project will focus on the innovation in both interaction interfaces and the technical aspect of privacy management, while maintaining the user experience and its actions and features as simple and with an accessible learning curve.

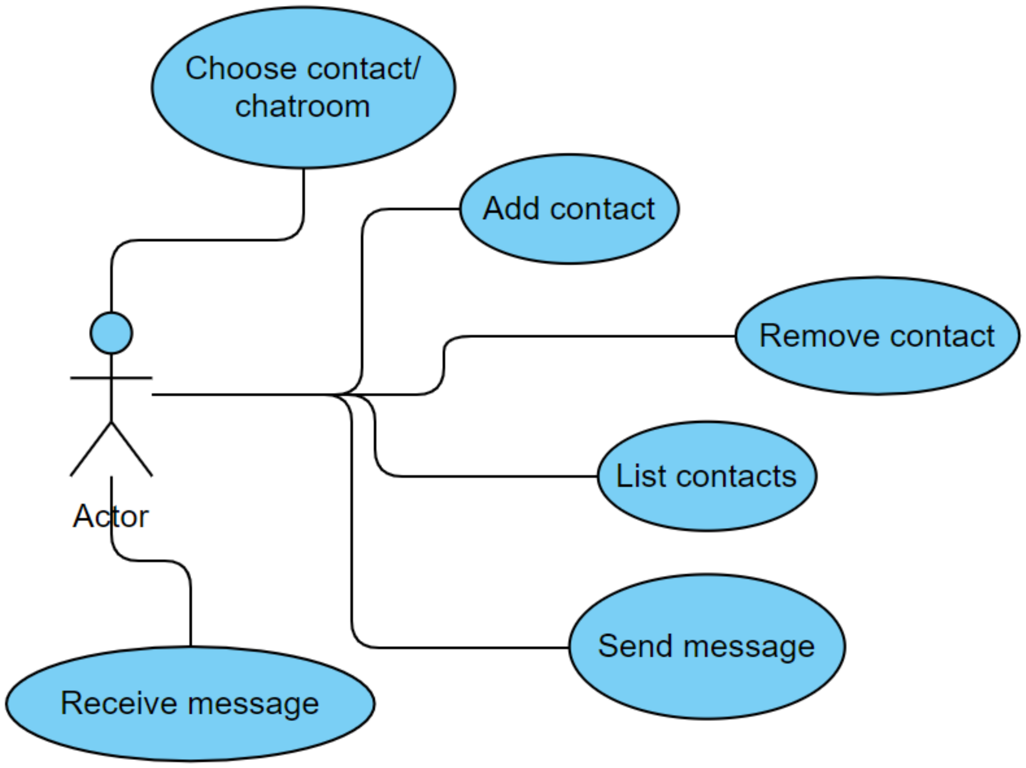


Figure 2.1: Use case diagram of the project.

As with any project, a functional requirements analysis would be important to have a sense of what it is needed to be implemented. But, as it was previously mentioned, the original draft of the application has a low number of use cases directly referring to the user chatting with other peers. Although these can be deepened afterwards, the current basic use cases can be shown in Figure 2.1.

2.3.3 Limitations

There are a handful of restrictions inherent to this project. Regarding the SSI counterpart, accuracy is not guaranteed to be with no errors and with complete accuracy. Being still an area that is still under research, utilizing machine learning methods and algorithms to obtain proper SSR results through electromyography surface electrodes are usually more accurate when they are user and session dependent, not yet being applicable on a commercial scale that can be used by a general audience and still produce good accuracy output. Additionally, a similar narrative can be referred to the BCI component. Although accuracy can be higher, it is never fully accurate across any subject that utilizes it and can fluctuate from person to person.

Chapter 3

Value Analysis

In this chapter, the value analysis of the application will be explained and how it can be commercially viable. Chronologically, this chapter will firstly encompass the definition of what a value analysis of an application is, as well as the value and its perceived value. Furthermore, the market in which this application relates to will be analysed, paving the way to define the value proposition of the application. To finalize, the Front End Innovation process of the application will be explained, proceeded by the Canvas model and value network of the project, ultimately concluding with the Analytic Hierarchy Process (AHP) analysis of the possible approaches taken into consideration while designing this project.

3.1 Concepts Definition

Before delving into this analysis, it is useful to clarify a few aspects revolving around the value analysis concept and their inherent processes.

3.1.1 Value Analysis Definition

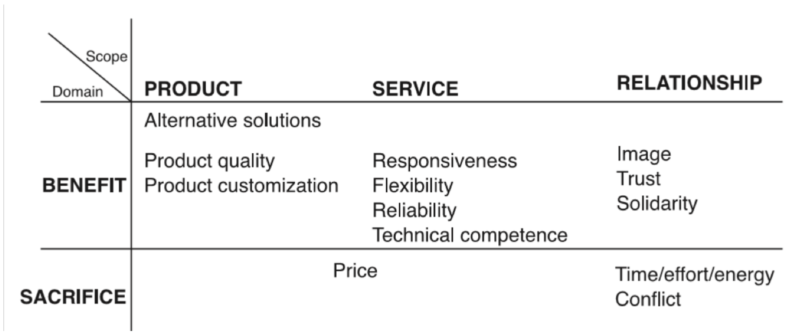
The Value Analysis (VA) of a project/product is created in order to boost its value with the least cost possible, all the while maintaining quality. It allows for the cycle of development to stay on track and consistent throughout and can lead to design choices that, ultimately, affect the product itself. VA is a systematic and formal process of evaluation that comes from management activity, that analyzes the function of a product and how does it serve its purpose. VA is expected to be an incremental process of improvement and the goal is to maximize the following function:

$$Value = \frac{(Performance + Capability)}{Cost} = \frac{Function}{Cost} \tag{3.1}$$

VA defines a basic function (shown in Equation 3.1) for the product which cannot change since it must be the base feature that is fundamental and makes the product sell. In addition to this, there are supporting functions which are features that only support the basic function and help in selling, and can change over time and even be eliminated to reduce the cost of the product, increasing its value without removing product worth. Value is not about minimizing the cost, but can rather about increasing the function more than the cost.

3.1.2 Value, Perceived value and Value Proposition Concepts Definition

When attempting to improve the value of a product, there are three elements that ought to be considered: the use value, which refers to how functional the product is (i.e. performance; capability); esteem value, related to aesthetic value and subjective value (i.e. emotional appeal; style); and market value, referring to how much the market is willing to pay for the product. These are subjective, since each customer perceives the value of a product differently (Ulaga and Eggert 2006). The aspects that come into play for a customer to value a product are shown in Figure 3.1.



Scope / Domain	PRODUCT	SERVICE	RELATIONSHIP
BENEFIT	Alternative solutions Product quality Product customization	Responsiveness Flexibility Reliability Technical competence	Image Trust Solidarity
SACRIFICE	Price		Time/effort/energy Conflict

Figure 3.1: Table defining value-based drives (source: (Lapierre 2000)).

"A product's value proposition is a statement of the functional, emotional and self-expressive benefits delivered by the brand that provide value to the target customer" (Aaker 2002). The value proposition of a product comes out of the necessity in identifying the product, the target segment of the market (to whom it is intended to create value), which specific value the application offers and to explain why the client should purchase the product being presented in lieu of those that are already in the market (Prince Sales et al. 2017). Therefore, this definition can be broken into key components, including what the product offers to the customers, what type of value is associated with such offering and to whom the value is being offered. The value proposition should, therefore, in a business perspective, persuade the potential client by inserting the product within the market and showcase how there is value in acquiring the product and why it does better than the rest of competition.

3.2 Market Analysis

Analysing and identifying target market segments and specific clients and individuals for whom the application has the most potential to deliver value is a crucial step in determining the value proposition (Lanning 1998). Therefore, before stating the value proposition of the application, a market overview will be provided to justify the reasoning behind the application that was developed.

As it was aforementioned, the application being developed inserts within the messaging and communication market. Therefore, its serviceable market comprises of anyone that communicates through digital devices. There are two, however, segments that are especially targeted and which this project aims to target, one being disabled individuals who have arm, hand or vocal impairments and want to communicate with other peers like regular people, and the second being users that value the privacy of their data and the exchange of messages within a secure and private environment. These two segments can overlap.

The most popular global mobile and desktop messenger applications have a staggering number of monthly users which perfectly displays the immense potential of usage and revenue of this market. For example, WhatsApp has 500 million daily active users (Facebook 2019), WeChat attains 113.7 million monthly active users (Tencent 2019) and Facebook's Messenger registers traffic of 1300 million monthly active users (Johnson 2017). The aforementioned comprise of three of the most popular messaging applications (Figure 3.2) that can be found on both mobile and desktop applications.

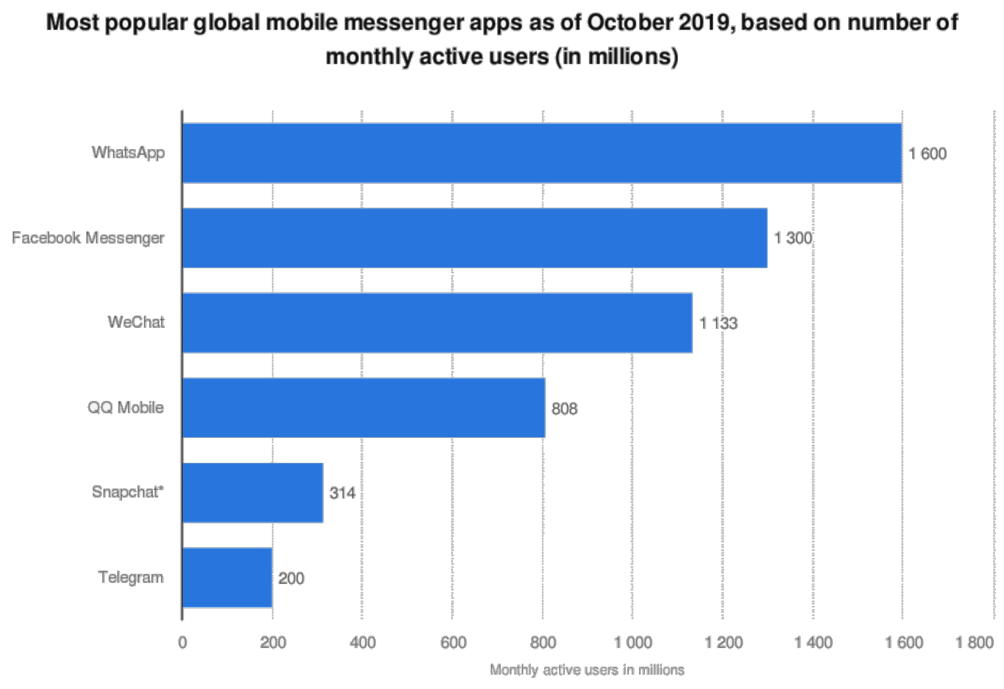


Figure 3.2: Histogram showing the most popular messaging apps with the number of monthly active users, as of October 2009 (source: (Clement 2019)).

All the applications shown in Figure 3.2 are centralized. That is, there are central servers in which data passes through between the sender and the recipient. Signal and Telegram are highly regarded as the most secure mainstream messaging applications, the latter boasting 200 million monthly active users (Durov 2019). All of the aforementioned applications do not have support for disabled individuals, which are one of the target audience, as explained in section 3.2.1. Incidentally, in the United States of America alone, 2.1 million people live with limb loss in 2014 (Ziegler-Graham et al. 2008). This exhibits an opportunity to integrate these individuals and incorporate within a single application that is both inclusive for disabled individuals and can also be used by regular users.

For a business, it is important to define three main segments in the market it aims to address: Total Addressable Market (TAM), Served Available Market (SAM), and Target Market. After the analysis above, it can be stated that the total of monthly messaging apps active users is approximately 6 billion, being that the total market reachable. That is, in other words, our TAM.

The concern in privacy narrows down the project's focus to a SAM that encompasses users who deeply care about these concepts and aspire to take action switching to new solutions where they find these ideas being applied. That might be the example of the users of

Telegram and other similar apps, which subsequently reducing the target audience to 200 million monthly active users.

The Target Market of this application encloses people with arm/hand disabilities, as we provide the solution at hand provides solutions for these individuals to communicate with peers using a fairly new set of technologies. These disabilities stem from a myriad of different sources such as Parkinson's, amputations, or paralysis.

Regarding upper-limb amputees, 30% of amputees are arm-related with 3 million people with these conditions, 90% of which are in developed countries (LeBlanc 2008). Within the United States of America, it is revealed that nearly 1 in 50 people have paralysis from different causes, a total of 50 million (Armour et al. 2016). Considering the same ratio in worldwide internet users, which is estimated to be 4 billion people (International Telecommunication Union 2019), people who are intended to use our platform translate to a total of 80 million people with paralysis.

For this matter, individuals affected by Parkinson's we will not be counted, considering this illness mostly appears after the age of 60 (Reeve, Simcox, and Turnbull 2014), and the number of internet users in this age is not relevant for these values. The aforementioned niche market has little to no competitors for the type of platform and service provided and after the previous analysis, it can be concluded that the Target Market for the current application revolves between 80 and 85 million people, which are directly designated as the target audience.

3.2.1 Value Proposition of the Application

The main value that stems from this application entails support for disabled individuals and heightened levels of privacy of data. In a longitudinal perspective of the value that is being offered, it is important to state that the value of this product will always be present since it is inherent to its implementation and design choices. The proposed solution encompasses both SSI and BCI for the interaction counterpart of the application. This is one of the relevant aspects which differentiates the solution with the rest. It essentially provides value by including and giving support for disabled individuals, more specifically people with arm/hand and speech impairments, and presenting these with a way to efficiently communicate with other peers, regardless of disability status.

On another hand, every user partakes in a decentralized ecosystem that is user-centric that aims to have a high throughput of message exchange and maintaining a high bottom-line of privacy of data. This not only showcases a new approach to decentralized messaging applications but also demonstrates the maintenance of a high level of security through cryptography that is peer-reviewed. Therefore, users may find value in secure communication between peers, something that is above the level of the mainstream solutions that are presented currently.

Another aspect that further boosts the value of this application is its wide range of applicability. In other words, this messaging application can be used in different scenarios (e.g., education). By including support for these type of disabled individuals, many barriers are broken down and allow for a more efficient peer-to-peer contact.

On a business perspective, the users could have access to the application for free, albeit some premium features would need to be purchased for the user to have access to these. This would be the only expense from the user, which would have benefits even if the free version was used. On a long-term perspective, the product would not lose value for its clients, since the former will be maintained to incorporate new aspects regarding security and peer-reviewed cryptographic methods. The same occurs on the SSI and BCI counterpart, in which we would strive for better accuracy in the interaction between the user and the interface.

Inserted in this thesis's scope, this project's aim (and value) is to provide a new insight on decentralized application with support for disabled individuals, therefore pushing further the state-of-the-art when it comes to accuracy in SSI and BCI and privacy of data in messaging platforms that are decentralized and based on distributed ledger technologies that are viable on a production level.

3.3 Front End Innovation

On the product development paradigm, the Front End (FE) is considered as the first stage of new product development, which concerns the period from the idea generation to its approval for the development, or its termination for specific reasons. Essentially it is the starting point where the main opportunities are outlined and concepts are developed before entering the formal product development process (Peter A. Koen, Bertels, and Kleinschmidt 2014), partaking the first part in Figure 3.3, where the New Product Development (NPD) is where the products are indeed developed, proceeding with its launching and commercialization activities.

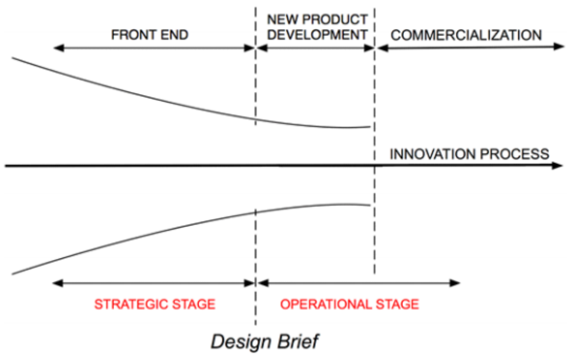


Figure 3.3: The overall product innovation process (source (P. Koen et al. 2001)).

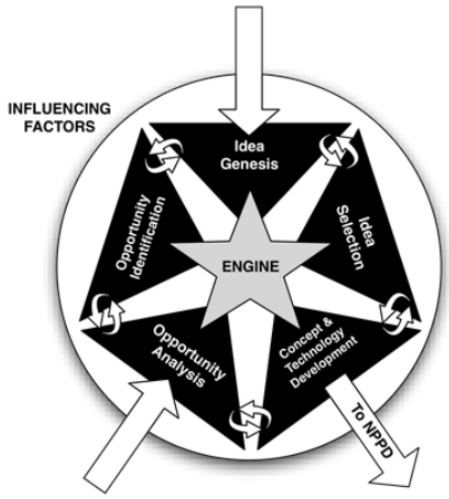


Figure 3.4: Diagram showcasing the New Concept Development model, according to Koen, reproduced by the latter (source (P. Koen et al. 2001)).

The New Concept Development (NCD) diagram showcased in Figure 3.4 provides an overview of the main FE activities that transpire prior to the product development stage and commercialization as is split into three counterpars: the surrounding influencing factors, the engines that drives the activities of the FE and five activity elements, of which this document will frame concerning the application being developed.

3.3.1 Opportunity Identification

The opportunity identification part relates to the technological changes and opportunities that are identified and are meant to be pursued (Peter A Koen et al. 2002). Concerning the project, the opportunity that was identified was the apparent lack of support for individuals with arm/hand and vocal impairments to communicate with other people on a common messaging platform. To add to this, there are not available solutions that incorporate EEG and subvocalization on a decentralized ecosystem with the privacy of communication in mind, with the potential of growing into more than a conventional peer-to-peer or chatroom messaging platform and be applied to a wider range of scenarios.

Moreover, the presented application showcases a new product that encompasses two growing fields: machine learning and blockchain and its variants. Decentralization of data allows for a higher level of security in communication between peers and a new approach to messaging.

Market research and technology trend analysis are tools that can aid and reinforce this concept. Referring to this project, the increase of scepticism of privacy on messaging applications creates a space in which this application can strive (Schrittwieser et al. 2012). Furthermore, it was recorded that approximately 2.52 people worldwide made use of messaging applications at least once time per month (Enberg 2019). The messaging market is extensive and panders to the human's most natural trait, which is communication. With this in mind and with growing concerns regarding the security of data in communication, messaging platforms like Telegram have been thriving mainly due to the end-to-end encryption features it provides. However, these methods have not been peer-reviewed in an open-source environment and have been found to be vulnerable to certain attacks (Lee et al. 2017). This poses as an opportunity to bring a peer-reviewed, decentralized platform that utilizes encryption methods that are secure to at least quantum levels, all the while providing support for disabled individuals at arm and vocal level, something that can not be found currently.

3.3.2 Opportunity Analysis

Opportunity analysis pertains to if the idea/opportunity being assessed is worth pursuing. This encompasses market and technological assessments. For the latter, refers to the level of maturity of the current technologies that can aid development and if it can be reliable on a production level (Peter A Koen et al. 2002). Framing this activity within the project being developed, this application exhibits an opportunity to showcase a new prototype that concerns subvocalization and decentralization of network for communication purposes. The value inherent is that it is inclusive to arm and vocal impairment, thus allowing people with this kind of disabilities to communicate with virtually anyone on a platform that is similar to others within the market, doing so with security and communication that is not centralized and compromisable.

The methods to validate this activity are similar to those used in Opportunity Identification (section 3.3.1), which can include market research and technology trend analysis. The market research conducted in Section 3.2 is an example of this, which explicitly shows the unexplored niche market that this project addresses.

3.3.3 Idea Generation and Enrichment

The idea generation and enrichment activity concerns the genesis, development and maturation of a specific idea and is meant to be evolutionary (Peter A Koen et al. 2002). About the project, the idea generation stemmed from the will to encompass two different fields that were booming and gaining traction, those being machine learning and blockchain/distributed ledger technologies. Besides, a distinctive lack of BCI solutions coupled with SSI within the messaging paradigm led to the genesis of this project. The current state-of-the-art of SSR does not showcase high accuracy, therefore providing an opportunity to further push the limits of the current state of the field.

When it comes to decentralized systems, the notion of using a decentralized network came with the idea of empowering the user on a more democratic ecosystem. Additionally, the advantages of using decentralized systems over centralized ones on communication platforms are crucial when it comes to privacy and integrity of data that is being exchanged.

As a method to verify this activity within this project, a former role was assigned for supervisors and co-supervisors to coordinate ideas from the generation of the project, throughout its development and assess it iteratively to maintain the development cycle on track.

3.3.4 Idea Selection

Frequently, the main conundrum is not creating new ideas and innovations but rather choosing the best one that would achieve a higher value. This process is not straightforward but rather iterative that should culminate in a final outlining of the product idea (Peter A Koen et al. 2002). Concerning the project presented in this report, the selection of the idea stemmed from an iterative process of what would be liable, market-wise and value-wise. The main focus of the application is the push the state-of-the-art on both synthetic telepathy and decentralized systems on messaging platforms with private and secure data exchange. The conjunction of these target markets where the support for disabled individuals are scarce, all the while addressing common messaging security issues.

In all, techniques that aid verifying the Idea Selection encompass the success probability on technical and commercial, leveraging with a selection process overseen by supervisors which prompt feedback on different methodologies throughout. The latter is an iterative process (displayed in Figure 3.5) and has been present in choosing which technologies would be adequate in addressing the problem the application is addressing. An example of this was straying away from a pure blockchain solution as it was initially thought out, opting for variants that allow for high throughput within the network.

3.3.5 Concept Definition

The Concept Definition pertains the final element and stage of the NCD. As shown in Figure 3.4, it is the only exit to the NPD. To reach the latter, it must make a compelling case for investment in the proposition. Methods to validate this activity include setting criteria that describes how the application fairs in term of the market, how it addresses it and how it adds values in comparison to others (its objectives).

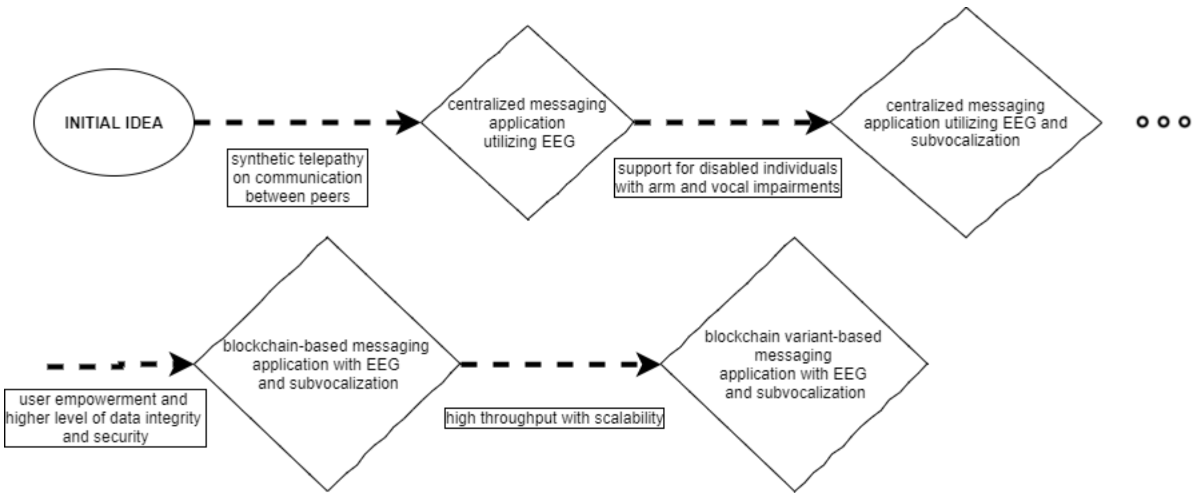

INITIAL IDEA
synthetic telepathy on communication between peers
centralized messaging application utilizing EEG
support for disabled individuals with arm and vocal impairments
centralized messaging application utilizing EEG and subvocalization
user empowerment and higher level of data integrity and security
blockchain-based messaging application with EEG and subvocalization
high throughput with scalability
blockchain variant-based messaging application with EEG and subvocalization

Figure 3.5: Idea selection process for the application at hand, in which culminates in the last stage that will eventually lead to NPD.

For this project, the main objectives, as it was aforementioned, are to provide a new solution to the messaging communication panorama that offers state-of-the-art privacy and security, all the while making it inclusive for individuals with arm and vocal impairments and not only. The serviceable market for this application is widely broad, since it can be used by both regular or disabled individuals. However, it is tailored to the latter and are targeted for individuals that value the privacy of their communication and data when messaging other peers.

3.4 Canvas Business Model

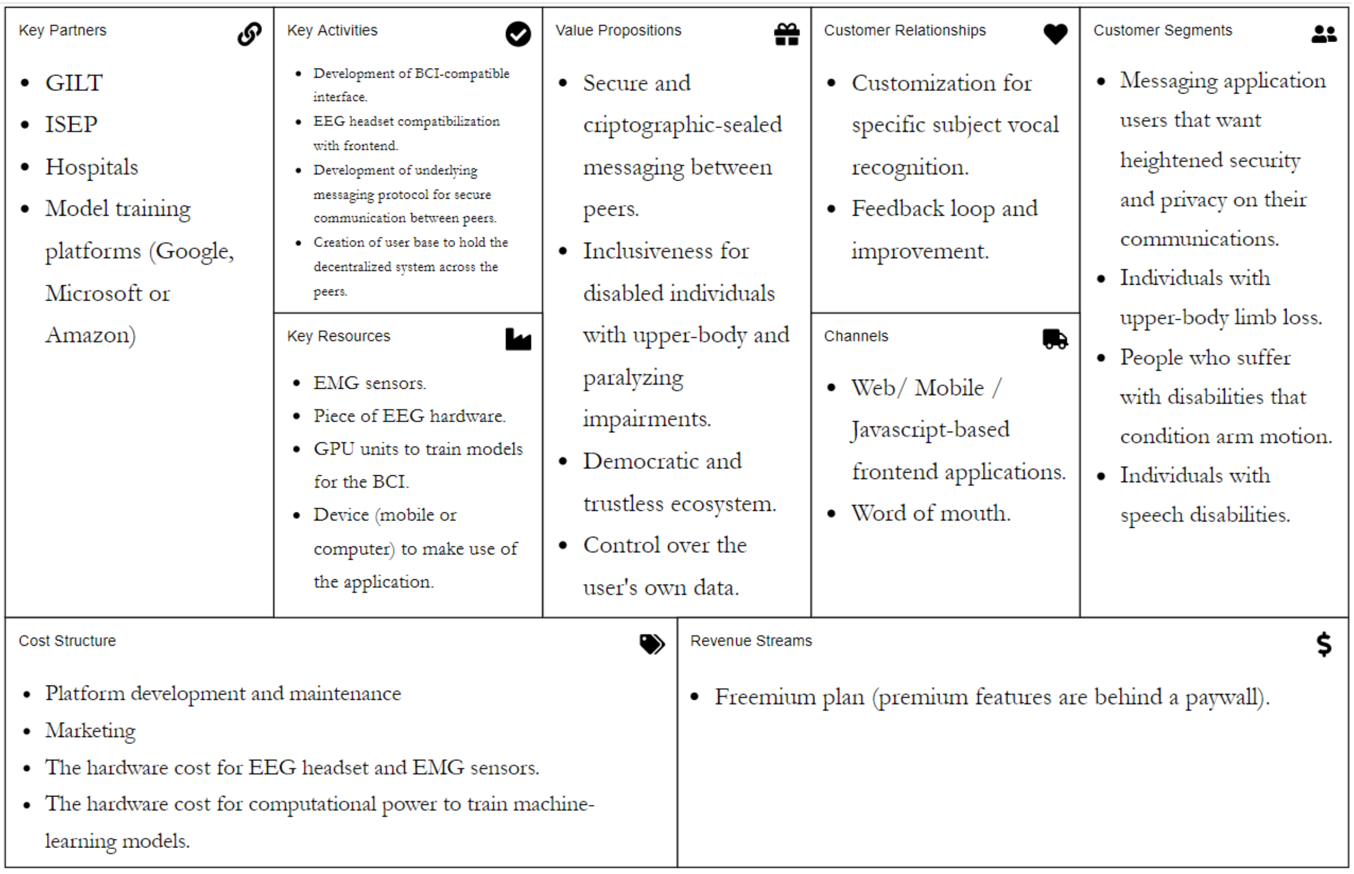
A business model describes the rationale of how a company generates value by analyzing a problem, creating a solution, and using the correct means to supply its customers. A business model is the logical process to define who the customer is, what the problem is, how to solve it and how to benefit from it, so a product or service is not a business model by itself (Osterwalder and Pigneur 2013).

The Canvas model is a simple and visual model for business definition which briefly introduces the key aspects of a company. It analyses nine viewpoints on which the company and its activities are based on, which are the following:

- **Key partners** - Who are the key partners/suppliers? Which resources do we acquire from them?
- **Key activities** - Which activities are fundamental for our value proposition?
- **Key resources** - What resources do our value proposition require?
- **Value propositions** - What do we offer to the customer?
- **Customer relationships** - What relationships do we establish with our customers?
- **Customer segments** - For whom are we creating a product?
- **Channels** - How do we reach customers?

- **Cost structure** - What are the most important costs for creating our product?
- **Revenue streams** - For what are our customers paying? How do we profit?

The Canvas model attempts to answer these questions above in a simple format. For our project, the Canvas model is showcased in Figure 3.6.



Key Partners

- GILT
- ISEP
- Hospitals
- Model training platforms (Google, Microsoft or Amazon)

Key Activities

- Development of BCI-compatible interface.
- EEG headset compatibilization with frontend.
- Development of underlying messaging protocol for secure communication between peers.
- Creation of user base to hold the decentralized system across the peers.

Key Resources

- EMG sensors.
- Piece of EEG hardware.
- GPU units to train models for the BCI.
- Device (mobile or computer) to make use of the application.

Value Propositions

- Secure and criptographic-sealed messaging between peers.
- Inclusiveness for disabled individuals with upper-body and paralyzing impairments.
- Democratic and trustless ecosystem.
- Control over the user's own data.

Customer Relationships

- Customization for specific subject vocal recognition.
- Feedback loop and improvement.

Channels

- Web/ Mobile / Javascript-based frontend applications.
- Word of mouth.

Customer Segments

- Messaging application users that want heightened security and privacy on their communications.
- Individuals with upper-body limb loss.
- People who suffer with disabilities that condition arm motion.
- Individuals with speech disabilities.

Cost Structure

- Platform development and maintenance
- Marketing
- The hardware cost for EEG headset and EMG sensors.
- The hardware cost for computational power to train machine-learning models.

Revenue Streams

- Freemium plan (premium features are behind a paywall).

Figure 3.6: The Canvas Model for the project, showcasing each aspect that were explained prior.

As shown in the Canvas Model, the key activity is indeed the development of the messaging platform and the compatibility with the frontend interface composed with EEG and EMG sensors. This presupposes the integration between the sensors and our platform.

The target audience of the platform are people who desire to message securely and privately, although keeping in mind people with some specific disabilities and pursue their inclusion within the community. The way the majority audience is reached will be exclusively online. However, during the development phase, some tests might be arranged in person, as it is planned to establish contact with hospitals and related entities to aid testing some features that aim to help impaired subjects.

Key resources include the hardware that makes it possible for the planned unorthodox interface between the user and the computer. Some of the costs include these and the extra hardware to train the machine-learning models that make it possible to translate thoughts into tangible navigation actions. The main revenue stream includes premium features that are hidden behind a paywall.

3.5 Value Network

Proposed by Verna Allee, the value network is a new model to analyze a company based on its relationship network with other entities. Although knowledge and intangible value exchanges are the base of most emerging companies, most business models do not consider these. This model takes into account two key types of value exchange, those being tangible and intangible. The first is related to revenue and goods that two entities transact with each other. The latter refers to benefits and knowledge trade, which is not quite considered in other models, although it is a key element in some businesses like ours. In a value network model, the focus is on the stakeholders of a project and their exchange of tangible and intangible benefits, usually all providing and receiving value, this way contributing to the success of each and the success of the network (Verna 2008).

As this model appears more promising to fit the needs of the project and its field of work, when compared to a value chain, the value network referring to this current project is shown in Figure 3.7.

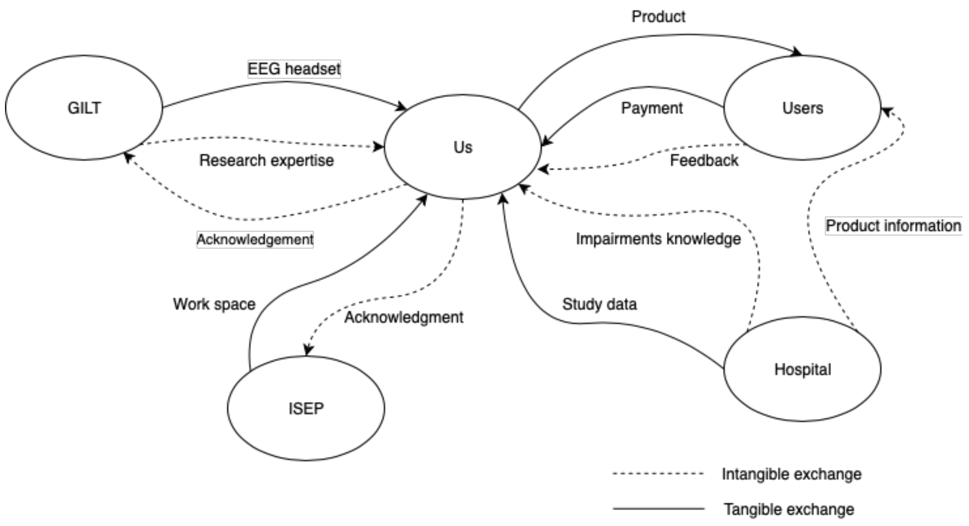


Figure 3.7: Proposed value network for the project following Verna's model.

As Figure 3.7 depicts, since we are working with public institutes that are essentially helping us with intangible goods, the value exchange occurs mainly towards us, as we do not have much to offer besides the research and build of the product to the costumers, and recognition to those who support our project. These exchanges are a key factor for the research and development, as otherwise the development group would not have access to an EEG device that GILT offers us to work with, nor direct contact with impaired people who we are offering value, that can be established through the hospitals.

3.6 Analytic Hierarchy Process

As it was previously showcased, when outlining and creating this project, value is automatically generated. This value must be, however, quantified in order to have a better understanding on its advantages and drawbacks. To quantify and analyse the creation of value, analytical methods have been created, including the AHP.

AHP is essentially a decision making model with the goal of aiding entities to make a decision about a specific scenario based on pre-set criteria. It encompasses a three-part process which includes identifying decision objectives, criteria, constraints and alternatives in to the hierarchy; analyse comparisons between the elements of each level of the hierarchy; and synthesising by using the solution's algorithm of the results of the previous comparisons (Saaty 1988).

The AHP method follows a sequence of events in criteria of outlined based on a problem statement. Afterwards, the decision making process occurs, by comparing each criteria pairwise. Following this, the alternative is therefore chosen based on the input and analysing each alternative. The process applied for this project is what follows, where in each step, the process is explained and promptly applied to the problem at hand.

3.6.1 Decision Scenario

Within the context of the project, it being a research one and attempting to prove the benefits of decentralized systems, it is important, however, to take into consideration other approaches, even centralized, since the latter are one of the reasons this project exists. For this case, the problem statement is 'What backend design structure should we choose?'. The answer to this question exists within the context of explaining why centralized methodologies are not considered under this project's set of criteria and main objective. Even though it might seem trivial to answer this question for this project, having this statement as the basis to our AHP process helps reinforce why decentralization was chosen.

This problem statement regards the foundation of the application, how it works and deals with the user data that is exchanged between these. The criteria chosen for this problem statement are outlined in Figure 3.8, which encompass innovation of the solution (for research and state-of-the-art purposes), high throughput of messaging between peers, data security in message exchange, inherent resilience to attacks and, on a production level, cost of operation and hosting of the platform.

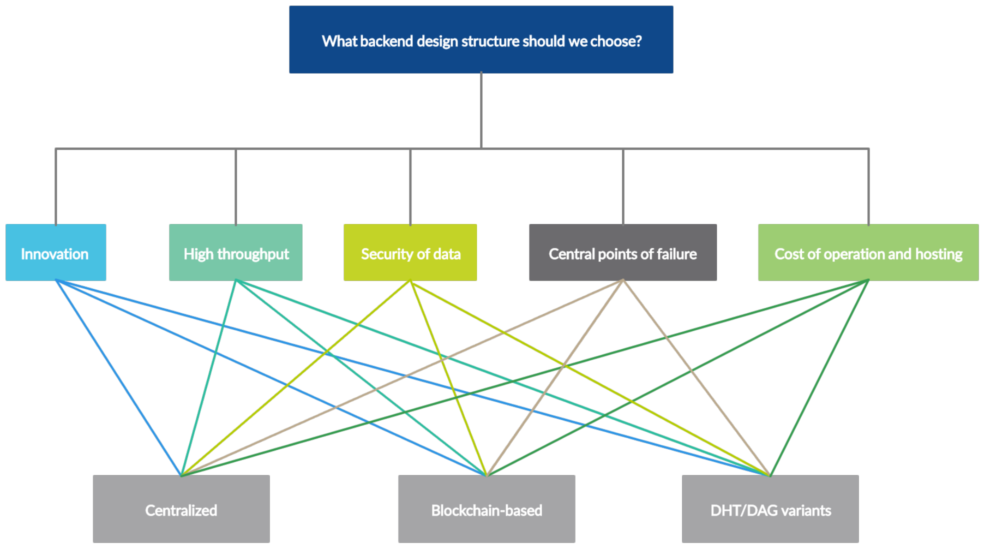

What backend design structure should we choose?
Innovation
High throughput
Security of data
Central points of failure
Cost of operation and hosting
Centralized
Blockchain-based
DHT/DAG variants

Figure 3.8: Hierarchy tree decision diagram showcasing each criteria for the decision of the problem, as well as the possible alternatives.

3.6.2 Pairwise Comparisons

Following what is shown in Figure 3.8, pairwise comparisons between the criteria are needed. For this, it is created a matrix that encompasses every criteria and promptly compares each one with the level of importance over the other. The number is not arbitrary. It ranges from 1-9, from equal importance to absolute importance over the other criteria, respectfully. This is based from Saaty (2008). The matrix for is shown in Table 3.1.

Table 3.1: Pairwise comparison matrix with each criteria being compared to each other based on importance scale. The sum of each column is also displayed.

	Innovation	**High throughput**	**Security of data**	**Central points of failure**	**Cost of operation and hosting**
Innovation	1	3	4	6	9
High throughput	$\frac{1}{3}$	1	1	3	5
Security of data	$\frac{1}{4}$	1	1	1	5
Central points of failure	$\frac{1}{6}$	$\frac{1}{3}$	1	1	3
Cost of operation and hosting	$\frac{1}{9}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{3}$	1
SUM	1.86	5.53	7.2	11.3	23

To normalize each criteria to the same unit, it is needed to divide every element of each column by its sum. With this in mind, the normalized matrix is shown in Table 3.2.

Table 3.2: Normalized pairwise comparison matrix.

	Innovation	**High throughput**	**Security of data**	**Central points of failure**	**Cost of operation and hosting**
Innovation	0.54	0.54	0.56	0.53	0.39
High throughput	0.18	0.18	0.14	0.27	0.22
Security of data	0.13	0.18	0.14	0.09	0.22
Central points of failure	0.09	0.06	0.14	0.09	0.13
Cost of operation and hosting	0.06	0.03	0.03	0.03	0.04

To calculate the criteria importance vector - which states the importance of each criteria - from Table 3.2, the arithmetic mean is calculated, row-wise. With this, the following vector is achieved, with the same order of each criteria as shown in previous tables and figures.

$$P = \begin{pmatrix} 0.512 \\ 0.198 \\ 0.152 \\ 0.101 \\ 0.037 \end{pmatrix} \tag{3.2}$$

The vector 3.2 showcases the level of importance of each criteria (i.e., for example, the Innovation factor has importance of 0.512).

3.6.3 Relative Priority Consistency Analysis

The next step of this process is to calculate the Consistency Ratio (CR), which measures the consistency and the veracity of the obtain results. In case CR is bigger than 0.1, it means it is safe to infer that the random evaluations made are not reliable. For this, it is necessary to multiply the priority vector (which we referred as "criteria importance vector" prior - P) with the criteria matrix (as shown in Figure 3.1 - M).

$$\begin{pmatrix} 1 & 3 & 4 & 6 & 9 \\ \frac{1}{3} & 1 & 1 & 3 & 5 \\ \frac{1}{4} & 1 & 1 & 1 & 5 \\ \frac{1}{6} & \frac{1}{3} & 1 & 1 & 3 \\ \frac{1}{9} & \frac{1}{5} & \frac{1}{5} & \frac{1}{3} & 1 \end{pmatrix} \times \begin{pmatrix} 0.512 \\ 0.198 \\ 0.152 \\ 0.101 \\ 0.037 \end{pmatrix} = \begin{pmatrix} 2.653 \\ 1.008 \\ 0.764 \\ 0.515 \\ 0.198 \end{pmatrix} \tag{3.3}$$

To calculate λmax, it is needed to calculate the mean of the resulting values after the division of the resulting vector and the priority vector (the second element in Equation 3.3), like so:

$$\lambda max = \mu \left(\frac{2.653}{0.512}, \frac{1.008}{0.198}, \frac{0.764}{0.152}, \frac{0.515}{0.101}, \frac{0.198}{0.037} \right) = 5.148 \tag{3.4}$$

Having found the value of λmax, it is possible to calculate the Consistency Index (CI) by the following equation:

$$\frac{y_{max} - n}{n - 1} = \frac{5.148 - 5}{5 - 1} = 0.037 \tag{3.5}$$

In Equation 3.5, *n* relates to the order of the matrix of criteria (the first element in Equation 3.3).

The last step to calculate CR utilizes Saaty (2008)'s table of indices, in which it is divided the CI with the respective value of the index in the aforementioned table of indices, thus resembling the following equation:

$$CR = \frac{CI}{index} = \frac{0.037}{1.12} = 0.033 \tag{3.6}$$

Considering that the CR is 0.033, which is lower than 0.1, this means that the weights attributed to each criteria are correctly set.

3.6.4 Alternative's Priority Matrix for each Criteria

For this phase, it is necessary to build a matrix for each alternative (solution) based on each criteria and how these fair in comparison with others. What follows are the tables for each criteria, with the priority vector already calculated (the same procedure found in Section 3.6.2).

Table 3.3: Comparison matrix for the Innovation criteria.

INNOVATION	**Centralized**	**Blockchain-based**	**DHT/DAG variants**	Priority Vector
Centralized	1	$\frac{1}{6}$	$\frac{1}{6}$	0.076
Blockchain-based	6	1	1	0.462
DHT/DAG variants	6	1	1	0.462
SUM	13	2.17	2.17	

Table 3.4: Comparison matrix for the High Throughput criteria.

HIGH THROUGH-PUT	**Centralized**	**Blockchain-based**	**DHT/DAG variants**	Priority Vector
Centralized	1	7	1	0.467
Blockchain-based	$\frac{1}{7}$	1	$\frac{1}{7}$	0.066
DHT/DAG variants	1	7	1	0.467
SUM	2.14	15	2.14	

Table 3.5: Comparison matrix for the Data Security criteria.

DATA SECURITY	**Centralized**	**Blockchain-based**	**DHT/DAG variants**	Priority Vector
Centralized	1	1	1	0.333
Blockchain-based	1	1	1	0.333
DHT/DAG variants	1	1	1	0.333
SUM	3	3	3	

Table 3.6: Comparison matrix for the Central Points of Failure resilience criteria.

CENTRAL POINTS OF FAILURE	**Centralized**	**Blockchain-based**	**DHT/DAG variants**	Priority Vector
Centralized	1	$\frac{1}{8}$	$\frac{1}{8}$	0.058
Blockchain-based	8	1	1	0.471
DHT/DAG variants	8	1	1	0.471
SUM	17	2.125	2.125	

Table 3.7: Comparison matrix for the Cost of Operation and Hosting criteria.

COST OF OPERATION AND HOSTING	**Centralized**	**Blockchain-based**	**DHT/DAG variants**	Priority Vector
Centralized	1	$\frac{1}{9}$	$\frac{1}{9}$	<u>0.052</u>
Blockchain-based	9	1	1	<u>0.474</u>
DHT/DAG variants	9	1	1	<u>0.474</u>
SUM	19	2.111	2.111	

Considering all the Tables 3.4, 3.3, 3.5, 3.6 and 3.7, the diagram in Figure 3.9 showcases the priorities in both criteria level and alternatives/solution level.

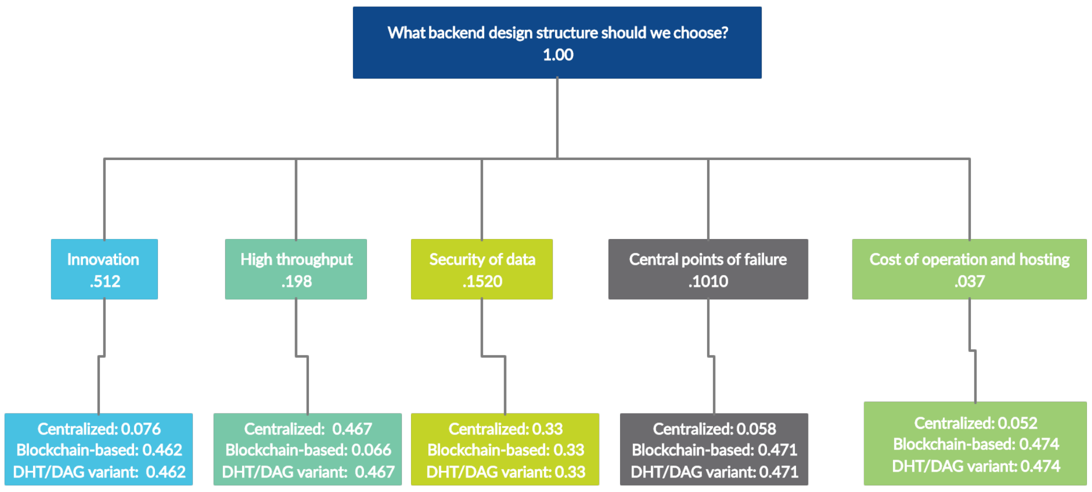


Figure 3.9: Hierarchy tree decision diagram showcasing the priorities in each criteria and how each alternative is best fit for each criteria.

3.6.5 Alternative Priority Compound Result

Combining all the priority vectors of each alternative within each criteria in a single matrix and multiplying it by the priority vector that states the weight of each criteria in Equation 3.2 will provide the priority compound result for each alternative and, therefore, showcasing the best solution for the problem.

$$\begin{bmatrix} 0.076 & 0.467 & 0.333 & 0.058 & 0.052 \\ 0.462 & 0.066 & 0.333 & 0.471 & 0.474 \\ 0.462 & 0.467 & 0.333 & 0.471 & 0.474 \end{bmatrix} \times \begin{bmatrix} 0.512 \\ 0.198 \\ 0.152 \\ 0.101 \\ 0.037 \end{bmatrix} = \begin{bmatrix} 0.19 \\ 0.37 \\ 0.44 \end{bmatrix} \tag{3.7}$$

where the first element is a matrix with every priority vector of each criteria combined (in order).

From Equation 3.7, it is safe to assume that the third alternative - DHT/DAG variant - is the best solution for the problem at hand.

Chapter 4

State of the Art and Approach Evaluation

This chapter provides an explanation pertaining to the state of the art of approaches within the messaging panorama and their related technologies. Alongside these topics, some current solutions related with each possible approach will also be made explicit. Finalizing this chapter, each approach will be evaluated based on promptly defined criteria. This will ultimately justify the approach for the problem that was explained in Chapter 2.

4.1 State of the Art

The phase of research of the current state of the art are important to localize the current project and also compare how the presented solution fares with the rest of the solutions and approaches available. This study facilitated grasping key concepts and paved the way to further showcase a new approach on the current messaging state of the art and develop the new approach presented in this report.

As it was aforementioned, this section will provide an insightful view on the current landscape of messaging applications and related possible approaches. A few examples of competing solutions will be shown in the following section, showcasing examples from three approaches considered: centralized, classic blockchain-based and blockchain-variants.

4.1.1 Current Related Solutions

The messaging paradigm and industry have been increasingly becoming more prevalent in user's daily lives, with technology advances serving as a catalyst for widespread adoption, as it was showcased in Section 3.2. There have been strides regarding inclusion for blind and/or visually impaired (Jackson 2012) individuals to ease communication between peers, for example. One of the central themes of the solution that is being developed is inclusivity for people with arm impairments and the support for these specific disabilities is extremely scarce in peer-to-peer messaging applications that, at the same time, upholds heightened levels of data security and privacy.

Notwithstanding, considering the solution being presented is split into two counterparts - non-invasive human-machine interface through EEG headset and subvocalization and a dedicated decentralized, scalable backend - there have indeed been studies conducted utilizing the EEG method that indirectly relate to the project at hand. For example, Katona and Kovari

(2015) conducted studies on EEG-based computer control interface for brain-machine interaction and netted positive results on user intention and consequence. Additionally, Kuber and Franklin P Wright (2013) applied the concept of EEG reading, similarly to our project, to the instant messaging panorama for detecting emotions and facial gestures. Findings from such evaluation revealed that this approach facilitated communication and strong levels of confidence were expressed when using the system to convey emotions which contributed to richer user experience. Although this feature can also be applied to our project, the use of such headset will be used primarily for navigation within the application. Furthermore, another project that is worth mentioning that makes use of EEG headset for instant peer-to-peer messaging is EmoChat, which the main objective is to showcase the emotions of users whilst chatting (Franklin Pierce Wright 2010). This feature is doable for the project but, as it was mentioned before, the main utility provided from the EEG headset will be utilized for navigation purposes.

Additionally, there are other interface hardware options that allow contactless interaction with the interface, including eye-tracking devices or speech-based commands.

However, one of the features that are simply not found in related work and applied to instant messaging panorama on a high level of accuracy refers to SSI or, in other words, subvocalization, where there has yet to be found an approach that yields a high level of accuracy (Rosello, Toman, and Agarwala 2017) (Wand, Janke, and Schultz 2014) for peer-to-peer communication.

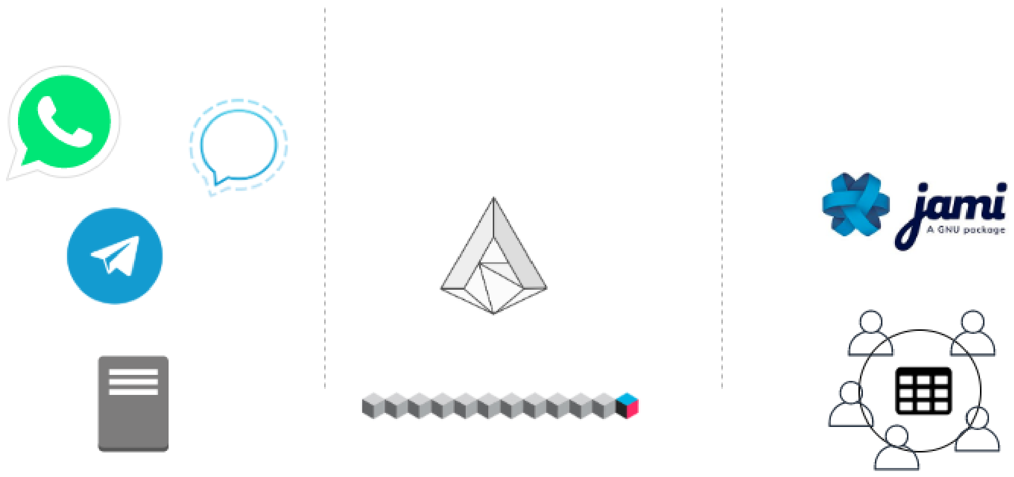


Figure 4.1: Diagram showing that WhatsApp, Telegram and Signal are decentralized, while Adamant is based on a pure blockchain implementation whereas Jami uses a decentralized system based on a distributed hash table.

On the other hand, referring to the decentralized backend, there are a myriad of solutions that are worth mentioning and that tackle different approaches that are further referenced in Section 4.1.2. These solutions represent different variations of decentralization and distributed systems and lean towards the messaging panorama and these examples are shown in Figure 4.1.

The secure transmission of data in communication has been studied and inclusively been already commercially reproduced. There have indeed been attempts to develop decentralized messaging applications that aim to address many issues present in centralized applications that are usually perceived as secure (e.g. Telegram) and allow decentralized authoritative messaging (Leavy and Ryan 2018).

The Telegram Messenger was created by Pavel Durov, the same individual behind the social networking site VK, largely used in Russia (Akbari and Gabdulhakov 2019). Telegram utilizes a centralized approach on messaging, more specifically cloud-based over IP. Telegram open-sources its frontend counterpart whereas the backend component is close-sourced. Each message and media in Telegram are end-to-end encrypted and, since it's a decentralized platform, stored in its own servers. Telegram's main selling point and marketing advantage rely on the privacy of communication between peers. However, Telegram's security model and use of custom-designed encryption protocol (MTProto) that has not been proven reliable and secure have received criticism by cryptography experts (Abu-Salma et al. 2017) (Jakobsen and Orlandi 2016).

Within the same centralized paradigm, Signal is a cross-platform messaging service where each connection and peer-to-peer communication is encrypted with perfect forward secrecy, using cellular telephone numbers as identifiers. The code for Signal is fully open-source and, as opposed to Telegram, Signal's security and cryptographic model is peer-reviewed and has been met with appraisal where, according to Cohn-Gordon et al. (2017), "have found no major flaws in the design, and hope that our presentation and results can serve as a starting point for other analyses of this widely adopted protocol".

Evgenov et al. (2017) proposed a platform based on blockchain which allows data/message transfers alongside integrated payment systems. Code-named *Adamant*, the project aims to showcase an example of fully anonymous, private and secure messaging between peers. The whole system is based on a platform-specific token and the latter is spent for each transaction (i.e. message). Inclusively, Adamant relies heavily on their anonymity principle where no personal or sensitive data is extracted from the user. Each message can be traced back to neither the recipient nor the sender since the whole platform is based on blockchain. A setback for this is the commissioned transactions and the throughput of the platform, which is extensively slower than mainstream applications. This is an example of a messenger solution based on the classic blockchain approach, build on an independent platform with a specific global consensus protocol.

There are other decentralized, blockchain-based messaging apps similar to Adamant, as it is the case with SenseChat (Sigman 2019). However, it was decided to delve into Adamant because the latter's main objective is true, full anonymity and security in the exchange of data.

Jami (formerly known as Ring) is yet another messaging platform that is distributed, differing from other decentralized applications on the underlying technology since it uses a distributed hash table instead of a classic blockchain approach. Similarly to Adamant the open-sourced Jami project aims to offer digital identity with accounts that are not linked with personal sensitive data. The connection is peer-to-peer, therefore both users must be online to be able to send messages. Undelivered messages will be stored locally on the device. Each connection is served with end-to-end encryption with forward secrecy between communications. Since the platform is distributed, no limits of file size and speed restrictions are imposed because the communication between peers is direct (Blin 2019).

As showcased in Figure 4.1, these solutions, albeit having some similarities when it comes to decentralization and message exchange, do not share the same technology stack with our proposed solution. However, our project is, similarly to Jami, based on a distributed hash table instead of going for a fully-fledged blockchain approach. A brief summary of some aspects regarding the discussed applications can be found in Table 4.1.

Table 4.1: Table comparing the different discussed applications on a set of characteristics.

	WhatsApp	**Telegram**	**Signal**	**Adamant**	**Jami**
No centralised storage for any part of the user data	Operator stores data of all conversations including images, video and files	Operator is able to log all data on servers	Operator stores data of all conversations including images, video and files	All user data is stored in the blockchain	Peer-to-peer, but there are intermediate servers which store undelivered messages
No explicit user identification	Mobile number is used for authorization	Mobile number is used for authorization	Mobile number is used for authorization	Yes	User account creation in the Jami Network
End-to-end encryption	There is a potential ability for operator to read all messages	Peer-reviewed and approved after skepticism	Peer-reviewed	Not yet peer-reviewed	Not yet peer-reviewed
Usage cost	Free	Free	Free	0.001 ADM (0.00024 USD) per message	Free
Delivery of messages	Less than 3 seconds	Less than 3 seconds	Less than 3 seconds	3-9 seconds	Less than 4 seconds. Direct messages can be sent only when the recipient is online.

To summarize, from all the current and recent solutions analyzed, within the decentralized messaging paradigm, there is not an apparent support for people with disability and for communication between individuals who suffer such impairments with those who do not. This can also be applied within the educational panorama, more specifically on communication between peers and peer-to-teacher. Our solution aims to improve some aspects on distributed messaging systems and also translate privacy and data security benefits. At the same time, it has support for impaired individuals offering a more inclusive and reliable system for communication with their peers.

4.1.2 State of the Art of Approaches

As it was shown in the previous section, there are a handful of projects and solutions that are based on different approaches towards the backend component. For this area and taking into account this project's purpose is mostly experimental and research-oriented, the possible approaches were narrowed down to three: centralized approach, blockchain-based decentralized system and blockchain-variants/DLT-based decentralized systems.

Centralized Approach

Centralized systems are widely used in a variety of situations spanning across the IT panorama. The client/server architecture is the *de facto* way of developing most of the applications and has gained support and momentum for the past years. It allows for easy maintenance and is well supported by a marketplace of service providers, such as AWS from Amazon and Azure from Microsoft.

It is worth noting how the *centralized* term is being used in this context. It strictly refers to the deployment of the product and where it is hosted. Centralized systems do not necessarily mean the inherent architecture, design and implementation are also centralized, or more commonly named as monolithic. Distributed models, such as the ones based on microservices that can, in fact, have distributed databases or follow a one-per-service database pattern, can be centralized in the way that they are hosted within the confinements of a single server or cluster of servers.

The microservices-based architectural model coupled with centralization in cloud computing has recently been under the spotlight, slowly shifting from a monolithic standpoint towards distribution, modularization and loose coupling provided by microservices (Mazzara et al. 2020). In fact, one of the main advantages one can obtain from developing a microservices-based application is the scalability on a long-term perspective and also the redundancy of each service for deployment, employing both horizontal and vertical scaling, both of which are represented in Figure 4.2. As shown in this figure, the microservices scale on the X and Y axis in terms of design and on the Z axis on the deployment phase.

Cloud hosting platforms are frequently associated with these type of architectures, providing horizontal scaling and thus adding availability and redundancy to the application as a whole, in terms of deployment. This effectively makes the platform centralized since it is hosted within the servers of the entities that provide such a service. Cloud computing and deployment is currently the preferred deployment method for companies and products to deploy their services and applications (C. Wu, Buyya, and Ramamohanarao 2018) because of the flexibility and ease of deployment it provides. Cloud computing, however, faces criticism from research conducted in the area regarding Infrastructure-as-a-Service (IaaS) platforms and alike, most noticeable regarding data confidentiality, performance unpredictability, scalable storage and bug propensity in large distributed systems (Josep et al. 2010).

With this in mind, serverless computing does, in fact, liberate people from deployment details and allows for focus on the core of the application. It is an abstraction but, in the end, it still runs on rented hardware with recurring costs and central points of failure.

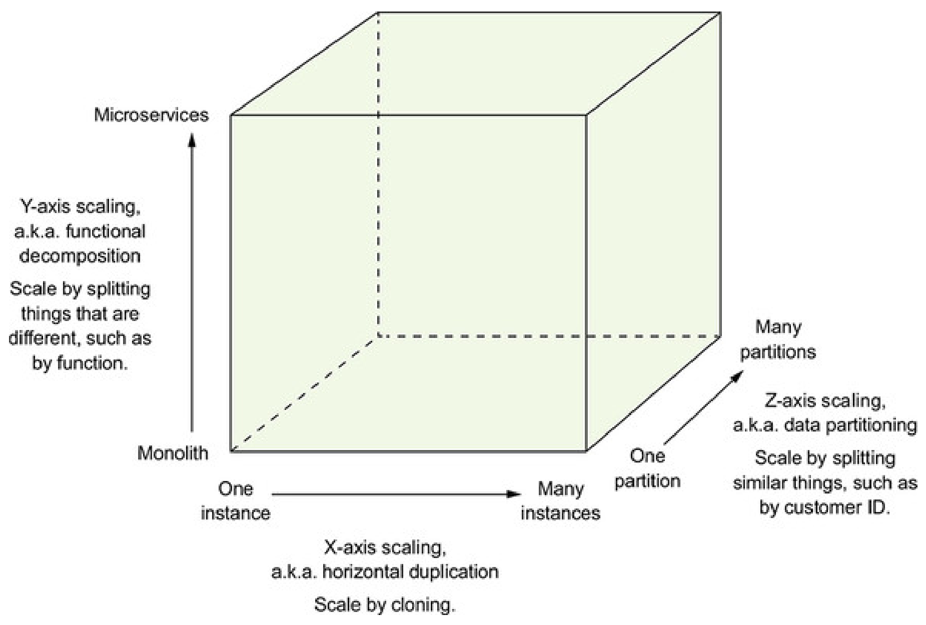


Figure 4.2: The Scale Cube first outlined by Abbott and Fisher (2009) showcasing the three dimensions of scalability.

Blockchain Approach

In comparison to the previous approach, generally, distributed computing efforts, like blockchain, attempt to tackle the aforementioned issues by creating a network of participants who all hold a public, global data set. Each node aids maintaining the availability and the integrity of the data through global consensus protocols, thus ensuring all data that is added to the blockchain is reliable across all users. Most of the centralized deployment vulnerabilities are eliminated because, in essence, the deployment is not centralized to a physical location but rather distributed across all the users that make part of the blockchain - everyone holds its history. Blockchain is, all in all, a growing list of records that are linked through cryptography, each block having a hash of the previous block, timestamp and transaction data (Nakamoto 2008).

The original concept of blockchain was originally pioneered by Nakamoto (2008) with the purpose of serving as a public transaction ledger for Bitcoin, a cryptocurrency. This new approach addressed the double-spending problem without the need for a central authority regularizing each transaction. From this stemmed other applications and blockchain-variants, either permissioned or permissionless and for business use which incorporated not only the support for cryptocurrencies but also for business logic, through the use of smart contracts, something pioneered by Ethereum (Wood et al. 2014) (Buterin 2013).

Currently, within the blockchain paradigm, the most debated issue and news regard Ethereum's shift from Proof-of-Work (PoW) consensus to Proof-of-Stake (PoS) protocol which translates in a myriad of advantages regarding electricity consumption, reduces centralization risks and sets up the blockchain for higher scalability and Transactions-per-second (TPS) potential through implementation of sharding of the blockchain (Tikhomirov 2017).

One of the gripes about current blockchain technologies refers to the limited TPS and scalability problems that are present in the two biggest cryptocurrencies in the market: Bitcoin and Ethereum. Previous solutions to address this issue temporarily were to make blocks of transactions bigger so these could handle more transactions. This, of course, entails a higher computing power from the miners - users who uphold the blockchain integrity by undertaking processing power to maintain the state of the blockchain.

From this necessity and one of the current battles within the blockchain world arose the term of Layer 2 solutions. These solutions are intended to be built on top of the current blockchain solutions (Layer 1) and address the scalability problem that currently plagues the blockchain paradigm. These solutions have been researched and implemented in Bitcoin, by the name of Lightning Network (Figure 4.3) - where a temporarily communication channel between two parties to make transactions and after closing the channel, the outcome is relayed back to the blockchain, making it a much faster process for transaction - , and Ethereum, code-named Raiden and also making use of sharding. These are designed to speed up the overall system by offloading transactions to a secondary network, thus achieving a much higher TPS while still connecting to the main blockchain and maintaining its important characteristics: distributed, permissionless, and trust-minimizing (Poon and Dryja 2016).

As shown in Figure 4.3,

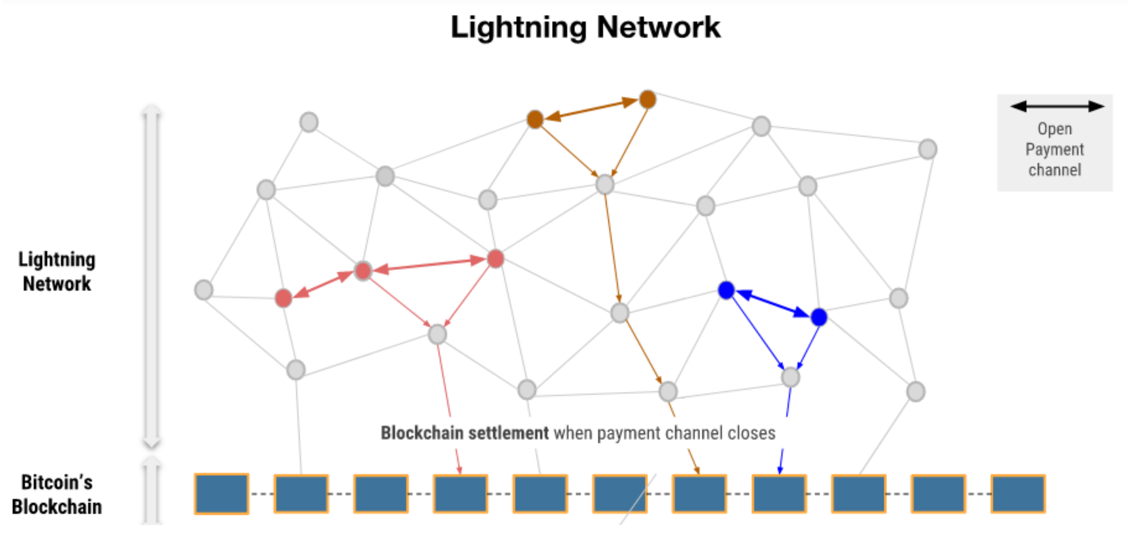


Figure 4.3: Lightning network created by and for Bitcoin (source: (Obondo 2018)).

All in all, the main issues that are present in blockchains deal with its speed (that is more prevalent in permissionless blockchains) and scalability as new nodes join within the network. Notwithstanding, the security issues and data integrity are main factors that drive blockchain forward and make it a great application for finances, making each transaction tamper-free and less costly than contemporary means.

DLT/blockchain-variant Approaches

The third and last type of approaches being discussed is usually referred to as Blockchain 3.0, even though these approaches and designs do not necessarily make them relate to the concept of "blockchain". As it was mentioned in Section 4.1.2, Bitcoin and Ethereum, the two largest blockchains and platforms currently respectively, to date, have not yet achieved substantial scaling and TPS success as opposed to, for example, Paypal. The reason for this lies on the technical design choice that is inherent to the blockchain; it is because of global decentralization and equality between all peers and global consensus to add data to the blockchain. Every node processes every single transaction and has to store the entire blockchain history.

Bitcoin and Ethereum's attempt with Lightning Network and Sharding, respectfully, as it was aforementioned, agree that the answer to the current scaling problem is for that not all nodes within the blockchain need to know all the information at all times to be in sync with the network. This is the principle that Directed Acyclic Graph (DAG) bases on. Essentially, a DAG is a network of individual transactions that are linked to other (can be multiple) transactions. Therefore, the concept of blocks does not exist in the DAG. Whilst blockchain can be compared to a linked list, a DAG is a tree of transactions. Consensus across the DAG is achieved by validation of transactions block by block (Dillenberger and Su 2019). Individual transactions provide validation for one another, as shown in Figure 4.4. Generally, nodes are both miners and validators, albeit not being able to validate their own transaction. This translates in fewer fees to pay to handle a transaction. All in all, this means networks based on this approach scale very efficiently and are well suited to high volumes of transactions - the higher, the faster a DAG validates each transaction (Benčić and Žarko 2018).

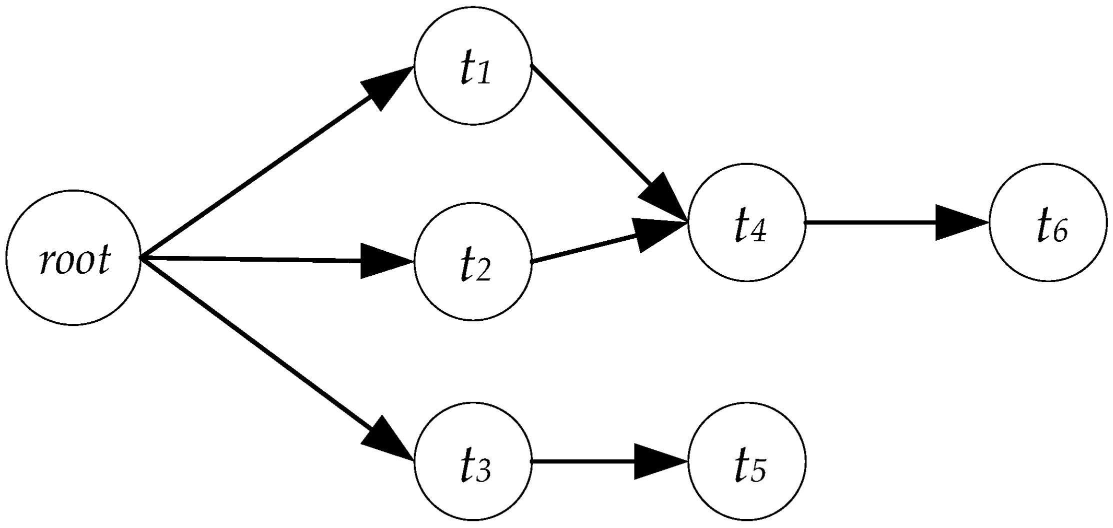


Figure 4.4: Diagram representing how a DAG works. Each node is validated by the previous one. In this example, the node t_4 is connected from two parents (source: (Xu et al. 2019)).

In addition to the DAG, there is another approach to this problem that fits within post-blockchain/blockchain-variant solutions. Hashgraph also falls into the category of DLT. Similarly to DAG, the concept of Hashgraph is not based on blocks that are chronologically linked to create a chain. Instead, it is based on events that are hashed to each other, as shown in Figure 4.5. Each event contains a timestamp, two different hashes from the parents and multiple transactions. Whilst in the blockchain, the miner that successfully mines the new node and adds the new block to the chain, in Hashgraph all nodes that are in the network inform each other and exchange their information with each other. Similarly to blockchain and DAG, all transactions are arranged chronologically. This transaction history allows a consensus on the sequence of individual transactions (Baird 2016).

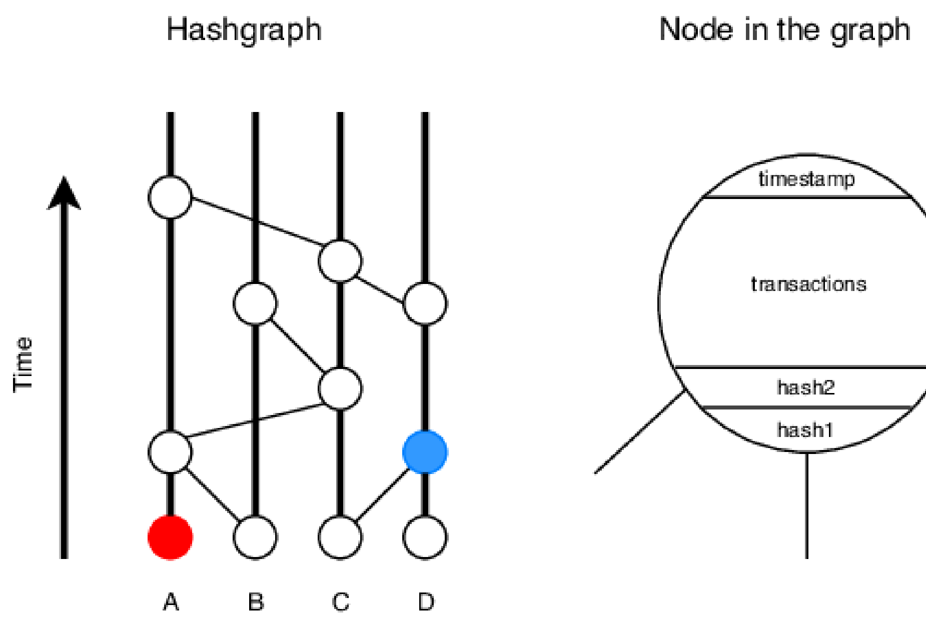


Figure 4.5: Figure showcasing a rough model of an Hashgraph and the information residing in each event. A, B, C and D refer to different nodes/users within the network (source: (El Ioini and Pahl 2018)).

The information within the network is disseminated via the Gossip protocol, which is a communication protocol. This essentially entails random gossiping to a node that, in turn, will gossip to another randomly selected nodes, resulting in exponential propagation. Additionally, each node knows the transaction history and, therefore, the exact sequence of individual transactions. Therefore, when a node gossips with another node, the information relayed includes which nodes spoke to what, to whom and when, thus transmitting the entire transaction history. This is a new form of global consensus within the network (Baird 2016).

Another approach that is also distributed is through the use of a DHT, which provides a lookup service similar to a hash table. Each key-value pair is stored in a DHT and nodes that are in part of the network retrieve the value associated with a key (Galuba and Girdzijauskas 2009), as shown in Figure 4.6. The management of the DHT is done by the peers within the network and are responsible to maintain its state. The DHT is also shared amongst every node. This system is often used in peer-to-peer file sharing systems, such as BitTorrent's distributed tracker. Using this method allows for fault-tolerance and the system to be reliable even with nodes leaving, joining or even failing. The scalability issue is resolved, the system functioning efficiently regardless of the number of nodes (Zhang, Goel, and Govindan 2003), as showcased in Section 4.1.1. This is a method that is used by Jami to find system user's IP address.

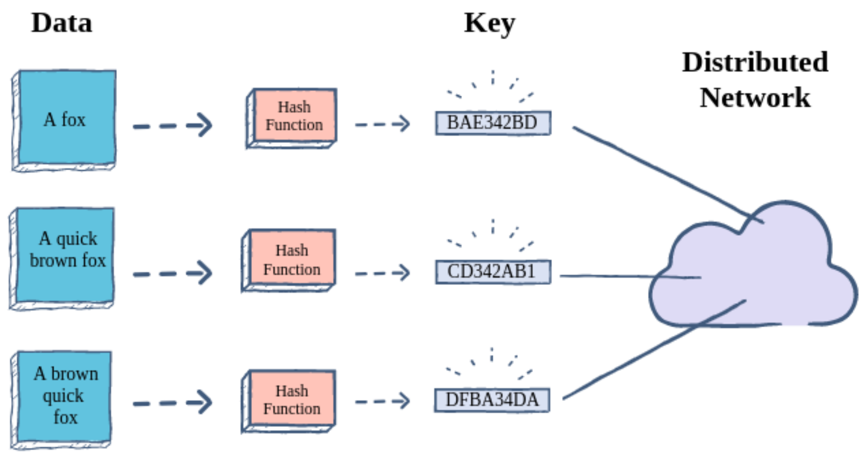


Figure 4.6: Diagram showcasing a simple example of a DHT. Each piece of data is hashed and paired with a key that can later be retrieved in the DHT by peers of the network (source: (Edspresso 2019)).

4.1.3 State of the Art of Technologies

Following the description of the currently available and most preponderant technical approaches there is for the problem at hand, this section will focus on the technologies/frameworks that facilitate the development and implementation of applications in each respective approach. Each framework will be briefly showcased and described in order to obtain a clear comparison between them in each chosen approach.

Centralized Approaches

Discussing current technologies of centralized solutions can be referred to both monolithic or microservices-based architectures of solutions, as long as it means these are hosted on servers (or clusters). The current state of this ecosystem is the culmination of many years of maturation. There are not go-to solutions, frameworks or technologies because this paradigm is simply too broad. Programming languages such as Java, Javascript, C#, Python or Go are all viable options to design and implement solutions, those either being microservices or not. For example, Signal's messaging application is centralized and written in Java, with its protocol being implemented in C, Javascript and Java, inclusively (Cohn-Gordon et al. 2017).

For example, ASP.NET Core, being an open-source modular multi-platform framework developed by Microsoft, is one of the frameworks that allow the development of web applications and backend services for a wide range of scenarios, with an official ORM plugged and with access to asynchronous programming, also allowing dependency injection (Hur and Ubelhor 2017). For the case with Java, that is used for the Signal messaging app, Spring Boot is often associated with the development of such services with a robust framework that is easily plugged with gateway technologies such as Redis or Zuul, as well as having a softer learning curve, thus effecting positively the development cycle (Walls 2016). These aspects could also be analogous to Node.js, for Javascript-written programs, which provide asynchronous I/O concurrent requests.

Referring to the microservices paradigm, Spring Boot is often associated with this type of architecture. One prime example of an API gateway is Spring Cloud Netflix, a Java stack which provides support for various patterns used on microservices architecture, such as Service Discovery (using the framework Eureka), Circuit Breaker (Hystrix), Routing (Zuul framework) and client-side load balancing (utilizing Ribbon) (Cosmina 2017). For the messaging pattern utilized in microservices, publish-subscribe technologies such as RabbitMq and Kafka are the two most reference frameworks for communication between microservices (Dobbelaere and Esmaili 2017).

The Go programming language has been increasing in popularity and is both present on both monolithic and microservices-based architectures, mainly due of the community support, the easy-to-learn syntax and optimized memory management features it provides, rivalling high-performance languages like C, C++ and Rust (Balbaert 2012). Utilizing in Go for microservices is also possible because of the creation of frameworks that allow cross-cutting concerns regarding networking and sockets to be something abstract from the developer, all the while achieving high levels of performance on requests, and on compilation and execution times when regarding the development cycle (Domingues 2017).

Notwithstanding, the current *status quo* of traditional development is not narrowed enough to warrant a thorough analysis of each technology. It is, in fact, leaning out of the scope of this thesis, since it pretends to delve deeper in blockchain-related technologies and approaches.

Blockchain Approaches

As it was previously mentioned in Section 4.1.2, the blockchain technology is not only used for cryptocurrency and transaction exchange. The concept of smart contracts - a piece of code that is present within the blockchain which automatically executes when met with specific conditions and allows for the exchange of assets beyond digital currency (Buterin 2013) - was originally introduced by Ethereum. In fact, the latter, unlike Bitcoin, has grown to provide a platform for developers to develop Decentralized Applications (dApps) on, provisioning with tools to build these.

Therefore, Ethereum, besides being a blockchain with its own currency, is also a platform for users to develop applications within the same blockchain. This is possible by the Ethereum Virtual Machine (EVM) that runs on the Ethereum network, enabling anyone to run any program, regardless of the programming language given enough time and memory. Therefore, instead of having to build an entirely original blockchain for each new application, Ethereum enables the development of these in a single platform (Buterin 2013).

Ethereum is an example of a permissioned blockchain. In other words, Ethereum's network is public, meaning any application developed over the network can be seen by anyone and is maintained by miners. Developers have no control over this factor. In contrast to this, permissionless blockchains deal with private blockchains, requiring authorization to partake within the network. Within this panorama Linux Foundation's Hyperledger framework arises, with the main goal of allowing developers to build permissioned blockchains, specially tailored to Business-to-Business (B2B) transactions. The main differences between Hyperledger and Ethereum rely on confidentiality, where the latter is more transparent in lieu of the former, where transactions are confidential; on consensus mechanisms, where Ehthereum currently employs PoW (albeit shifting to PoS consensus whereas Hyperledger has pluggable consensus algorithms, making mining unnecessary; on the programming language, which Hyperledger's chaincode is written in Golang whilst Ethereum's smart contracts are written in Solidity; and on cryptocurrency, with Hyperledger not having a built-in cryptocurrency, as opposed to Ethereum (Androulaki et al. 2018).

Hyperledger effectively targets a more corporate-leaning target audience and facilitates the development of permissioned blockchains. These, compared to what Ethereum offers, provide a higher level of performance/TPS, a higher resilience by the deployment of redundant peer nodes and better privacy, through the form of confidential transactions which hold sensitive business data. Additionally, Hyperledger adopts a modular architecture that supports plug-in components (Androulaki et al. 2018). The consensus protocol for Hyperledger is based around validating (who validate transactions and maintain the ledger) and non-validating nodes, as opposed to miners. Hyperledger allows the developers to choose between No-Op, or an agreement protocol to reach the verdict. Therefore, all the parties involved agree in such a way that everyone can influence the final outcome. This participation restriction allows for better scalability and privacy (Androulaki et al. 2018). Some key differences are showcased in Table 4.2.

Table 4.2: Table comparing key features between Hyperledger and Ethereum.

FEATURES	**Hyperledger**	**Ethereum**
Purpose	Tailored for B2B networks	Generalized applications
Peer participation	Permissioned network	Permissionless network (public)
Consensus	Pluggable Consensus Algorithm: no mining required	PoW consensus protocol that entails mining (shifting to PoS)
Confidentiality	Transactions are confidential	Transactions are transparent and public
Built-in cryptocurrency	No	Yes
Programming Language	Golang	Solidity

On another hand, the Cosmos framework, which incorporates the Tendermint technology, aims towards interoperability between blockchains. Actually, Cosmos aims to become the "internet of blockchains" which targets the problem of interoperability between blockchains, regardless of permission status. Strictly speaking, "Cosmos is a decentralized network of independent parallel blockchains, each powered by Byzantine Fault-Tolerance (BFT) consensus algorithms like Tendermint consensus" (Kan et al. 2018). In other words, Cosmos is an ecosystem of blockchains and aims to solve the scalability and TPS problems inherent to common blockchains. For Cosmos, a typical blockchain architecture is composed of three conceptual layers: application, which is responsible for processing transactions; networking, tasked for the propagation of transactions and consensus-related messages; and consensus, the protocol that enables each user to agree on the current state of the system. This is showcased in Figure 4.7.

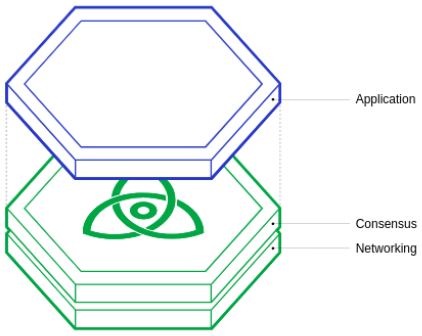


Figure 4.7: Common blockchain architecture with three layers: application, networking and consensus (source: (Kwon and Buchman 2019)).

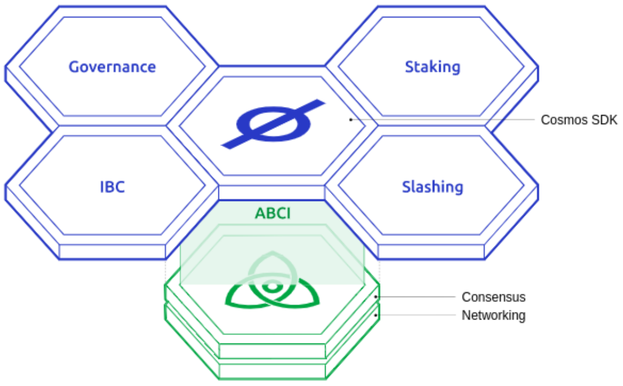


Figure 4.8: Overview over the Cosmos' framework, making use of Tendermint and the Cosmos SDK (source: (Kwon and Buchman 2019)).

To achieve Cosmos' vision of communication between blockchains, it provides a set of open-source tools that include Tendermind, the Cosmos SDK and an Interblockchain Communication Protocol (IBC). This whole framework allows people to build custom and scalable blockchains within the ecosystem of Cosmos. Tendermint can be analogous to an engine which deals with the networking and consensus layers of the blockchain, allowing the developers to focus on application development through Cosmos' SDK, that is accessible regarding of programming language (shown in Figure 4.8 (Kwon and Buchman 2019). The main advantages of using this framework are the level of abstractions between modules and the possibility of the developers to create a blockchain that is customized without needing to waste development time on creating an networking and consensus infrastructure, all with the advantage of being interoperable with other blockchains that use the same Cosmos' framework, allowing for exchange of data between them.

DLT/blockchain-variant Approaches

In lieu of classic blockchain frameworks, there are indeed frameworks and technologies that ease the development of decentralized systems based on variants of the blockchain concept, namely DAG and Hashgraph.

For the former, the IOTA technology surfaces as one of the most well-maintained technologies which utilizes a DAG-based approach for the development of decentralized and distributed systems. IOTA is an open-source DLT which allows connected devices to transfer data and its own property token among each other with no fees. Similarly to the blockchain, nodes consist of the backbone of IOTA's network and immutable record of transactions which is called the Tangle. Each node runs the same node software, allowing these to validate each transaction accordingly (Popov 2016), as shown in Figure 4.9. By utilizing this framework, it is expected for a higher level of confidentiality and security, by the use of one-time signatures on each transaction; trust, where every node validates transactions and gossips the outcome exponentially across the tangle; and scalable, because each transaction that is attached to the network, two previous transactions are validated, making IOTA scalable as new transactions lead to faster validations (Alexander 2018).

Any developer can start and develop their own networks on top of Tangle on either the Go programming language, Java, Javascript or Python. Moreover, IOTA provides tools for developers to create their own private Tangle, which is essentially a DAG-based network where the developer has control over it with a tool named Compass, which the developer can use to allow nodes to reach consensus on transactions occurring in the private network created (Divya and Biradar 2018).

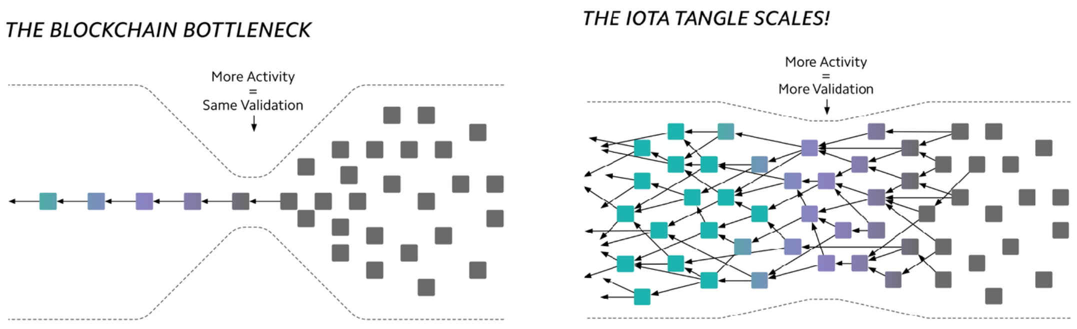

THE BLOCKCHAIN BOTTLENECK
More Activity = Same Validation
THE IOTA TANGLE SCALES!
More Activity = More Validation

Figure 4.9: Difference between a classic blockchain approach with a variant with the IOTA framework for DAG-based networks. The difference is in the scalability, where blockchain decreases over time, IOTA increases (source: (Divya and Biradar 2018)).

Referring to the Hashgraph approach, Hedera is the only DLT platform that uses the hashgraph consensus. The approach, as it was mentioned in Section 4.1.2, is fast because of the Gossip protocol employed, provides fairness via consensus time stamping (meaning that if a transaction reaches two-thirds of the network ahead of the rest of transactions, it is therefore considered the first) and is more secure, being asynchronous BFT. Being BFT means no group of members can prevent the overall network from reaching a consensus, nor change its outcome once the consensus has been reached (Baird 2016).

One can build and deploy a decentralized application on top on Hedera's network, using its services that are offered, which include cryptocurrency, smart contracts (that are, similarly to Ethereum, written in Solidity) and file services, all atop hashgraph's consensus algorithm. The current network consists of permissioned, public nodes, although it is planned to move towards a permissionless model in the future (Baird, Harmon, and Madsen 2018). Currently, Hedera provides two officially supported SDK for the Java and Javascript libraries. However, there are community-supported SDK's for support in Python, .NET, C, Rust and Go.

Finally, referring to the DHT approach, there are two possibilities in which one can develop their applications. One of them refers to OpenDHT which is, essentially, a distributed hash table implementation in C++14 or, in other words, a distributed in-memory data store, in which every node within the network can read and write values to the store, with values being distributed over the network with inherent redundancy (Rhea et al. 2005). As it was mentioned in Section 4.1.2, this results in a much more faster and scalable solution compared with the rest of the aforementioned networks. Additionally, with this technology, public-key cryptography is provided for data signature, support for Python 3 in addition to

C++ and a RESTful Application Programming Interface (API) for HTTP-based client/server applications (Béraud 2019).

The second technology with a DHT inherent is Holochain which is a framework that is comparable with OpenDHT, albeit with a few technical exceptions. The first one is the robustness of the framework, being that it offers a testing framework on Javascript in which the developer can do functional, end-to-end and unit tests. Holochain is a framework that, essentially, allows its developers to focus on the core application logic but also allow to customize and tinker with networking and consensus protocols inherent to their own decentralized networks. In Holochain, every user runs the same version of the code and utilizes DHT to share information between them which is, in turn, distributed across every node within the network, thus reducing central points of failure and bottlenecks (Harris-Braun, Luck, and Brock 2018). With the validation rules of the application being present in every peer within the network, each participant can validate each other's data and exclude disruptive peers through gossiping with random peers. Additionally, each participant within the network supplies their own compute and storage resources. The Holochain framework allows development solely in Rust, a low-level language.

4.2 Approaches Assessment

Continuing from the previous section, after providing different approaches and their respective state of the art, a thorough analysis of comparing these for the problem at hand will be delved into this chapter. Although some traces of comparison can be found in the previous chapter, this chapter will showcase a more detailed analysis and contrast each methodology with a set of criteria that are fit for the problem proposition. Ultimately, the choice will be explained coupled with outlining testing methods for these solutions in order to verify the choice made.

4.2.1 Criteria and Approach Comparison

The approaches that are taken into consideration were already explained in Section 4.1.2 and have slightly been compared. Additionally, through the AHP method (refer to Section 3.6), the best approach was already selected. Nonetheless, a specific set of criteria were established and are the backbone of making such a decision.

The centralized approach, although considered, was quickly dismissed mainly because the project at hand aspires to be research-oriented and provide a new perspective on inclusiveness within the decentralized panorama, which is rather scarce, as it was debated earlier in this document. Choosing a centralized methodology would go against the hypothesis that was firstly stated in Chapter 1.

Notwithstanding, a centralized approach strictly refers to the deployment of the application. As it was mentioned in Section 4.1.2, an application can be based on microservices (which is distributed) but still be centralized because it can be hosted on physical, targeted servers or on cloud platforms, even if these would offer horizontal scaling or serverless computing features (this was explained in Chapter 2). To make an application which can be deployed without the need of servers and relying solely on its peers, distributing computing efforts was the main resolution to the criteria of innovation this project attempts to tackle. Therefore, albeit the centralized solution has great benefits, including its well-establishment within the industry and a wide range of well-documented technological options to go by, its implementation is not adequate for the scope of this project. Additionally, decentralization allows for a more democratic and equal distribution of data, computing power and influence of each peer within the network, which is one of the assets this project aims to target.

As it was mentioned in Section 3.6, the criteria for which the approach was decided upon was also based on throughput/performance of the application, security of data and computing decentralization. These three aspects are highly debated within the blockchain ecosystem and are often known as the "Impossible Triangle", which states that for a public blockchain, achieving proper decentralization, data security and consistency and scalability is impossible (Buterin 2019). To leverage these three on its full capacity has been dubbed as impossible, hence the name "Impossible Triangle" (as shown in Figure 4.10), named by Buterin (2019).

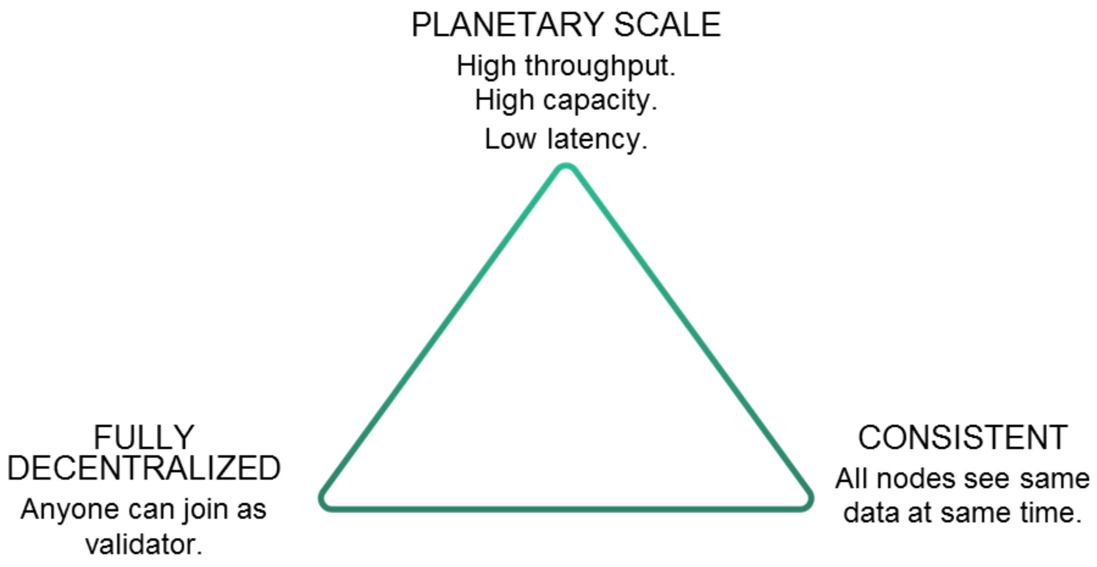


Figure 4.10: The trilemma triangle, also known as the "Impossible Triangle", explains that one can fully achieve only two of the three aspects within it (source: Buterin (2019)).

The Scalability Trilemma claims that blockchain systems can only, at most, have two of three probabilities, those being *scalability* which can be roughly analogous to performance (TPS); *security*, defined as being resilient against attackers with up to $\mathcal{O}(n)$ resources; and *decentralization*, which means that the system is able to run in a scenario where each participant has the same amount of importance and has access to the same amount of resources. Ethereum has claimed to solve this trilemma through sharding (Buterin 2019). The way sharding works is to split Ethereum's history into "shards" (partitions), each one having its own transaction history and managing their subset of transactions. Each shard can communicate with each other and this, subsequently, makes it that Ethereum can process transactions in parallel, rather than synchronously. Therefore, the system's throughput is not limited to a single node's capacity, as it was previously. Nonetheless, sharding has yet to be implemented and rolled out to production in Ethereum's platform. Besides, people argue that sharding lessens decentralization (Bez, Fornari, and Vardanega 2019).

Nonetheless, even comparing with the centralized approach, blockchain fares better and seems to be more adequate for the application, even if it achieves at most two of decentralization, scalability and data consistency. Most modern mainstream messaging applications claim to use modern, quantum-proof encryption methods to boost user's privacy and security. However, as is the case with Telegram, for example, these covert proprietary source codes are not peer-reviewed and can not, therefore, make conclusive claims about data security, which is why open-sourcing this project's code (alongside other decentralized messaging applications) is one criterion for this project, so as to be reviewed and accepted as a secure solution. Additionally, another issue that almost all messengers have are the requirement of direct access to device's address book and pass, alongside other sensitive data (such as device ID, phone number, name of the user), to their servers which can, in turn, be vulnerable to attacks. This approach creates a great threat of leakage and unwanted data usage on all stages of interconnection. Moreover, user IP-address disclosure, that happens when connecting to central servers) is yet another problem which most current messenger users encounter at the security level. For this, utilizing an approach with blockchain can remedy these concerns, since every node is simply known through its address and unless otherwise, can not be associated with a specific entity. The term of anonymity is vastly addressed within blockchain systems. Furthermore, data integrity is also assured by blockchain's inherent technology, as it is decentralized and invulnerable to attacks that are normal when compared to centralized solutions.

However, one criterion of this project, besides having decentralization and proper data security, is to be scalable as time progresses and more users join the network. With the current state of the blockchain ecosystem, achieving is difficult because of the lack of platforms that do so. EOS was considered to develop such platform but it is widely considered to be a "centralized decentralized" platform to develop dApps, mainly due because of their consensus platform based on delegated proof-of-stake that gives more power to nodes that have a higher amount of tokens. In other words, the power of decision is not common amongst all nodes and is instead concentrated in a few nodes, which goes against one of the principles of blockchain. This highly correlates with what is shown in Figure 4.10, specifically the "Fully Decentralized" vertex. One of the reasons this trilemma still plagues the blockchain ecosystem is mainly because of the consensus protocol employed by these blockchains. All of these target a global consensus protocol, in which all nodes need to form a mutual agreement for all the data that enters the system. To have this global consensus on a messaging panorama not only is excessive, but it is also unnecessary, hence why blockchain is not adequate for the project at hand.

Notwithstanding, the consensus concept is tackled with a different perspective if a DHT-based approach is to be used. When it comes to peer-to-peer communication, a global consensus is irrelevant between the two peers making a communication. Even in situations with chat rooms, the scope is much smaller. The question with data integrity can be solved by each client have the same code and abide by the same "rules" of the network, therefore having an *a priori* stance on data input into the system. Besides, consensus would only be feasible in conflict scenarios. It is more efficient to bear the computational and overhead cost of achieving global consensus only on these cases and treat it as a special scenario, not the norm. This simple alteration makes it that the system has a significantly higher throughput and performance across the board (D. Wu, Tian, and Ng 2006), meaning there is a less delay in delivering messages. In fact, as more users partake in the network, more computational power is added, therefore the performance of the network grows linearly over time.

Another issue in which a DHT can uphold levels of security regards the data residing within. A proposition of using a DHT within the messaging context would be to hold user IP-addresses so they are discoverable from other peers within the network. This, of course, raises security issues. However, they can be addressed by generating temporary public/private key pairs to be used whilst making connections. Additionally, onion routing [1] can be employed to store and locate these addresses and to make it more difficult to associate two different users together.

In addition to all of this, if the Holochain framework is to be used, interoperability between networks is achievable, since all of these share the same technology and communication protocol. This can be useful for future prospects of the project and expand features that encompass other assets that are part of the ecosystem.

4.2.2 Chosen Approach

In the light of the above, it has concluded that going for a decentralized, blockchain-variant approach is most adequate for the project at hand, considering it is within the peer-to-peer messaging paradigm, regardless of the design choice is based on DAG, Hashgraph or DHT. One of the reasons for this regards the scope of the project, which leans toward innovation and is research-oriented and meant to showcase a new approach on decentralization within the peer-to-peer, messaging panorama with disability support through no physical contact regarding the human-machine interface and trace comparisons between blockchains and one of its variants. Being that the goal of the project is showcasing a decentralized platform that has a competitive performance when compared with centralized applications, this is a sensible choice.

Regarding the interoperability, both blockchain and its variants have platforms (Cosmos and Holochain, respectfully) that allow the development of applications that are decentralized and can be interoperable with other blockchain/systems because they also share the same framework and communication protocol.

Another reason why a DLT/blockchain-variant approach was selected was, comparing to a centralized solution, at a business level, it lessens the costs of deployment to zero (to the entity that creates the platform, as the deployment is shared across its peers, the expenses being the responsibility of the agents that partake in the network) because the system by itself is hosted on each peer that constitutes the network. Additionally, the concept of decentralization panders to the goal of the application to be more secure, with a lower amount of possible points of failure and a more democratic ecosystem where each peer has the same influence over the network.

Comparing a, for example, DHT design (that is inherently a blockchain-variant approach) to a classic blockchain system, the former is much more scalable (Harris-Braun 2018). With the technologies inherent with this approach (as it was discussed in Section 4.1.3), the performance scales linearly with new users since every node contributes useful computational power and storage resources to the network. Although some of these aspects can also be found in blockchain systems, the consensus protocol used is global and it is mandatory for every node to comply, denting the throughput of the system in favour of full data integrity

[1] Technique for anonymous communication. Messages are multi-layered with different layers of encryption, each one being decrypted at each onion router the messages reaches its destination.

within the chain. This aspect is more customizable in a DHT approach and the consensus protocol for data does not necessarily need to be global.

Chapter 5

Application Requirements and Design

Throughout the planning and development phase of the project, some diagrams were created so as to have a better grasp of the business rules and how the development cycle would unfold. These aid in helping design each step of the project, identify key components for the final working application and lay out the relationships between these. This chapter will focus on the overall concept of the application, its architecture and design choices, as well as some processes that are inherent to the application. Technical specifications will be showcased by the use of Unified Modeling Language (UML) diagrams.

5.1 Functional and Non-functional Requirements

As with every project, the design process must be preceded by a thorough requirement analysis. In this chapter, the functional and non-functional requirements of this project will be explained according to the Software quality classifying model standing for Functionality, Usability, Reliability, Performance and Supportability (FURPS)+[1].

5.1.1 Functionality

The FURPS model usually pertains to the definition of non-functional requirements in lieu of functional requirements that directly related with use cases. However, in this case, the Functionality subset will tend to both aspects: the non-functional requirements and use cases from functional requirements.

Regarding the latter, representing the functional requirements of the project which, in turn, are related to the use cases and user stories of the application, as it was mentioned in Section 2.3.2, the current project does not have several use cases, since the focus will be on providing an inclusive and safe experience. The following Table 5.1 depicts the use cases of the system.

When it comes the non-functional requirements of the project, regarding Interoperability, the system is set to be compatible with other decentralized systems under the same protocol and be possible to exchange information with other and maintain a common profile. Furthermore,

[1]FURPS+ is a model for requirement specification in projects. It represents an acronym, meaning Functionality, Usability, Reliability, Performance and Supportability, thus dividing the model in five domains. The "+" symbol represents other attributes.

Table 5.1: Table showcasing the use cases of the system and the correspondent actor.

Actor	Use case
As a **common user**, I want to...	Create contact
	Remove contact
	List all contacts
	Choose contact/chatroom
	Send message
	Receive message

the Portability aspect of the system should pertain to its ability to be run regardless of the target platform, this either being mobile or desktop.

In addition to these, the functionality aspect of the application also correlates with Security. For this project, the Security of the application has as requirements the existence of a public/private key pair to prove the authorship of data to specific users and to be used on communication with other peers. Another requirement is it is mandatory for all users to have the same version of the backend/frontend bundle and, therefore, system configuration. Communication between users also has to be end-to-end encrypted.

5.1.2 Usability

This topic covers the requirements regarding interaction with users through user interfaces. For this project, usability is key given the target audience and their capabilities in commonly used interfaces, which we are trying to overcome.

One of the requirements for the User Interface (UI) is to be intuitive and compatible for usage through an EEG headset. Therefore, the frontend has to have a simple and intuitive design for users from different backgrounds and age ranges. Moreover, the frontend must also allow support for text input through silent speech using SSR techniques. Finally, the frontend must have a consistent design spanning its screens and be responsive.

5.1.3 Reliability

This section concerns requirements related to service availability and failure recovery. For this project, the project at hand must be able to handle increased network traffic. Additionally, no central points of failure that propagate and affect several users may occur. In addition to this, redundancy of data is a requirement for users to maintain data even if the original author is offline.

5.1.4 Performance

Performance covers requirements associated with the response time and recovery time of the system. Specifically to this project, the system must have a reduced response time and prompt feedback. Moreover, the time of each transaction must be low and consistent as the system scales with more users joining the network.

5.1.5 Supportability

For this section, supportability is concerned with characteristics such as testability, adaptability, maintainability, and scalability. The latter aspect is a requirement, it being scalable without compromising performance to a limitless amount of users. Additionally, the platform has to be available in all operating systems and web browsers. The system has to be inclusive for users with upper-body limb impairment and be able to pander to both regular and disabled users and maintain usability. Lastly, the system must allow text input from the silent speech feature for at least the English language.

5.1.6 + (Others)

This final section covers physical, design and implementation constraints. For this project, one constraint indicates that the system will use the Emotiv EPOC+ headset, as it is the only EEG device the team has access to at this current time.

5.2 Conceptual Design of the Application

The problem that is tackled by this project and by this report regards the scarcity of secure, scalable, interoperable and decentralized messaging solutions with support for individuals with arm/hand impairments and vocal disabilities.

The proposed solution encompasses SSI and BCI for the interaction of an application that partakes in a decentralized ecosystem that is user-centric, with high throughput and maintaining a bottom-line of privacy of data and message exchange. It envisions a way for people with arm/hand and speech impairments to be able to communicate with anyone.

It is, therefore, divided into two distinctive counterparts: human-machine interaction through methods of machine learning and decentralized communication between peers. This is further outlined in Figure 5.1.

On a conceptual point of view, for the BCI component of the project, an EEG headset (more specifically, the Emotive EPOC+) will be used alongside the Emotiv BCI software for the training and classification of the data gathered from the user. Some mental commands are already predefined, which speed up the project development considerably. These commands are applied to the UI navigation. The latter ought to be accessible and intuitive for users with or without impairments, with a specific design that allows navigation through an EEG headset or conventional means (mouse movement). Text input will be, on the other hand, dealt by the subvocalization counterpart, through a pair of EMG surface electrodes.

The other counterpart refers to the decentralized backend in which each peer holds part of the DHT and information is shared and verified by the nodes that make up the network. Each node has its own identity and private and public shared data. Each data has ownership of a specific user, in which the user has control of. Having this type of system allows for future expandability of features and expand from peer-to-peer messaging to a social network

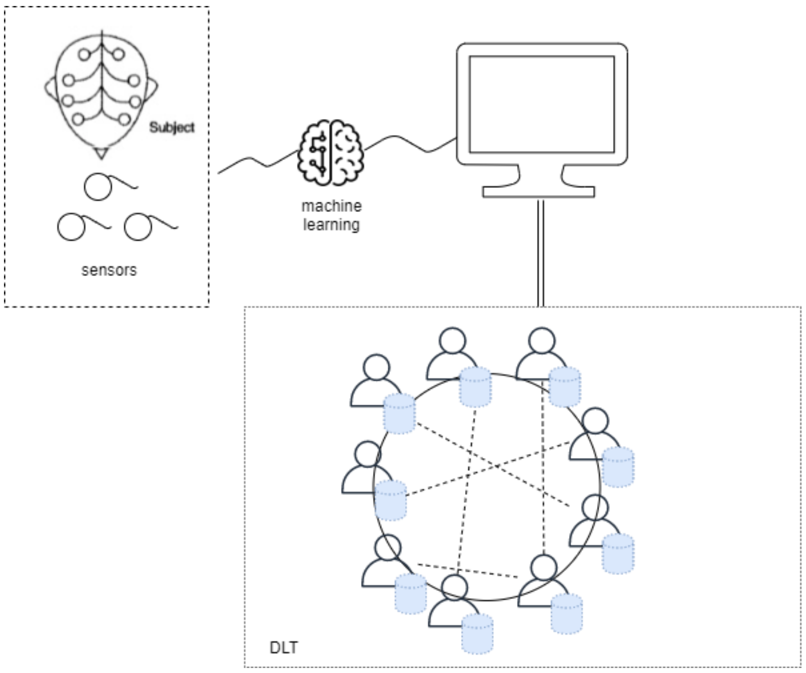


Figure 5.1: Overall conceptual view of the platform, outlining the interaction between the user and interface and also the decentralized system where the user is inserted (DLT).

with shared public data. Each user has the same influence over the network, there are no supernodes.

5.3 Architectural Design of the Solution

Knowing this application is subdivided by components, it is important to distinguish how these interact with each other, including the interaction between the user and the machine, the frontend and its relationship with the backend and how the latter interacts with the common ecosystem and the DHT. The diagram shown in Figure 5.2 is meant to showcase, with higher granularity, how each component interacts with each other.

Although what is shown in Figure 5.2 leans more towards public data sharing within the DHT and not specifically direct peer-to-peer communication, it is enough to outline a broad concept of the application and its inherent layers. From this picture, it is clear to outline four significant layers within the architecture: the bridge between the user and the frontend, which takes form of classification algorithms of machine learning; the frontend counterpart; the backend component with its inherent mechanisms; and the public DHT where the information will be shared for other users to retrieve and redundantly store within their own devices.

Regarding the interaction between the user and the frontend part of the application, this will be made through an EEG headset which noninvasively monitors electrical activity from the brain. By using the Emotive App coupled with a classification algorithm, thoughts from the user can be translated into commands to navigate through the application. Regarding

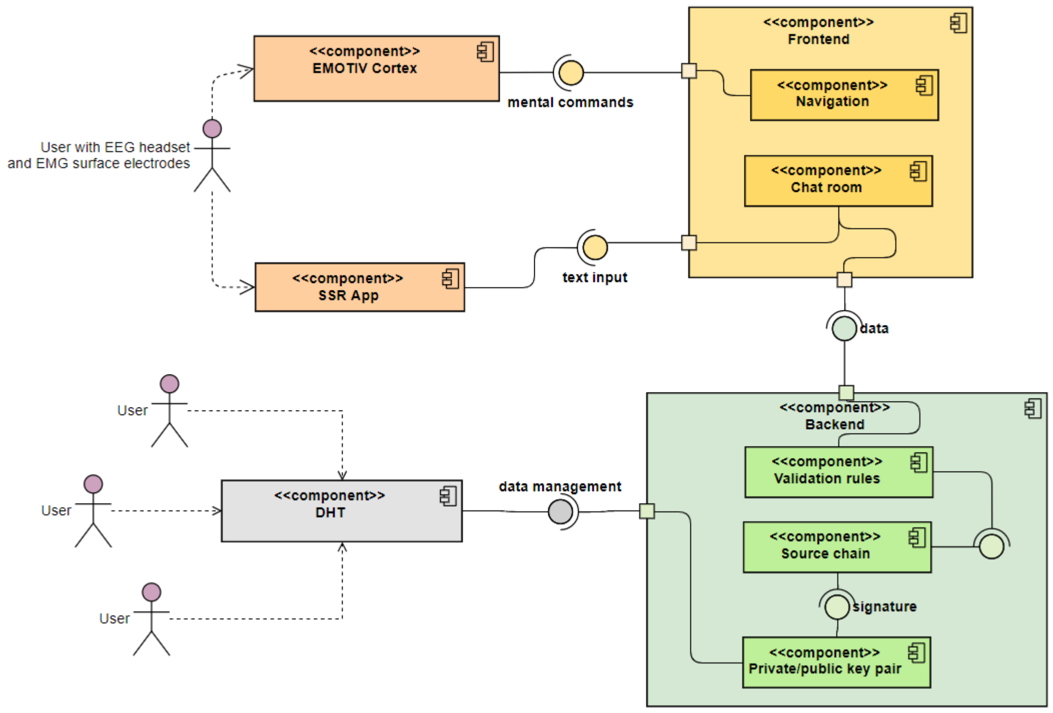


Figure 5.2: High-granularity diagram showcasing how each component/concept interacts with each other across the application. The frontend and the backend counterpart both reside within the user's device.

the SSI part, these will make use of a self-developed algorithm which will roughly translate muscle movements to text input that is to be sent through the network and to other users.

Focusing on the backend counterpart, the incoming data (which is the message) is checked in a set of validation rules. The validation rules are present in all peers and are the same throughout. This maintains data integrity within each peer and, directly, of the DHT. In this component, a user maintains its own data (shown as "Source Chain" in Figure 5.2) which was previously validated. Every piece of data of the user is present in his own source chain and whatever data that is meant to be public in it is promptly shared in the DHT. To claim ownership of each piece of data, a public-key cryptography method is employed which refers the data to the user that created it, all the while keeping it secure from other peers to see.

The DHT is, as the name implies, distributed across the peers in the network. Therefore, from the backend component to this one, there are regular data creation and retrieval operations and these have to be propagated to the rest of the peers to maintain data integrity across the network. Every node runs the same version of the backend and frontend component on their respective devices.

This was an overall perspective of the application with a clear emphasis on the backend counterpart. Many more aspects of the application regarding data consistency, security and user privacy will be further discussed in greater detail within the next chapters.

5.4 Anatomy of the Decentralized System

Considering that the application at hand is non-conventional and non-centralized, the previous architectural diagram shown in Figure 5.2 merely showcases a high granularity view of the application as a whole, hiding details that are important and distinctive from conventional software engineering applications. The purpose of this chapter is to showcase technical details and design choices that are meant to be carried out during the implementation phase.

5.4.1 Backend Stack

This paper focuses more on the decentralized backend rather than the interface between the user and the machine of the other counterpart. Considering that the Holochain technology and framework was chosen to be used in this project, the naming conventions of certain components within the application will be the same as the ones used by the creators, developer community and contributors.

Within the framework nomenclature, an application built with the Holochain framework is called Holochain Application (hApp), which refers to the whole solution, from front to back. In the case of this project, the hApp refers to the overall messaging application that can, in the future, have more features and lean more towards a social network panorama.

From a top-down architectural view, an hApp bundle (that is present in every node within the network) is composed of the frontend and the DNA. The latter is what defines the basic functionality and the rules in which every user will abide. It can be compared with a microservice. It is with the DNA that the frontend counterpart talks with, via Remote Procedure Call (RPC) [2] and calls the appropriate functions to interact with the data inherent to the user. Using the RPC protocol allows for this communication regardless of the technology or language the frontend is written in. Moreover, it is crucial to denote that each DNA in the hApp has its own dedicated and separate private network and, therefore, distributed data store (DHT).

Alongside the hApp bundle, there is another layer that makes up the node within the network. This layer, architecturally-wise, will mediate the communication between each user with the rest of the network. This layer will serve as the entry point for each peer and every transaction stemming from and going to the DHT will go through this layer, which acts as a controller. In Holochain networks, this layer is named as Conductor. The Conductor will expose APIs to make function calls into the components that make up DNA instances. In addition to this, the Conductor also allows bridging and communication between different DNAs of the project. These components are shown in Figure 5.3. Every node has a copy of the backend, the frontend and a conductor that will mediate processes in and out. Each DNA refers to its own DHT and network and it's made up of zomes which hold the logic of the application. So, for example, one DNA can be associated with communication between users and another DNA (with its own DHT) can refer to a Wikipedia-clone, for example.

As it was shown in Figure 5.3, DNAs are made up of pieces of application logic which, within Holochain, are called *zomes*. This is where the core logic is defined and contains functions to initialize the user's account. It is also where data is stored, validated and retrieved from.

[2] A form of caller-executor interaction and are a form of inter-process communication, in which different processes have different address spaces (or virtual address spaces). It is often used for communication between distinctive processes.

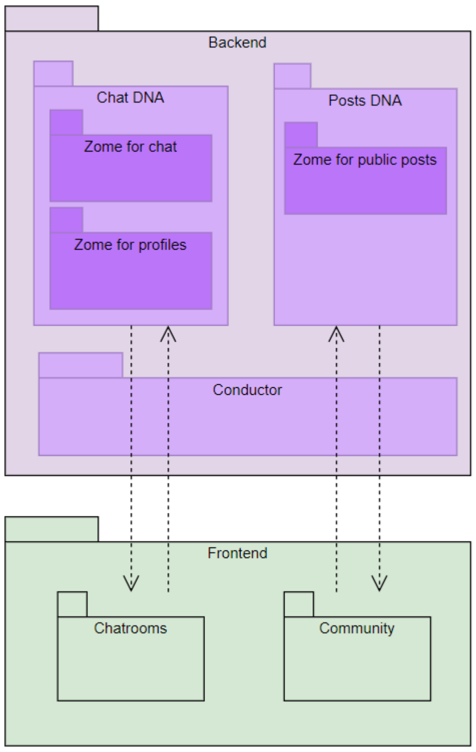


Figure 5.3: Coarse-grained diagram showing the packages of the stack that makes up the hApp bundle present in every node.

The functions of the zome are exposed as an API for the Conductor to call and yield back to another user that might want to invoke it. Within a single DNA, there can be various zomes and different pieces of logic. For this project in specific, two different paradigms were identified: one for direct communication, one-on-one users and for chat rooms that involve more than one user. These have different validation rules and the application logic behind it.

This is a more detailed view of what makes up the application. An example of peer-to-peer communication can easily be explained. Every user runs the same hApp bundle and, therefore, same versions of frontend and the backend. For the project, besides direct communication, a community-based feature can also be implemented. This is why in Figure 5.3 has two DNAs, one for communication between users and another for a Twitter-like network on public posting/blogging. These entail different DHTs and networks. However, bridging between these two different features is possible through the conductor that is present alongside both DNAs and associate the same user on data that are on both different DHTs. The frontend counterpart will fetch and post data through the conductor and consume API functions exposed by the zomes, which validate data and maintain the logic of the DNA intact. An example of validation could be, for example, strings can not be more than 20 characters. The advantages of having these components decoupled and separated allow for better code maintenance prospects but also code modularity, bug isolation, better unit, integration and stress testing and more manageable scalability.

Zome Detailed Look

The way the zomes are laid out within the application follow some concepts that ought to be cleared to grasp the design on a more fine-grained perspective. Since the programming language supported by Holochain is Rust, it is advisable to follow a Object-oriented Programming (OOP) style.

On a functional programming level, all of the components are stored within the application folder which is consisted of two folders: the DNA folder and the UI folder. The UI folder, as it was mentioned previously, represents the frontend part of the application and can be developed on any technology stack and the functions from the backend will be called through RPC.

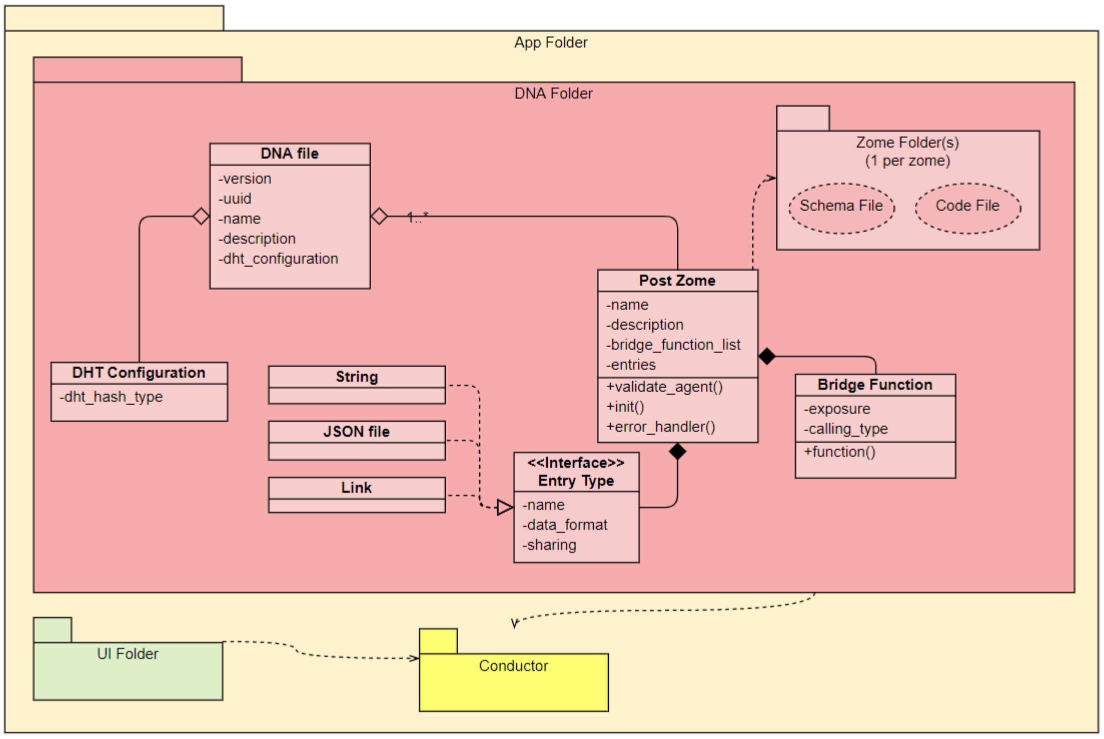


Figure 5.4: Fine-grained diagram showcasing the different conceptual components that make up the application definition, outlining the DNA and its composing zomes. Also, in the diagram it is slightly referred how each folder will be laid out, so as to follow Holochain's coding conventions.

In turn, the DNA folder will contain a DNA file and a folder for each zome that makes it up. This structure is recommended by Holochain because it allows for the hash digest of the entire content of the DNA folder to be validated when the user joins in the network, making it possible to validate that every node is running the same DNA and zomes. This validation process will only occur on the DNA folder. Continuing, each DNA will withhold its name, Universally Unique Identifier (UUID), description and configuration attributes regarding the DHT that is associated with the DNA present within the application. Additionally, each DNA will consist of one or more zomes.

Each zome has its own name, description, a list of entry types and functions that it has implemented. Each function implements the behaviour of the zome and all of these include its own name, a calling type (indicates whether the parameters are simple strings or more complex entry types) and an exposure parameter that defines if the function is only callable from within the zome, callable from other zomes within the same DNA or it is fully public. Additionally, each zome offers a specific data structure that, in this design, is named as an Entry Type, which can be either strings or JSON structures (can be found in Figure 5.4 as the Schema File).

As it is also shown in Figure 5.4, each zome class has an *init()* function that is called when the user first joins the network. This function is called on all zomes that make up the application and are to validate its integrity before joining the network.

All of the DNA and UI will go through the Conductor which will validate data going in and out of each node and maintain a private/public key pair cryptography in each communication with any other node.

5.4.2 Source Chain and DHT

It has been mentioned in Section 5.3 that each node within the network has their own chronological record of all the data the node has produced within the app, either private or public. This history of data is composed of entries and can be of any type, as it was shown in the previous chapter. In addition to this, users have their own identity within the network and it is made through cryptographic public/private key pairs, provided by the Holochain framework. With these we can associate pieces of data with users using the public key. Users prove authorship of data through digital signatures created with their private key. Any data tampering is detected by using the author's public key to verify the signature made with the private key. These validations and public/private key pairings are provided by Holochain's HDK API. The genesis entries of the user's personal record starts with the DNA's hash and the user's public key, to be compared with each entry within the record to validate and make it easy to detect any attempts at tempering information.

It is important to notice why this record is called a Source Chain. Every entry that is added to the Source Chain by the Conductor is chained with previous records and holds the hash of the previous header, as shown in Figure 5.5. This is what ensures the integrity of the whole record. If a third-party tries to change an entry, it will break the chain of records and will raise alerts for tampering and breach.

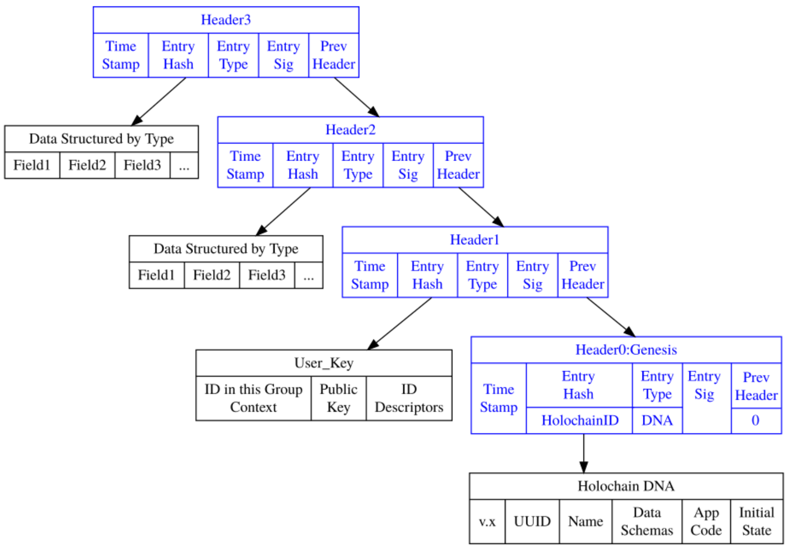


Figure 5.5: Closer look at each entry header within the Source Chain (source: (Turland 2018)).

Whenever a user wants to post something to the DHT, it first writes to its own personal data record and then decides if it wants to be made public or not. Every piece of data in the DHT is retrieved by its unique address. Nodes are randomly selected to validate and store an entry of the DHT based on how near they are to the entry's address. This is closely related with the partial consensus approach taken, where a random selection of nodes that are nearby validate the entry that goes in and out of the DHT. Holochain provides a special type of entry called Warrant, which carries evidence that a bad node is trying to forge or change the history of the DHT and is propagated throughout the network through Gossip protocol. This Warrant is produced by the node that found an entry to be invalid.

As it is showcased, validation is extremely important and is a multi-step process that guarantees data integrity throughout the network and it occurs both at zome-level, Source Chain-level and DHT-level.

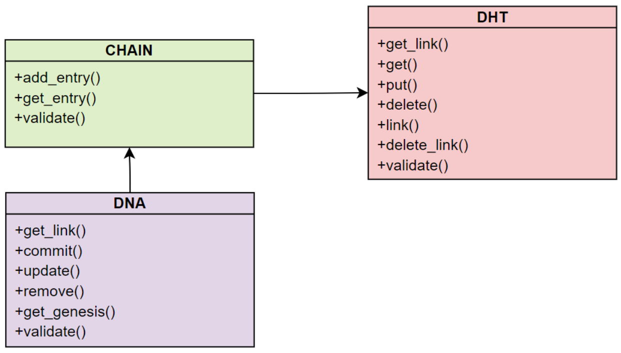


Figure 5.6: Topology of the decentralized backend encompassing the DNA of the application, Source Chain and the shared, distributed hash table.

Figure 5.6 showcases a good example of the overall topology of the whole decentralized application. Within the DNA is where one inputs data to later be stored in the DHT. This data is stored in the Source Chain which is connected into the DHT which assigns it a hash address by which others can access the data. Therefore, both the data and the chain stay local to the own peer, but the hash addresses are shared so other nodes can access the data without tampering it.

As it is shown in Figure 5.6, each component has a set of functions associated with it. In the DNA case, functions such as `get`, `update`, `remove` and `validate` are all data related. Adding and retrieving entries from the Source Chain are self-explanatory, as well as in the DHT. These functions are an example of those that can be found in the application for data retrieval and validation.

The main takeaway and overall idea from the program is that each user will want to add and validate data, append it to their own chain and update the DHT according to the validation rules, and then access that data via DHT that refers to a specific chain. This, however, poses the question: what happens when the user goes offline? For this, Holochain provides a mechanism to store multiple redundant copies of each entry so that the information is always readily available even when its author is offline. This resilience factor can be fine-tuned.

5.4.3 Deployment View

All network nodes maintain two persistent data stores for each DNA the node has: a local Source Chain which holds the records of data of the user and a shard of the shared DHT. Each node retrieves information from each other and essentially does communication through gossiping. This protocol is used to propagate validating data and to expel malicious nodes through warrants. Each node validates any creation, deletion or update of any data for the DHT. Additionally, the gossip channel is used to detect if the author of specific data entry is offline or not. Figure 5.7 showcases the deployment view of the application and these associations in a clearer manner.

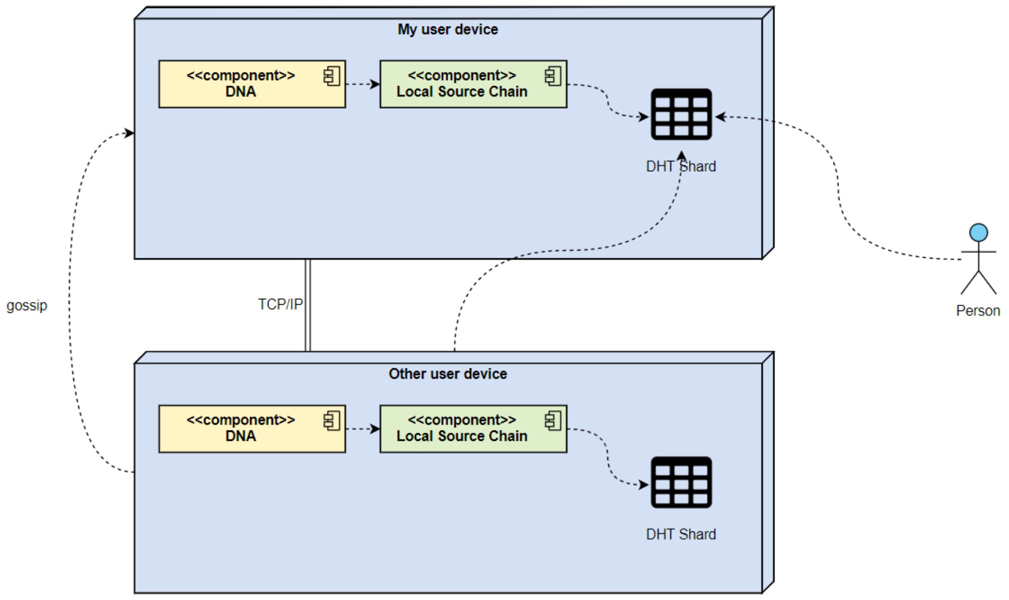


Figure 5.7: Deployment view of a single DNA deployed on two different nodes with different shards of the DHT. This diagram is more centered on the user on the top, as the latter also communicates and has associations with shards of DHT from other nodes.

5.4.4 Common Processes

It is worth demonstrating how specific processes and use cases showcase the interaction between the different components that make up the application so far. The following Figure 5.8 represents data creation generically, as specific use cases will follow the same pattern. Data retrieval will follow approximately the same process.

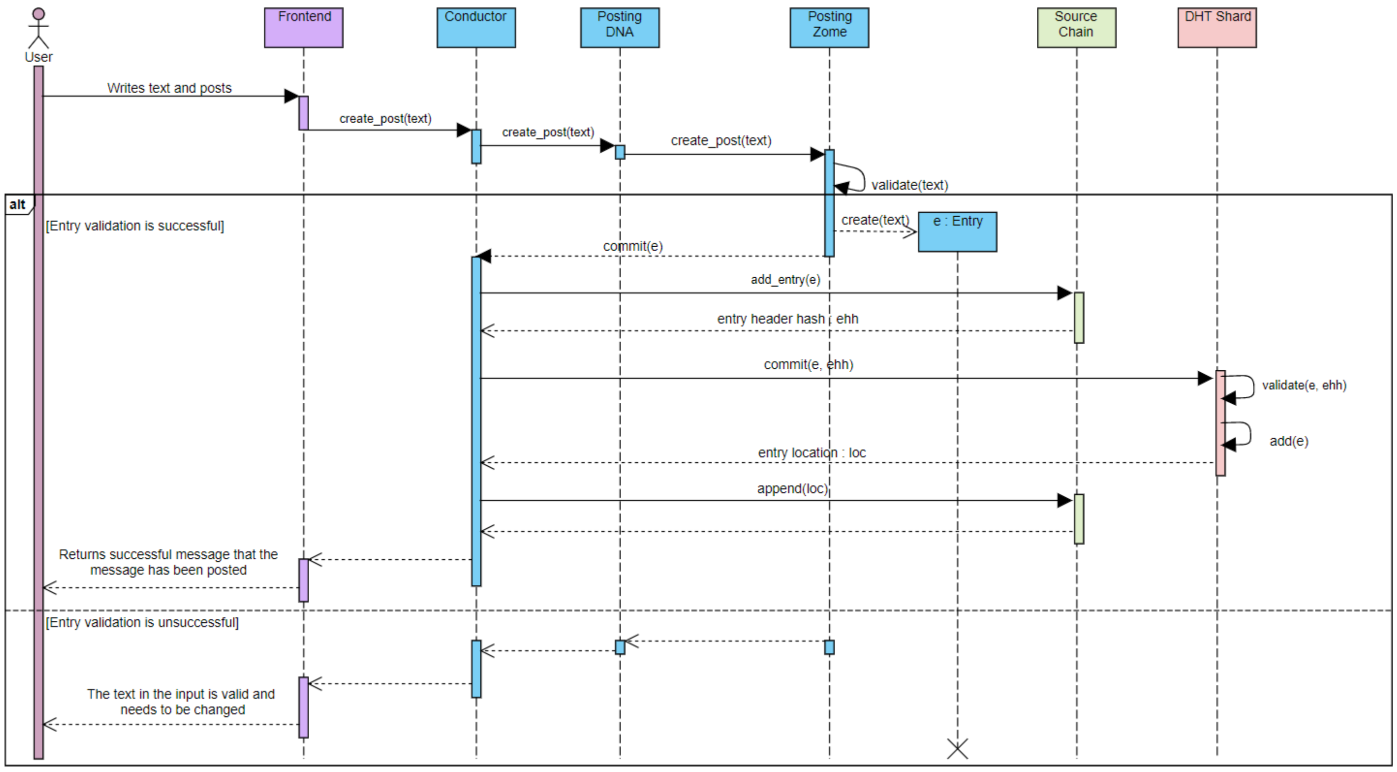


Figure 5.8: Sequence diagram outlining the process of a user posting public data in the DHT.

The validation rules instilled in each user's DNA is used on two scenarios: on data creation and authoring; and peer validation. For the latter, each user compares a new entry with their set of validation functions and if the entry is invalid, peers will issue warrants that will warn other users that the entry is invalid. Knowing who created the entry, users, through gossiping, will know the user did a malicious act and together will blacklist the user from the system.

However, the sequence diagram shown in Figure 5.8 depicts a scenario of data creation and how it involves the different components that make up the application. The user first starts by inserting text and ask to publish it from the frontend. Since the frontend is bundled together with the backend, it invokes the Conductor's function via RPC which, in turn, calls the DNA of the application that checks with the validation function present within its zomes, which holds all the rules and logic of the application. If the validation function yields a negative result, the user is prompted with a message that the data inserted is invalid and to repeat the action.

On another hand, if the data is valid, a new Entry object is created by the zome and asks the Conductor to commit it to the Source Chain and the DHT. The Conductor attempts to add the entry to the Source Chain which, in turn, returns the header hash to be added to the DHT to claim authorship of the user who created the data. After this, the Entry is added to the DHT and, if the validation is successful, it is added to the hash table shard and the latter returns the address location within the table to be appended to the Entry within the Source Chain of the user.

Outside of the scope of this diagram, the data is later then reviewed and replicated by other peers to add redundancy in case the user goes offline and still have data accessible from other users.

5.5 Design Alternatives

In this section, some design alternatives are outlined when compared to the current design of the application some of the choices are justified accordingly.

5.5.1 Dedicated DHT for User's Address Discovery

Besides the approaches alternatives that were outlined and explained in Section 4.1.2, there are a few aspects to the current design depicted throughout the last chapters that could have been different. One of these aspects is regards having a dedicated DHT for agent/user discovery. This aspect is a sensible option for a faster connection between peers and mapping out agents with their respective addresses and, inclusively, creating a peer-to-peer channel for direct communication. It is, in fact, the same approach used by Jami, which has each IP address of the user stored in a DHT for easier node discovery. However, for the scope of this project, which relies heavily on node anonymity and data disassociation with a specific person, this solution is not viable, since tracking the IP address is possible since the DHT is public.

However, if a dedicated DHT that would hold the node's addresses, there would be ways to eliminate the risk of tracking people through their addresses. One of these ways would be to generate a temporary public/private key pair to make connections to peers found in the DHT. Onion routing could be used to make it more difficult to associate a pair of users together.

Utilizing a single DHT for all purposes is enough in terms of performance and retrieval efficiency because of its runtime complexity. Additionally, every piece of data within the blockchain has the hash of its author in its header, making it easy to associate it with a user if provided with the correspondent public key. Hash tables are $\mathcal{O}(n)$ worst-case time complexity, with $\mathcal{O}(1)$ average and amortized case complexity. Hash tables suffer from $\mathcal{O}(n)$ worst time complexity duo to either element having hashed into the same key (which is highly unlikely) and because of its load balance, having to re-hash and create a new table and re-insert each element to the table (Cormen et al. 2009).

5.5.2 Different DHT for Different Purposes

One constraint that might arise when developing and designing the application can be related with the long-term storage limit that each user device might have. Splitting business concepts with their own DHT can be a sensible option to tackle decoupling and allows the user to join which DHT they might want to retrieve data from. However, this would entail a mechanism of communication/bridging between DHTs. Additionally, if different business concepts are correlated with each other, problems of synchronization between DHT is an issue that can break the user experience and data residing wherein. Even though bridging between DHT may be possible, the impact on performance would outweigh the benefits because as more users would join the network, the more processing power would be needed to maintain the data between DHTs synchronized with each other.

Chapter 6

Implementation

The following chapter delves into the implementation details of the aforementioned design use cases and how the use cases are achieved on a more technical level. The overall project structure, architecture, technology stack and implementation patterns will be explained throughout this chapter. Additionally, some concepts regarding the structure and design choices will also be introduced here.

6.1 Technology Stack

As it occurs in most pieces of software, each application has a specific set of technologies and frameworks that make it whole, and this is no exception. The current platform withstands three different layers: the networking layer, the data/logic layer and the frontend layer, as shown in Figure 6.1.

The networking layer pertains to the backbone of the DHT and the Content-addressable Storage (CAS). These two are what makes it possible to implement a fully decentralized system based on a distributed hash table. CAS provides a way to store information that later can be retrieved based on its content. This mechanism essentially allows the developer to retrieve entries within the DHT based on its content and also its address within it.

In fact, Holochain provides much of the boilerplate backbone of the DHT and how it is distributed along the agents within the network. All in all, this framework handles the networking protocols and how the DHT is randomly distributed amongst the peers, allowing the developer to only focus on the management of entries and users that enter and exit the network.

Regarding the data/logic layer, since Holochain provides a Software Development Kit (SDK) on Rust, which is handy because this language provides low-level control. Therefore, the management of data and business logic is implemented in this language. This layer, alongside Holochain, is wrapped and compiled in WebAssembly. When outputting the `.wasm` files, a specific Holochain interpreter is used and will create the application's DNA and expose its the functions written originally in Rust to the Javascript-based frontend. As long as the frontend is Javascript-based, any framework can make use of the methods made available by the backend counterpart.

When it comes to the frontend, specifically for this platform, the connection between the frontend and the backend is mediated by the Apollo GraphQL middleware, which is able to translate calls from the backend to the frontend in a more readable, manipulative and understandable way, allowing to read and mutate data within the DHT easily. Furthermore, the main frontend framework used is React, which has the potential to be ported to React Native, which in turn allows the development for the mobile platform.

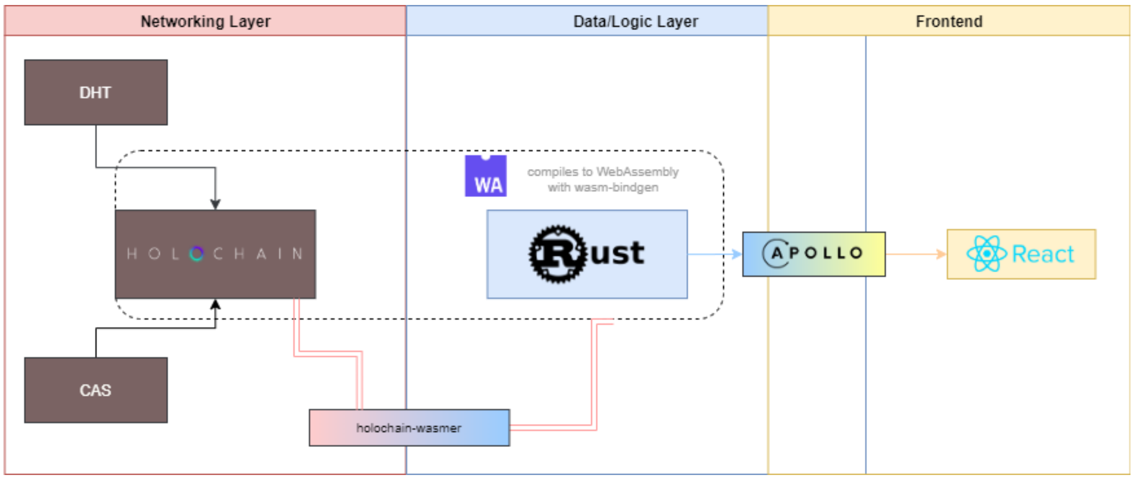


Figure 6.1: The different technologies and frameworks inherent to this application. Do note that the `holochain-wasmer` is used to interpret the WebAssembly (WASM) binary produced and create, amongst other files, the `dnajson` file.

6.1.1 File Structure

As it was previously mentioned, both the backend and frontend counterparts of the application are bundled together. When developing Holochain-based platforms and applications, the frontend is responsible for some of the logic that occurs within the platform, hence why the application is developed together within the same project. That is, the backend and the frontend are not truly decoupled, as it happens with most conventional projects.

Typically, the structure of the project is seldom shown because it is an irrelevant detail for the implementation. However, this project thrives on the connection between the frontend and the backend and how these relate with each other and, ultimately, make up the whole application. Since this bundle will be present in every user of the system, both the frontend and the backend need to be on the same version and make the appropriate calls for each one.

As shown in Figure 6.2, each user has a bundle that comprises of the `dna-src` (strictly referring to the backend and logic of the application and whole system) and `ui-src`, which refers to the frontend counterpart.

6.1.2 WebAssembly

With this project, one of the features wanted is the expectation to run with a near-native performance on any platform. For this to be possible, the compilation process of the project will go through WASM.

"WebAssembly is a standard being developed by the W3C group for an efficient, lightweight instruction set. This means we can compile different types of programming languages ranging

```
meletea
├─ conductor-config.toml
├─ dist
│  └─ dna-src.dnajson
├─ dna-src
│  ├─ config.nix
│  ├─ test
│  └─ zomes
│     └─ chat
│        └─ src
│           ├─ config.hcbuild
│           └─ Cargo.toml
├─ package.json
├─ Procfile
└─ ui-src
   ├─ env.env
   ├─ connect-to-conductor.js
   ├─ package.json
   └─ src
```

Figure 6.2: Structure of the overall project, including both the backend and frontend. For size reasons, some files are excluded, being the most relevant ones been narrowed down to these.

from C/C++, Go, Rust, and more into a single standard. WASM is memory-safe, platform independent, and maps well to all types of CPU architectures efficiently" (Haas et al. 2017). Even though it was initially designed for use by common browsers such as Chrome or Firefox, WASM grew to be taken up as a portable target for execution on native platforms as well.

WebAssembly is particularly useful in blockchain/decentralized applications. For example, Ethereum relies on EVM to run. Instead of relying on it, with WebAssembly, Ethereum can now run an efficient instruction set that can compile to a myriad of languages and be fairly confident that it will be performant, deterministic and executable across different types of platforms, which is ideal in decentralized systems - something that this project is also striving for.

In the case of our application, the set of instructions that compile the project and convert it to the `dnajson` file (the DNA file that will be run in every user and guarantees that each user has the same version of the application running) is located in the `config.hcbuild` file.

This file consists of the sequential steps every time the application is compiled.

```
{
  "steps": [
    ...
    {
      "command": "CARGO_TARGET_DIR=${CARGO_TARGET_DIR:-/tmp/meletea/
    target} && cargo",
      "arguments": [
        "build",
        "--release",
        "--target=wasm32-unknown-unknown",
        "--target-dir=$CARGO_TARGET_DIR"
      ]
    },
    {
      "command": "CARGO_TARGET_DIR=${CARGO_TARGET_DIR:-/tmp/meletea/
    target} && wasm-gc",
      "arguments": ["$CARGO_TARGET_DIR/wasm32-unknown-unknown/release/
    chat.wasm"]
    },
    {
      "command": "CARGO_TARGET_DIR=${CARGO_TARGET_DIR:-/tmp/meletea/
    target} && wasm-opt",
      "arguments": [
        ...
      ]
    },
    ...
    {
      "command": "CARGO_TARGET_DIR=${CARGO_TARGET_DIR:-/tmp/meletea/
    target} && wat2wasm",
      "arguments": [
        ...
      ]
    }
  ],
  ...
}
```

Listing 6.1: Snippet of the config.hcbuild file that is called after the application compiles. The program compiles and outputs a .wasm file that is then feeded to an Holochain-specific interpreter that converts it to a DNA file.

These steps allow the project to be converted into a `.wasm` file that is ultimately going to serve as the input for a Holochain WASM interpreter which will be tasked on converting these into a `dnajson` file and serve the `.wasm` files for the UI, together making up the whole application. Since the language of choice being used in this platform is Rust, the library `wasm-bindgen` was used to facilitate high-level interactions between WASM modules and JavaScript (the main programming language used for the UI counterpart).

To highlight some of the steps that are showcased in Listing 6.1, the `wasm32-unknown-unknown` command refers to the target wanted. In layman's terms, it is pretended to "Compile on almost any machine and run on almost any machine", all of this on an address space of 32-bits (`wasm32`). This was chosen instead of 64-bits because of the accessibility and availability that the 32-bit address space showcases in most current devices, including computers and mobile phones.

The sequence of steps shown in the picture is not arbitrary. The first two commands refer to optimizing the `.wasm` binaries and are essential to maintain the highest performance and a low sized file. The `wasm-opt` tool runs transformation and optimization passes on the file. Running it on `.wasm` binaries produced by the LLVM compiler (the same one that Rust uses), it will create binaries that are both smaller and execute smaller.

The step pertaining to the `wasm-opt` tool is simply to garbage collect the WebAssembly module and remove all unneeded exports, imports and functions. After this, the file is converted from a `.wat` file [1] to the wanted `.wasm` file to later be used on the Holochain interpreter.

6.1.3 Nix and Flow of Execution

Holochain provides a NixOS toolkit that allows for development and deployment of applications based on their framework. This is an useful tool that allows to run applications based on Holochain quickly. After using the `nix-shell` command, the shell will download the dependencies and create a shell session with all the environment variables and applications properly installed.

However, to have access to this toolkit and have a good development/deployment environment, some files are needed to setup and access the environment the files `config.nix` and `default.nix` are used to pin-point the version of Holochain and SDK of the shell environment.

The flow of execution to when an agent joins the network or starts it off begins in the `Procfile` which, in turn, details the processes that need to be spawned for the agent to partake in the network, start the conductor and connect it to the frontend. This file is run with the NodeForeman tool.

```
sim2h: RUST_LOG=debug sim2h_server -p 9000 -s 20 2>&1 | grep -v METRIC
holochain: yarn run hc:start
react: yarn ui:connect-to-conductor
```

Listing 6.2: Procfile that spawns parallel processes and connects the conductor to the frontend and streamlining the execution process.

As shown in Listing 6.2, the user, upon initialization, spawns three processes. The first one pertains to the `sim2h server` which is a multiplexer for nodes to connect with each other - a switch-board networking service. It is mainly used for testing and debugging purposes.

The `holochain` process runs a `yarn`[2] command that creates and executes the conductor through a configuration file (found in `conductor-config.toml`). This file allows to define various configurations, including the agent's public address, DNA instance and type of storage, as well as opening a port where the frontend will connect with to connect through a web-socket, outlined in Listing 6.3.

[1]WebAssembly is comprised of two formats that represent the same structures, albeit in different ways: the `.wat` text format and the `.wasm` binary format, which is lower-level and intended for consumption.

[2]Yarn is a package and dependency manager, mainly used for Javascript-based projects, similar to `npm`.

```
...

[[agents]]
id = '527baebc-c7c2-4789-ba45-1e9b2a9330c2'
name = 'alice::meletea::527baebc-c7c2-4789-ba45-1e9b2a9330c2'
public_address = '
    HcScJK4nsHzso5jd3nsRtWqSN4Gubon7c566a8e6kf4e77seji5s73NM7zSOwpi'
test_agent = true

...

[instances.storage]
path = '.persistence_directory/527baebc-c7c2-4789-ba45-1e9b2a9330c2'
type = 'file'

...

[interfaces.driver]
port = 33000
type = 'websocket'

...
```

Listing 6.3: Configuration file used by the conductor.

Similarly to the previous one, the `react` process runs a `yarn` command that, in turn, executes the `connect-to-conductor.js` file, tasked with connecting the frontend with the conductor. As shown in Listing 6.4, the frontend process waits until the conductor is successfully spawned and configured until it connects to the open port. In the case of the snippet of code shown, the port is 33000. Once the conductor starts, a `yarn` command is executed which effectively runs the React-based frontend and serves the UI files. The port in which the end-user can access the frontend of the application is detailed in the `.env` file (shown Figure 6.2).

```
let conductorBegan = false
const attemptConnection = () => {
  client.connect(
    { port },
    () => {
      client.end()
      if (!conductorBegan) {
        conductorBegan = true
        childProcess.exec('yarn start:live')
      }
    }
  )
}

attemptConnection()

client.on('error', () => {
  console.log('Waiting for conductor to configure and spawn')
  setTimeout(attemptConnection, 3000)
})
```

Listing 6.4: Snippet of Javascript file that is tasked with connecting with the conductor process.

6.2 Application Architecture

Mentioned in Chapter 5.4.1, to follow Holochain's conventions, the project ought to follow a specific structure, divided by zomes. For this project, there is only a single DNA (because we only want a single DHT for all users) and there are two different zomes within this: one namely called `chat`, which pertains to the chatting aspect of decentralized peer-to-peer messaging; and another one called `profile`, which is a simple zome that just manages profiles of each user on the public DHT.

Throughout there will not be a distinction between zomes because the existence of two is simply for organizational sake and does not have direct implications on the application on runtime. Within both of these, though, there are a myriad of components that make up the business logic and help fulfill the use cases. However, before delving into each component, it is useful to understand the entry-based architecture that was withstood.

6.2.1 Entries and Links

Holochain, as a tool, deals with the networking part of the platform and allows the developers to insert entries within it, as well as allowing the user to have private entries on their own device. Every entry has an header associated with it (automatically generated by Holochain) that has information about the user that created it and the corresponding public address. The content and the type of entry must be defined beforehand though.

It is possible to structure data within the DHT by having different types of entries. To create semantic link between entries within the DHT, an entry type named Link is created, associating a directional link between a base entry to a target entry and a string associated with the link type. This created entry type will be particularly useful on the Anchor pattern explained further on this document (refer to Chapter 6.3.1). Therefore, much of the architecture followed is to help structure the DHT to increase lookup times and instill a sense or organization within it.

6.2.2 Structure

The code structure and entry definitions are implemented in order to maximize decoupling between components, although they can be related and linked with each other. Considering we are using Rust, each component is namely called as a *module* and in most cases corresponds to an entry type. Therefore, from now on, these will be used interchangeably unless explicitly stated otherwise.

Each module has a set of handlers and a set of validators that are in charge of handling operations regarding the module and validating its integrity when committing to the DHT or updating an entry.

As shown in Listing 6.5, every component/entry type uses the `entry!` macro to define the entry type for the DHT table to understand. With this macro, the developer defines the name of the entry type, a descriptive string and visilibity - whether it is Private (only saved on the user's device) or Public (saved on the user's device and the DHT).

```
pub mod validators;

pub fn definition() -> ValidatingEntryType {
    entry!(
        name: "message",
        description: "lorem ipsum",
        sharing: Sharing::Private,
        validation: | validation_data: hdk::EntryValidationData<
    MessageSpec>| {
            match validation_data {
                // ...
                hdk::EntryValidationData::Modify{new_entry, old_entry,
    old_entry_header, validation_data} =>
                {
                    validators::validate_entry_modify(new_entry,
    old_entry, old_entry_header, validation_data)
                },
                // ...
            }
        },
        links: [
            from!("user_id_anchor",
                link_type: "user_id_anchor->message",
                // ...
                validation: |validation_data: hdk::LinkValidationData| {
     match validation_data {
                        hdk::LinkValidationData::LinkAdd{link,
    validation_data} =>
                        {
                            validators::validate_link_add(link,
    validation_data)
                        },
                        hdk::LinkValidationData::LinkRemove{link,
    validation_data} =>
                        {
                            validators::validate_link_remove(link,
    validation_data)
                        }
                    }
                }
            )
        ]
    )
}
```

Listing 6.5: Module entry file of a normal component (in this case the Message component/entry type). Only the modify validator call is shown here

In addition to this, it is possible to assert *a priori* validations of entries when creating, updating or removing these in the DHT. These validators can be as simple as limiting a string size, for example. In this snippet of code, and using the `entry!`, the developer can define an array of links with the `from!` and `to!` to define the previously mentioned semantic relations between different entries and entry types. Similarly, one can validate the creation or removal of these links, as shown in Listing 6.5.

As it was mentioned, validators are present in each component, allowing to better implement business logic. The piece of code in Listing 6.6 showcases the implementation of how we validate the authorship of each entry and compare it against the person trying to modify it. By verifying the header provenances of the entry, we can bar a random agent modifying the entry except for the very owner and creator of the entry being modified.

```
pub fn validate_entry_modify(new_entry: MessageSpec, old_entry:
    MessageSpec, old_entry_header: ChainHeader, validation_data: hdk::
    ValidationData) -> Result<(), String> {

     if let (Some(o), Some(p)) = (old_entry_header.provenances().get(0),
    validation_data.package.chain_header.provenances().get(0)) {
         if p.source() == o.source() {
             Ok(())
         } else {
             Err("You did not create this entry so you can't update it.".
    to_string())
         }
     } else {
         Err("Validation_data has no provenances.".to_string())
     }
}
```

Listing 6.6: Validator function snippet upon entry modification. This example checks the authorship of the entry to make sure only the author can modify it.

6.2.3 Components

The organization of the DHT and, therefore, the business rules are translated in components/entry types that occupy different roles within the DHT. Each of these components usually directly correlate with the objectives and use cases showcased in Section 5.1.1. The following sections provide an insight in every entry type included in the application and how the frontend ultimately interacts with these. It is relevant to notice that, as it was aforementioned, each component/entry type has a module definition, a set of handlers and validators. Therefore, each of following components are self-containable.

As it is outlined in Table 6.1, each component has a specific set of fields and a sharing status property. Some entry types have additional metadata that is required to differentiate between entries and that are relevant for the consumer of the backend to have to perform, for example, sort operations. The need for having this metadata is further explained in Section 6.4. This type of metadata is only returned to the consumer (frontend) and, in turn, this counterpart creates entries of a given type through a specification structure, referring strictly to the content of the entry and not the metadata.

Message

The message component is indeed the most important and relevant component within the application since the latter revolves around messaging with other individuals privately. This message component is mainly used for direct communication (i.e. when two users are online and communicate with each other). This entry type is private and it is solely used to record the history of messages the user has sent or received from other users within the system.

Table 6.1: Shortened description of each component, sharing status, fields and spec structs existence.

Components	Fields	Sharing status	Has spec struct
Message	From To Content	Private	Yes
Async Message	From To Content	Public	Yes
Friend	Agent Public Key	Private	No
Profile	Agent Username Avatar URL	Public	Yes
Public Key	Agent Key	Public	No
Secret Key	Key	Private	No

Each message must have the address of the sender and the recipient, as well as the content sent. Having the sender and the recipient respectively will allow to better lookup the messages and create anchor trees based on each user (information about anchor trees can be found in Section 6.3.1.

Async Message

The Async Message is also important and it is only used for communications where the recipient is not online. All in all, agents want to communicate privately with each other even one party is offline but since node-to-node message requires both the sender and recipient to be online, this becomes impossible. To fix this issue, the private messages are encrypted with the recipient's public key and they are published to the DHT. This how this entry is used and it explains the similarities with the Message entry type, except it is public and, therefore, posted in the DHT.

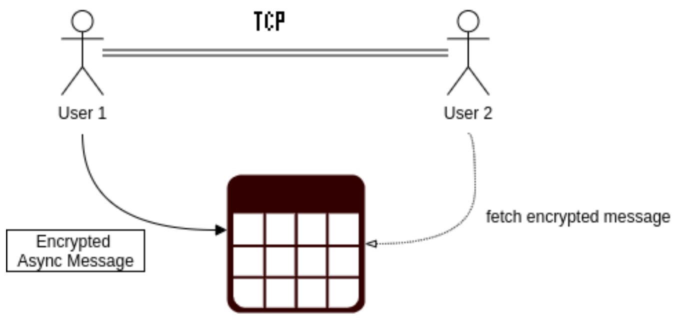


Figure 6.3: Diagram showcasing the usefulness of the Async Message message type.

As shown in Figure 6.3, the Async Message is used whenever direct connection through websockets between two agents is not possible (i.e. one is offline). Therefore, the wanted message is posted on the DHT and linked to later be retrieved by the recipient whenever the latter returns online.

Friend

The Friend entry type is supposed to have the same role as a contact of a person. In fact, the only way to message someone within the network is to first add the specific address as a friend/contact. This is to prevent spamming or unsolicited messages from unwanted contacts. This entry type is private and has the public key of the agent associated since, to send asynchronous messages, this key is important to sign and encrypt the payload to make the transaction and communication as secure as possible.

Profile

Each user as a Profile associated, with a username and an avatar URL to be displayed on the frontend counterpart. Every user has a profile associated whenever they join the network and it is useful to associate nicknames and visualize other users besides their address within the system.

Public and Secret Key

Each user when they join the network, are mandatory to create a profile and have a pair of public and secret keys associated with these. There are two entries to relate with the latter two: Public Key and Secret Key. They are public and private, respectively. Every time a user wants to communicate with another agent, this key pair is important to allow security and privacy on this process. Other uses can access the public key as they wish, since it resides in the DHT and is associated with the respective user. However, the secret key resides only on the user's device and is not meant to be shared with anyone else.

6.3 Organization

The organization of the application and how the components align with each other is crucial to maintain the flow of the business logic and increase performance of operations within the DHT. The section that follows will delve into the patterns and organization of the entry types/components and how these are laid out within the system.

6.3.1 Anchors

As it was aforementioned, it is possible to define entry types within the DHT. Technically, entry types refer strictly to records with some metadata attached to it which help facilitate queries and discovery. However discovering data in the DHT can quickly become hard and computationally intensive because data can rapidly spread out over the many nodes and can not be queried efficiently.

To solve this issue, a special entry type is created, namely called *anchor*, thus embodying easily discoverable starting points that act as bases for links to other data. Therefore, the entire DHT effectively becomes a graph database. These anchors can be created both at a private (user's own device) and public (DHT) level.

It is important, however, to acknowledge the anti-pattern potential this concept might have. If a given anchor is expected to have a considerable amount of links on it, this could create a hot spot in the DHT neighbourhood that stores the anchor. Holochain will interpret the nodes that are in the neighbourhood will be responsible for storing all the links and responding to any possible requests for them. To address this issue, creating sparse anchors and aggregate roots instead of a single one greatly increases query times and is not costly on write time.

This pattern is directly correlated with two structures introduced in the application to boost performance and lookup speed. An anchor can either be a value that is hard-coded into the DNA itself and is immutable (referring to Symbolic Anchors) or a tree of values that link to their parents, allowing agents to traverse the tree starting by the root node (named as Anchor Trees). Both of these are within the application and are explained below. Together partake in the structure of the components and anchors which are explained in Section 6.3.2.

Symbol Anchors

As it was stated before, symbolic anchors refer to hard-coded anchors. The application benefits from global registries of different types, such as profiles. But to query and entire DHT for this type of data can quickly become cost-inefficient, since the data is spread across many nodes. For this, this special anchor entry type is created, where only one entry of this type of anchors can exist within the DHT. To enforce this rule, the content of these anchors are empty - this is because entry hashes are based on the entry type and the content (per Holochain's documentation). Since the DHT stores data by its hash, no matter how many agents create and commit these anchor entries, only one will exist on the DHT.

Furthermore, for each anchor of this type, a link type connecting it to an entry type is also defined. Once the symbolic anchor is created, users within the network can start creating entries (for example, their own profiles), publish them and link these to the anchor. Therefore, when someone wants to query all entries of a certain type, they simply query the symbolic anchor and fetch all of its links.

A few considerations with this pattern should be noted, though. This type of anchor only makes sense in cases where the application has already a pre-defined business logic and knows about the points of globally queryable data in advance. In cases where the ontology is undefined and is dynamic, the pattern Anchor Tree is more fitting.

Anchor Trees

There are cases when agents want to find data but, from the get-go, there is no way to know what sort of anchor taxonomy one needs to create at application design time, making Symbol Anchor's usage inadequate. For this, an Anchor Tree is useful to create anchors and organize entries at runtime. This presupposes an entry type that holds all the anchors (the root anchor), a link type that links each anchor with each other and a link type that links an anchor to the entry type people are ultimately interested in. This way, users can easily find interesting information by traversing the tree, starting at its root node.

Users need to find data in a DHT (see Anchor 4), but in some situations you don't know what sort of anchor taxonomy you need to create at application design time so Symbol Anchor 2 can't be used. Examples include knowledge bases, category hierarchies, store departments, company directories, and address books that allow users to create custom fields.

Even though traversing the tree boosts performance, it is important to note the de-duplication behavior stemming from the DHT: anchors that have the same type and label do not exist and are de-duplicated. Since there might be a situation of duplication and parent/child links being smash together, this can easily be solvable by having the anchor's string tuple to be a value and its parents hash/random hash. This way, we are guaranteed to have the anchor be created.

```
pub fn anchor_entry(type_string: String, text_string: String) ->
    ZomeApiResult<Address> {
    let entry = Anchor::new(type_string.clone(), Some(text_string.clone
    ())).entry();
    if let Ok(None) = hdk::get_entry(&entry.address()) {
        hdk::commit_entry(&entry)?;
        hdk::link_entries(
            &check_parent(anchor_type)?,    // anchor type address
            &entry.address(),
            TYPE_OF_ANCHOR_LINK,
            &text_string,
        )?;
    }
    Ok(entry.address())
}
...
fn parent_check(type_string: String) -> ZomeApiResult<Address> {
    let type_entry = Anchor::new(type_string.clone(), None).entry();
    let type_entry_address = type_entry.address()

    if let Ok(None) = hdk::get_entry(&type_entry_address) {
        hdk::commit_entry(&type_entry)?;
    }
    root_check(type_entry_address.clone(), type_string)?;
    Ok(type_entry_address)
}
...
```

Listing 6.7: Snippet of code showing function to create an anchor type and check parent of a specific node. The first function creates an anchor with a specific type and if it already exists it will use the existing anchor. If it does not it will commit and check if the anchor type and root need to be created afterwards. The second function checks the parent node and checks if it exists. Create a new one if it does not.

To implement this, a specific module that allows the generic implementation of anchor trees was created, even though there is only one instance of an anchor tree in our application. It is possible to create anchors of a specific type and traverse the nodes and check the parent or the root node. Obtaining the root node's address and its children is a good way to improve performance and fetch the links straight from this address instead of going through the transversing the tree in its entirety. Some of these functions are shown in Listing 6.7.

A few precautions ought to be undertaken by using this anchor tree concept. One problem that may arise relates to ensuring referential integrity when it comes to links between anchors across the tree. A child always links explicitly to its parent, but the only relation a parent has with its children is via link. Therefore, if an instance crashes after creating a child before linking from the parent to the child, the link integrity is broken. To fix this problem, the agents verifies their anchors tree referential integrity on initialization and every time it logs on.

```
let level_1_anchors = hdk::query("user_id_anchor".into(), 0, 0)?;

for level_1_anchor in level_1_anchors.iter() {
    if hdk::api::get_links(
        &private_dm_anchor_address,
         LinkMatch::Exactly("private_dm_anchor->user_id_anchor"),
         LinkMatch::Any,
    ).iter().isEmpty() {
         hdk::link_entries(&private_dm_anchor_address, &level_1_anchor.
    address(), "private_dm_directory->user_id_anchor", "")?;
    }
}
```

Listing 6.8: Snippet of Listing that validates the integrity between the root anchor and the children for messages of an user. "private dm anchor" corresponds to the root anchor and "user id anchor" refers its children.

As shown in Listing 6.8, it showcases the integrity scan on the first level of an anchor tree on the user's device (the organization of anchors and anchor trees are better explained in the next section - Section 6.3.2), where we iterate through the children and check if there is any missing link between each one and the root. If a link is missing, it is created accordingly.

6.3.2 Components and Anchors Structure

The aforementioned anchors and entry types are structured both in the user's source chain (i.e. device) and DHT. This organization is depicted in Figure 6.4.

The user's device/source chain makes use of the anchor and anchor tree patterns. The latter refers to the hierarchy of the user's private communication endeavours. The root anchor pertains to the `private dm directory`, and its children refer to the agent's address he/she is talking to. We are guaranteed to always have different children because each `user id anchor` has the agent's public address as content and since these are always unique within the system, there is assurance on the uniqueness of the anchors as well. Each of these anchors, there are multiple Message entry types connected, referring to the messages sent and received with that specific user.

In addition to this, there is an `friends` anchor that connects to each Friend entry type which has the agent's public key in its content. This is important for sending and receiving messages since their encryption is based on it.

Finally, there is a single instance of a secret key that is created upon initialisation on the first time when the agent is created and joins the network.

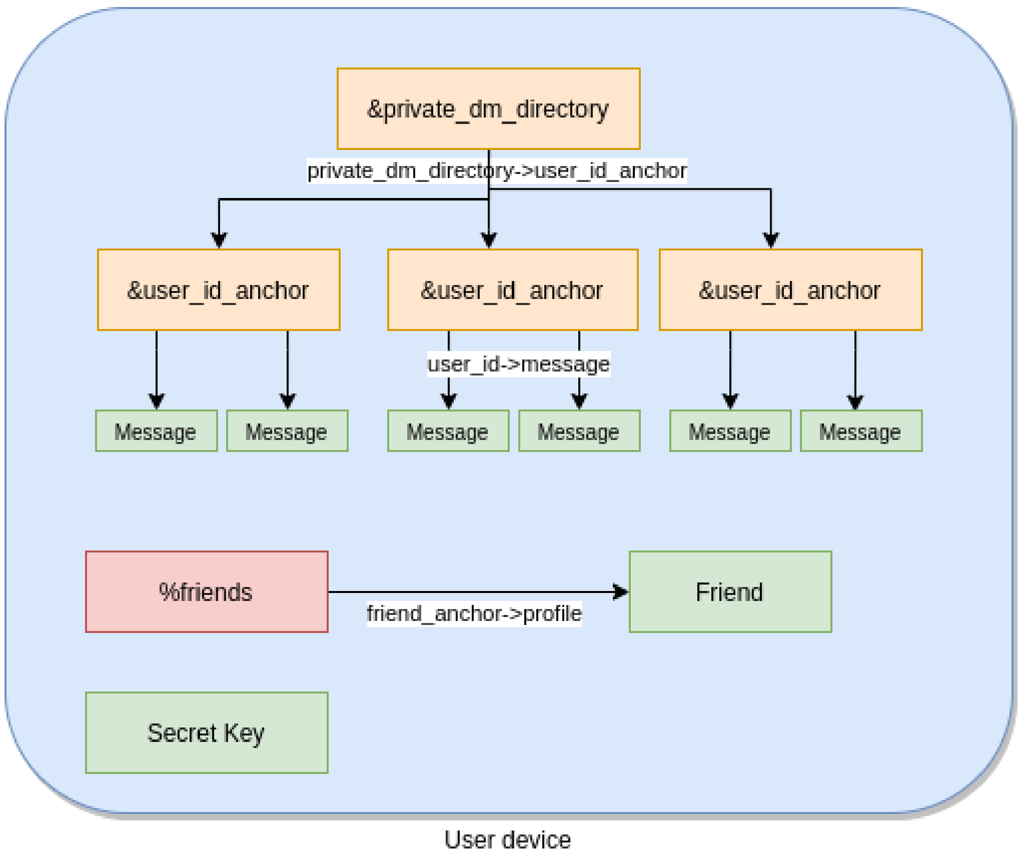


Figure 6.4: Source chain entry type structure, comprising of anchor trees (in orange), anchors (in red) and entry types (in green). The links are explained in between these.

Conversely, the DHT paradigm is quite different. Users have access to three symbolic anchors referring to the profile of users, own user's mailbox in which asynchronous messages are linked to and a directory of public keys, as depicted in Figure 6.5.

The mailbox concept was introduced to go hand-in-hand with asynchronous messaging, that is, when agents need to send and receive messages when one party is offline. Persistent messages are written to the DHT and are linked to an agent's mailbox, which is a one-stop entry where the user, when he/she goes back online, they know where to check for the messages and retrieve them. After retrieving these messages through the links established between the Mailbox anchor and the Async Message entry type, these links are destroyed.

Similarly to secret keys, the Public Key entry type is created every time a user joins the system and it is instantiated exactly one time and it is appended to the `public keys` anchor, to later be retrieved by anyone wanting to encrypt messages with the recipient's public key.

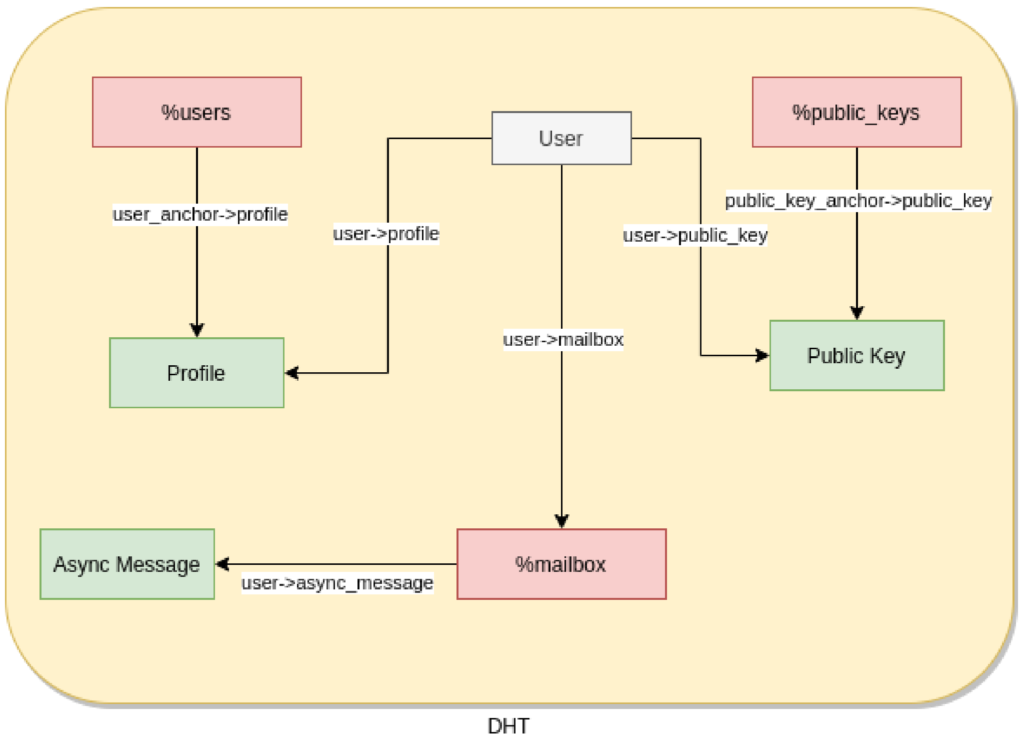


Figure 6.5: Organization of the DHT that includes symbolic anchors (in red) and entry types (in green).

6.4 Data Behaviour

The way data behaves (i.e. is committed, retrieved, updated or even deleted) from the DHT is important to address cases of duplication of data. By default, the data the is posted to the DHT is deduplicated because the content is hashed and a header is appended besides it (from which the CAS will look for). Since this deduplication behaviour is standard in the DHT, it is okay to assume that every entry within the DHT is unique.

However, this behaviour might not be ideal with storage constraints, both on the DHT level and the user's own chain's level. Even though the function provided by Holochain is idempotent [3] and this behaviour technically persists within the DHT, every time something is committed to the DHT (even if it is a duplicated or not), a header with the hash of the entry will be appended to the user's source chain - this is Holochain's default behaviour.

[3] An idempotent operation within the computing paradigm regards when a process has no effect on the final output, even if it is called upon it multiple times.

```
pub fn commit_entry_only_once(entry: &Entry) -> ZomeApiResult<Address> {
    let entry_address =  hdk::api::entry_address(entry);
    match entry {
        Entry::App(entry_type, _) => {
            if entry_address.is_err() {
                return entry_address;
            }
            return match entry_exists_local(&entry_type.clone().
   to_string(), &entry_address.clone().unwrap()) {
                Ok(exists) => {
                    if !exists {
                        return hdk::commit_entry(entry);
                    } else {
                        return entry_address;
                    }
                },
                Err(err) => Err(err)
            }
        },
        _ => panic!("Expected an App Entry TYpe.")
    }
}

pub fn entry_already_exists(address: &Address) -> ZomeApiResult<bool> {
    let initial_exist = hdk::api::get_entry_initial(address);
    return match initial_exist {
        Ok(res) => Ok(res.is_some()),
        Err(reason) => Err(reason)
    }
}
```

Listing 6.9: Simplified snippet of the toolset that verifies the entry before posting into the DHT and the user's source chain.

In other words, on runtime, idempotency is expected when committing the same entry multiple times but, due to Holochain's normal behaviour, it still increases the chain length because the header of created entry hash is appended to the user's own chain. This procedure poses a severe long-term storage limitation and problems for longstanding use of the application since the user's device storage is limited and effectively has to withstand a high amount of transactions since the application is being developed on a messaging panorama.

To address this issue, the deduplication and idempotency concepts need to be moved over to the user's source chain as well to save up as much storage as possible. Therefore, every time an entry is committed to the DHT, the idiomatic procedure it goes through is showcased in Listing 6.9.

In the snippet of code, it is showcased how the entry is validated against the user's source chain is checked for its existence. If the user already has an header hash on their own device, it means that the entry already exists and it will eventually be deduplicated in the DHT. Therefore, the entry is not committed and its corresponding address is returned to the caller and passed as reference to later be used for whatever purpose needed. This behaviour effectively allows for a better storage management.

However, as it was shown in the piece of code in Listing 6.9, the `hdk::api::entry address` call is used. This call reconstructs an address of the given entry data (i.e. fetches the address of an entry from the DHT). This call can be expensive since the DHT has to be traversed to find the entry (even after applying the patters found in Section 6.3.1) and requires a direct connection with it. Therefore, a Global Entry Address Store is used to engage on a faster address lookup.

6.5 Global Entry Address Store

To improve the performance of the aforementioned behaviour, the user's device holds a Global Address Store which, as the name suggests, saves the address of any entry that the user has created or has fetched beforehand. Figure 6.6 showcases the use of the Global Address Store role in the altered entry behaviour process whenever an entry is committed.

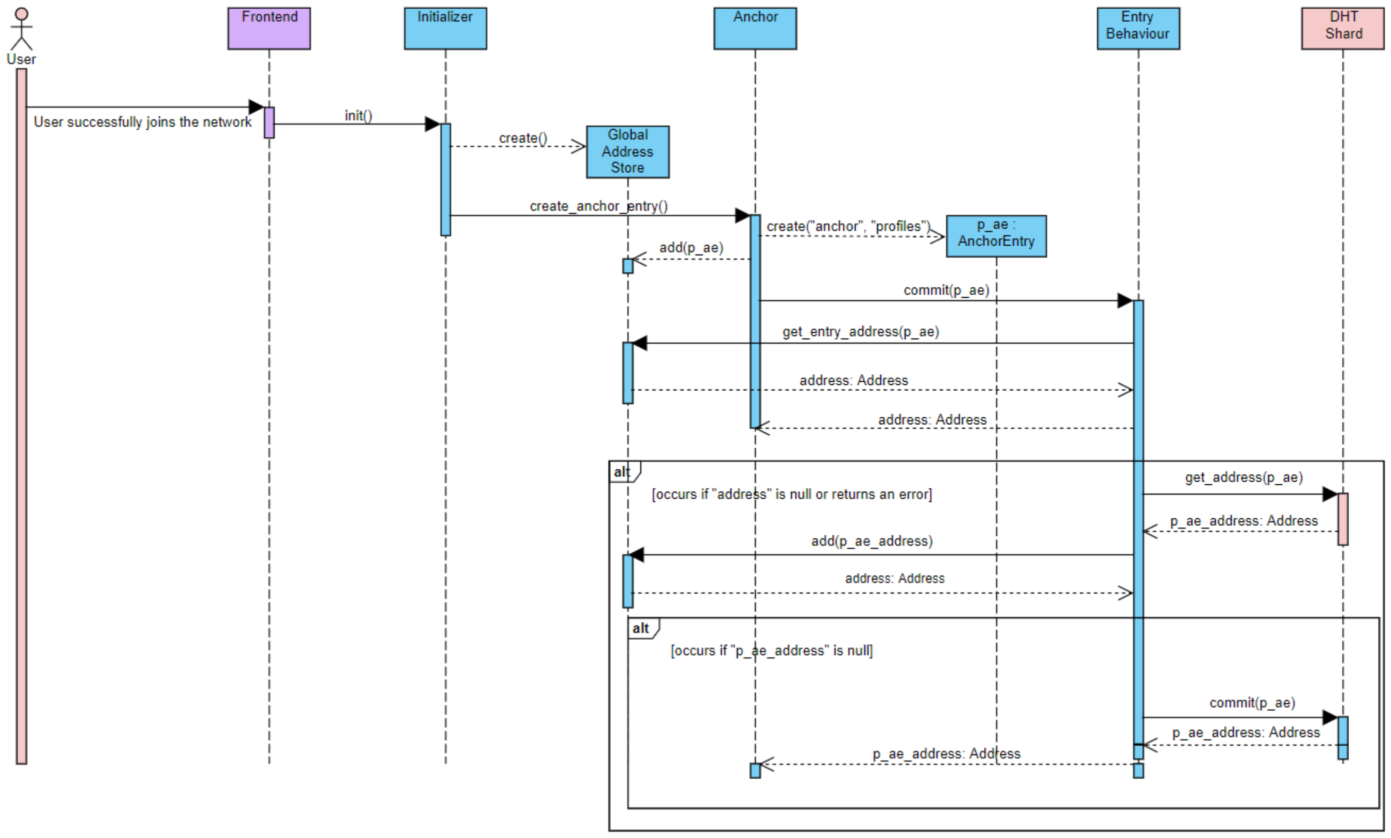


Figure 6.6: Sequence diagram showcasing the flow of execution while committing an anchor and using the altered behaviour to improve performance. In this example, a public anchor entry type is chosen to demonstrate the process and showing the difference with a Global Address Store.

As it is depicted in Figure 6.6, whenever an user joins or logs back in the network, the `init()` function is invoked. This function calls for the creation of all the anchors that were mentioned in Section 6.3.2 and other initialization processes. The call for committing this anchor is using the altered behaviour that was described in the previous section. After creating (if it was not already created) the `Global Address Store` and the `Anchor Entry`, the address of this entry is added to the store. It is worth noting that the store follows an HashMap structure and therefore, does not allow duplicate keys. This prevents for adding an address that already exists.

The `Entry Behaviour` class refers to the altered committing behaviour mentioned before. When the `Anchor` asks to commit the entry, the behaviour checks if it already is in the store and, in a positive scenario, returns the address to the caller (i.e. `Anchor`). In case it is not in the global store, `Entry Behaviour` will fetch the address of the entry from the DHT and add it to the global store, effectively synchronizing the user with the DHT regarding the sought entry.

In case the DHT does not return the address, it means that the entry does not effectively exist. In this case, the entry is committed to the DHT and its address is returned to the original caller to be later used on whatever purpose is needed.

These verification steps translate in lesser calls to the DHT and, consequently, less overhead on the system. All in all, the conjunction of the altered behaviour and the Global Address Store technique is used to avert committing duplicates on the user's source chain and to evade calls to the DHT, thus vastly improving the performance of the network and user's application.

6.6 Communication Flow

Communication between peers in a safe, private and resilient manner is extremely important in the application, as it is directly associated with the main objective of the system. The way the interaction between agents is two-fold: one can publish unencrypted public data for everyone to see or two users can talk with each other directly.

Whenever a user wants to send a message to another agent, the flow of execution alters depending on whether the recipient is online or offline. But on both cases, communication is end-to-end encrypted on a specific implementation that is further explained in Section 6.7.

The diagram portrayed in Figure 6.7 demonstrates the overall process of an user sending another agent a message and how its content is processed throughout. After the user proceeds to send a message, the recipient agent's address is first checked against, verifying if the recipient is a friendly contact. As it was previously mentioned, this requirement is instated to avoid agent spamming from unknown contacts and such.

When fetching the user's Friend entry, the associated public key goes through a verification process (e.g. checking if the key is the correct size) to make sure its integrity is intact, to later be used for encryption processes. Afterwards, the desired content of the message is used to create a Message Spec object which, in turn, represents the specifications to later create a Message entry type.

The Message Spec object is then serialized to later be sent to the user. This serialization process is important for all entry retrieval within the DHT and data passing from each user. Since WASM only supports working with integers and allocating memory manually, all data within the system ought to be serialized because there is not a way that WASM functions can understand the Rust type system natively. To address this, the library `serde` is used to serialize and deserialize strings into JavaScript Object Notation (JSON) objects, allowing us to make Rust's type system and WASM compatible and with decent performance.

```
struct StructType {}

impl From<StructType> for JsonString {
  fn from(own_type: StructType) -> Self {
    default_to_json(own_type)
  }
}

impl TryFrom<JsonString> for StructType {
  type Error = HolochainError;

  fn try_from(string_json: JsonString) -> Result<Self, Self::Error> {
    default_try_from_json(string_json)
  }
}
```

Listing 6.10: Default implementation of deserialization on a structure/entry type.

As it is depicted in Listing 6.10, some entry types implement specific functions that allow to deserialize (from `from` or `try from`) JSON strings into the desired structure. The serialization process, however, is made responsible by the `serde` library.

After the serialization process is complete, the returned serialized object is then encrypted using the recipient's public key, using methods of cryptography that are better explained in the following section.

Having had attained the encrypted object, the latter is sent to the user. There are two possible scenarios in which the user can send the message: either directly, through a Transmission Control Protocol/Internet Protocol (TCP/IP) connection with Web Sockets, whenever the recipient is online and with such connection open; or through the DHT, when the recipient is offline and has no direct connection with the user.

The former has the main advantage of being the fastest and to be safer, since the data is sent through a secure channel that is not available to the public eye. Regarding the latter, even though the cryptography methods make it secure and safe from user's eavesdropping the conversation, every entry shows who authored the entry, leaving a trail of metadata that allows thirds parties to see who is talking with who, even though they do not know the content of the messages.

If the scenario of asynchronous messaging (that is, when the recipient is offline) unfolds, the created encrypted message object is posted to the DHT to later be retrieved by the recipient. The committed entry is effectively linked to the user's mailbox to easily be retrievable. A particular implementation detail is that the message is encrypted through an array of 32 bytes but it exists within the DHT as a Base64-encoded string. This is to make the serialization and deserialization process much easier and to make available the transmission of data from byte into ASCII characters.

After the recipient is back online, he fetches all the asynchronous messages that were sent to him during his absence. Technically, a query is made to the user's mailbox and every linked entry is shown to the user. During this process, the links are removed, to signal that the messages have successfully been received and to avoid duplication on the frontend counterpart.

Afterwards, regardless of the scenario that unfolded, a Message entry is created and then committed to the user's source chain. This also happens whenever a user receives a message from any agent. This is due to the need to maintain the history of messages between each user and having this locally is essential to provide a fully decentralized experience.

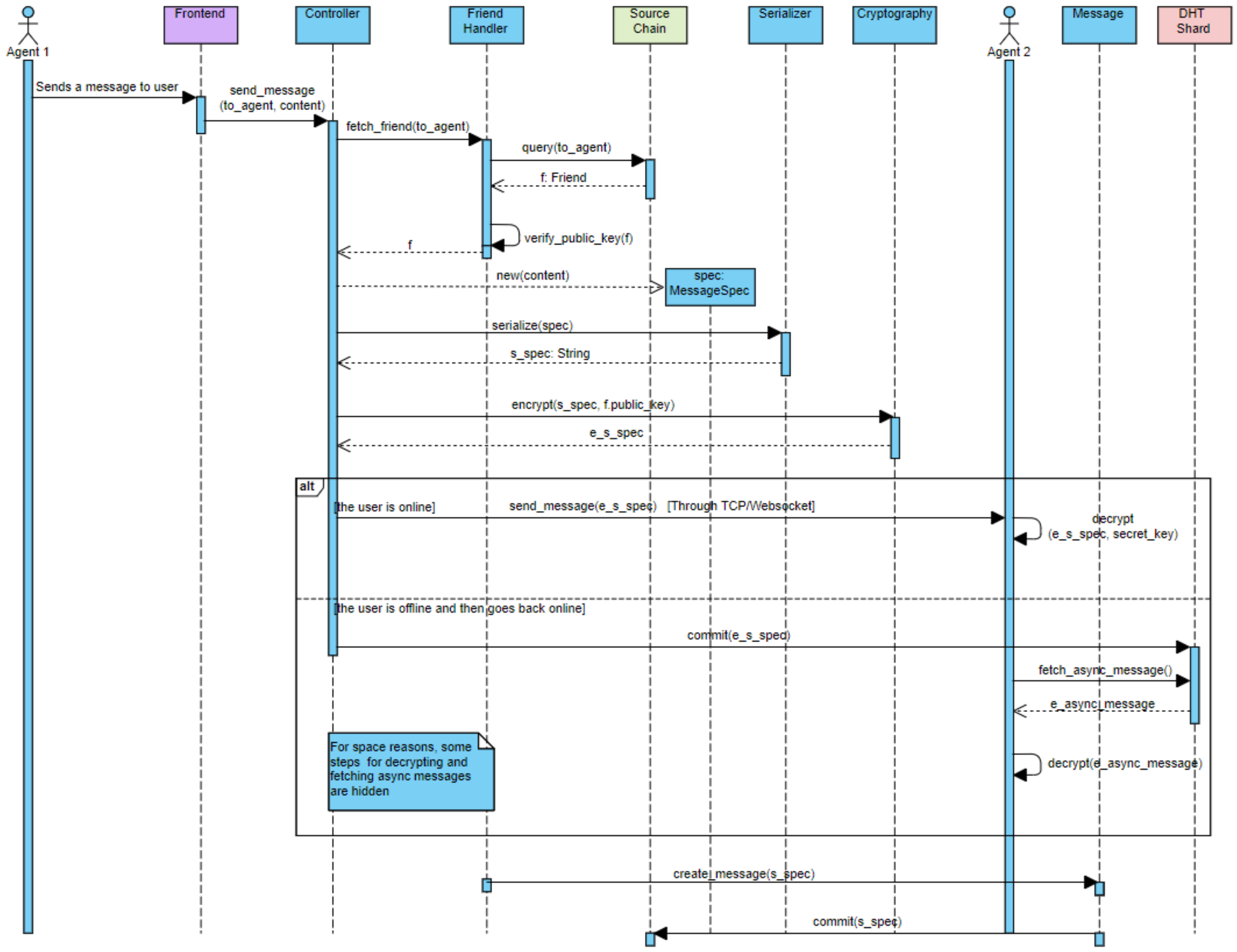


Figure 6.7: Sequence diagram providing a high level overview of the normal execution of when a user sends a message to another agent, outlining the different when one party is offline.

6.7 Cryptography

For the communication between agents to be secure, up-to-date and secure cryptography algorithms must be employed to comply with the main goal of the application. Various concepts were embedded in the implementation of a strong enough algorithm that would hold up to common current systems, employing peer-reviewed cryptographic methods.

6.7.1 Randomness

Randomness is crucial in cryptography. Intuitively, this provides a way to create information that a possible malicious third-party can not learn or predict. The quality of the random numbers used directly determines the strength of the security within a system and it directly influences how difficult it is to break into the system. Therefore, it is important to ensure secret keys are random since it has direct impact on the security against attacks (Gennaro 2006).

With this in mind, most languages provide a way to obtain a random number with an assigned length of bits quite effortlessly. However, considering how Web Assembly is being used and how the application is being compiled to `wasm32-unknown-unknown`, it is hard to determine how to obtain a random number when, at compile time, there is no way to know the runtime environment. The `wasm-bindgen` library, even though is in the phase of adding support for this, it is not stable and does not guarantee a successful compilation and, subsequently, it is not a viable option.

Therefore, to generate random numbers, this must be implemented manually. Is it normally advisable to have a generator that follows the Cryptographically Secure Pseudorandom Number Generator (CPRNG) sequence and is, subsequently, suitable for cryptographic usage. Having an algorithm that follows CPRNG's properties will result in generation of keys, nonces or salts that are high quality and significantly harder for intruders to crack.

There are algorithms that follow the CPRNG guidelines that are effectively used during the encryption process (this will be explained in depth in the following section). The reason for having to implement our own random number generator is merely to having a starting number to seed one of these algorithms that is used. Therefore, the manually implemented random number generator is not necessarily a CPRNG because there is no need to; it is simply to generate a random enough number to be fed into one of the CPRNG-compliant algorithms.

The implementation of our own random number generator follows Kuznetsov (2017)'s algorithm, which is depicted Listing 6.11 and shows the generation of a number based on a seed and `M`. The value of `M` is supposed to be $M = p \times q$, where p and q are high value prime numbers. Afterwards, this is used in calculating a random number, alongside with a seed. This is a snippet of the random generation found in the Rivest–Shamir–Adleman (RSA) algorithm [4]. Using the shown values of p and q are important for generating a long sequence. Although it lacks cryptographic strength, as it was stated before, it is not needed for this situation,

[4]The RSA algorithm was one of the primordial public-key cryptographic methods and it is used as core in secure data transmission. Public keys using this algorithm are created based on two prime numbers and an auxiliary value.

```
const M: usize = (38 * 4 + 5) * (62 * 4 + 3);
static mut SEED: usize = 0;

fn next_random(n: usize) -> usize {
    let x: f64;

    unsafe {
        SEED = SEED * SEED % M;
        x = ((n - 1) * SEED) as f64 / M as f64;
    }

    x.round() as usize
}
```

Listing 6.11: Kuznetsov's random number algorithm showcasing the use of a seed and a variable M.

The seed for the algorithm is, however, based on Holochain's feature of every agent's address being unique. Taking this fact into advantage, we guarantee that every seed for every user is unique and, therefore, the random generated is truly random. In addition to this, so as to not make the generation of this number predictable, the seed also consists of the output of a library called `fastrand`, which is a no-dependency generator of cryptographically insecure numbers, based on Permuted Congruential Generator (PCG) pseudorandom number generation algorithm (O'Neill 2014). The seed function is showcased in Listing 6.12.

```
unsafe {
    SEED = (OWN_ADDRESS.clone() + OWN_ADDRESS.clone() % 9) +
    fastrand::i32(..) % 13 as usize;
}
```

Listing 6.12: Seed used in generating random numbers.

The reason a PCG was not used by itself to generate a random number was simply because having it seeding the created algorithm makes the output of the latter more unpredictable and with a higher level of provable randomness. It does not make it cryptographically secure but it is a simple, lightweight extra step to strengthen our implemented random number generator.

It is important to reiterate that this random number generator is not a truly secure, cryptographically strong and safe random number generator. This is merely used to obtain a decently random number to then feed to CPRNG algorithms that will truly output a strong, safe random number than can then be used in cryptographic methods that are better explained in the following section.

6.7.2 Implemented Protocol

The implemented protocol on message exchange, and to make it viably end-to-end encrypted, the algorithms and procedures implemented take inspiration from the Signal Protocol, including its handshake mechanism [5] and primitives [6]. The Signal Protocol has been peer-reviewed

[5] Handshake is the process where two participants exchange information and establish a protocol of communication before full communication begins.

[6] Cryptographic primitives are rather low-level algorithms that are the backbone to building cryptographic protocols. These can include, for example, encryption functions.

and approved for security (Cohn-Gordon et al. 2017), and blends a methodology that can be found in TLS connections that provide Forward Secrecy, which are highly recommended as per Sheffer, Holz, and Saint-Andre (2015).

The implementation of our key exchange and messaging protocol is based from the Elliptic-curve Diffie–Hellman (ECDH) key agreement protocol, which effectively allows two parties to establish a shared secret over an insecure channel. We are considering the DHT space to be an insecure channel since, even though it is a permissioned network, anyone within it can see the data wherein. The public-private key pair for each agent is created through an elliptic-curve function.

More specifically, Curve25519 is used. This elliptic curve can offer 128 bits of security and it was especially designed for usage within the ECDH key agreement scheme, being one of the fastest curves in elliptic curve cryptography (Bernstein 2011). In our implementation, the public and private key pair will have 256 bits of length (an array of 32 bytes), in which this elliptic is capable of securing properly cryptographically (Langley, Hamburg, and Turner 2016). According to Informationstechnik (BSI) (2020), and most recent studies as of 2020, the minimal key size in bits for adequate security in key exchange is 250 bits when using elliptic curves. Within the system, we are using an array of 32 bytes, which translate to 256 bits, making it harder for third parties to break the keys if we were using, for example, 128 bit-size keys.

Although secure, ECDH can be subject of man-in-the-middle attacks because if static public and secret keys are used and a malicious third-party manages to compromise one's secret key, all of the messages written beforehand are compromised. This vulnerability ties together with the concept of FS, which refers to a protocol's ability to self-heal, that is, to protect past sessions against future potential compromises of secret keys. With this in mind, it is imperative for our application to implement FS effectively.

With the above in mind, our implemented messaging protocol, in order to include FS, bases off the Elliptic-curve Diffie–Hellman Ephemeral (ECDHE) algorithm. The way Ephemeral Diffie-Hellman differs from static Diffie–Hellman (DH) is on the way private/public key pairs are used. On static Diffie-Hellman key exchanges, both keys are always used on communication between the two peers statically and permanently, not changing as time unfolds. On the other hand, when a key exchange utilizes Ephemeral Diffie-Hellman, a temporary DH key is generated for every connection and message sent, thus the same key is never being used twice. This recurring key pair generation in each session enables FS, which essentially means that if the long-term private key gets leaked or compromised, past communication is still secure. This makes it so that if the malicious third-party wants to access all the communication between two users, he has to crack the session keys for each message, which is extremely hard to accomplish practically.

A pertinent question would be: Why use an Elliptic Curve Diffie-Hellman Ephemeral instead of the classic Ephemeral Diffie-Helman algorithm? The reason is quite simple. When comparing to non-ephemeral DH algorithms, having to provide Forward Secrecy usually comes with more computational cost and has, therefore, an impact on the performance (which is something our application is trying to maximize). The ECDHE variants reduce this computational cost by using an elliptic curve cryptography, which has been thoroughly proved and benchmarked by peers (Sheffer, Holz, and Saint-Andre 2015).

The whole cryptographic messaging exchange protocol implemented by us includes, summarizing, an Elliptic Curve Diffie-Hellman key agreement scheme with ChaCha20 as a key derivation function[7] and primitive for public/secret key generation and ephemeral key pair creation. The ephemeral counterpart of the protocol that grants it FS and authenticates each message is ultimately packaged by a Authenticated Encryption with Associated Data (AEAD) algorithm named ChaCha20-Poly1305. All of these concepts are thoroughly explained step by step on the following sections and an overview of the protocol is provided in Figure 6.8.

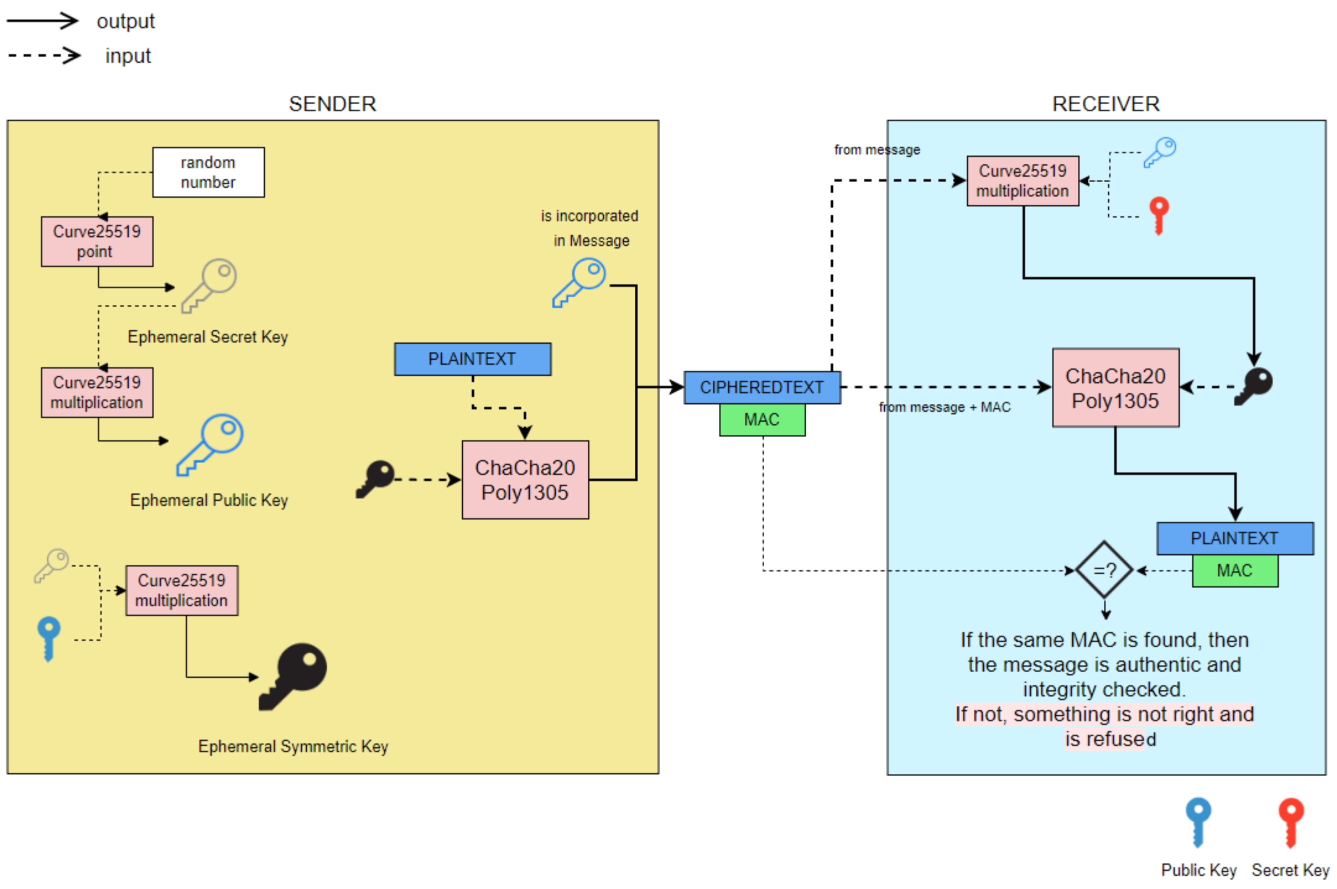


Figure 6.8: Overall representation of our implemented messaging protocol based from ECDHE with FS.

It is worth noting some of the development hiccups that occurred whilst implementing this protocol. The combination of cryptographic algorithms required a library that was effectively used, named `rust-crypto`. However, this library did not support WASM and, by using it directly, the application could not be compiled effectively. Therefore, a fork was needed to make it usable for our application, as well as fixing compilation errors that occurred on MacOS-based systems (Arteiro 2020).

6.7.3 Sequence of the Implemented Protocol

The aforementioned protocol and sequence flow has a specific order that has to comply with regular cryptographic conventions. The public and private key are generated with the

[7] A key derivation function is a function that, having an input of an initial keying material with a fair amount of randomness, derives from it cryptographically strong secret keys.

combination of the creation of a random number with the ChaCha20 algorithm which will later go through a combination of processes to create a secure cryptographic method. The ephemeral key pair creation is constructed and then deconstructed on the encryption and decryption phases, respectfully.

Encryption

The encryption process occurs every time a user wants to send a message to another agent within the network. The public key associated with the recipient is fetched from the local user chain because it is required to be friends with the recipient in order to talk with him/her. If the public key is not accessible, it can easily be retrieved from the DHT and query the Public Key entry type linked with the recipient's address.

The algorithm showcased in Algorithm 6.1 demonstrates the way a message is encrypted, using the recipient's public key to later be uncovered by the same agent with his/her own secret key.

Algorithm 6.1 Encryption algorithm

```
my private key: randomly generated with ChaCha20
my public key: randomly generated and connected with $myprivatekey$ through Curve25519

Input: The recipient's public key and the message being encrypted
Output: Encrypted message

procedure encrypt($rpk, m$)
    $r \leftarrow createRandom$
    $esk \leftarrow Curve25119Point(r)$        ▷ (Create ephemeral key from randomly generated number.)
    $epk \leftarrow Curve25119Multiplication(esk)$
    $symk \leftarrow Curve25119Multiplication(esk, rpk)$

    $c \leftarrow ChaCha20Poly1305(symk)$        ▷ (Encrypts text. Outputs 48 bytes.)
    $tag \leftarrow c.tag$        ▷ (The tag corresponds to the last 16 bytes.)

    for i = 0; i<32; i++ do
        $c.output[i] \leftarrow epk[i]$
    end for

    for i = 16; i<48; i++ do
        $c.output[i] \leftarrow tag[i]$
    end for        ▷ (Packages the ephemeral public key with the encrypted text.)

    return $output$        ▷ Returns the encrypted message
end procedure
```

The generation of the secret key is made through our own implemented random generator with the ChaCha20 algorithm to create a truly randomly generated key. ChaCha20 algorithm is used as a KDF, needing a key and nonce (which are outputted from our implementation of a random number generator) to generate a cryptographically strong key from these input materials. From the secret key, the public key of the user is created through the Curve25519 function, creating a public/private key pair that are mathematically and cryptographically related. The process outlined here is represented in Figure 6.9.

Having both the public/secret key pair generated, the encryption process now can be implemented. As it was stated before, the implemented protocol bases from the ECDHE protocol, which implies the creation of keys for each session which, in this case, translates to every message which, in turn and subsequently, will have FS protection. An ephemeral secret key `esk` is randomly generated, in the same fashion as the user's secret key, which is then passed through a Curve25519 point.

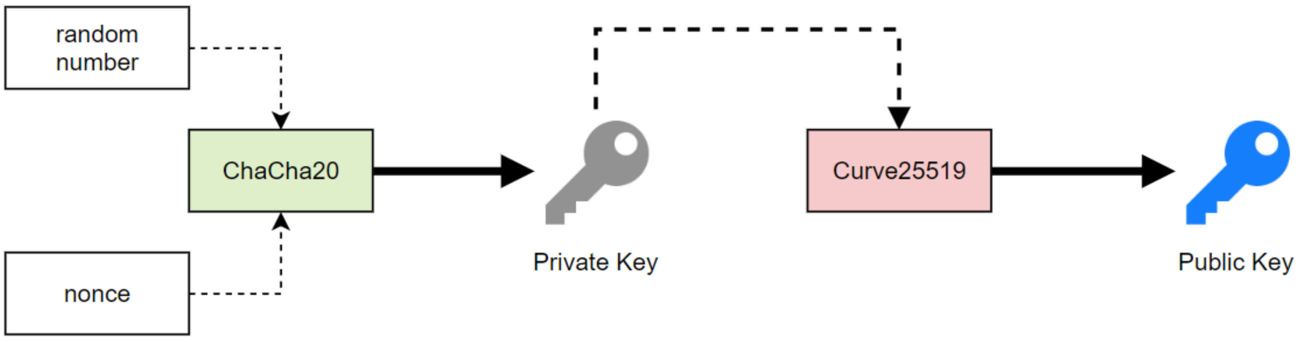


Figure 6.9: Representation of the generation of the public/private key pair. ChaCha20 is used to generate a secret key and Curve25519 is used to make these cryptographically related.

Afterwards, an ephemeral public key `epk` is created derived from the ephemeral secret key through the process of a Curve25519 multiplication. After this, a symmetric key `symk` is derived by combining the permanent recipient's public key with the ephemeral secret key, by the same process of Curve25519 multiplication.

With the created symmetric key, this is then fed to a ChaCha20-Poly1305 algorithm. ChaCha20 is a stream cipher[8] and is also considered a CPRNG (which was previously mentioned that was going to be used) and provides 256-bit security (Shi et al. 2012). On the other hand, Poly1305 is a cryptographic Message Authentication Code (MAC), also known as a tag, which is a snippet of information trailing the payload that is used to authenticate a message - it is therefore used to confirm that the received message came from the respective sender and has not being changed, thus guaranteeing the message's data integrity and authenticity by allowing the recipient to detect any changes to the content of the message.

Although both the ChaCha20 stream cipher and Poly1305 authenticator work as stand-alone algorithms, when combined they become a AEAD which indicate a system that supports simultaneously the confidentiality and authenticity of data. The implementation of ChaCha20-Poly1305 used had a specific authenticated encryption approach, named Encrypt-then-MAC (EtM), which is represented in Figure 6.10. This approach entails that the plaintext is first encrypted and afterwards, the MAC tag is constructed based on the resulting ciphertext, both being sent whilst packaged together.

The EtM approach is the standard method for ISO/IEC (2009), provides the highest definition of security when the MAC is fairly unforgeable when compared with the other possible approaches MAC-then-Encrypt (MtE) and Encrypt-and-MAC (EnM) (Bellare and Namprempre 2000) and has been thoroughly studied, peer reviewed and is inclusively used in TLS-based communications (Gutmann 2014).

Having had explained the implementation details of the ChaCha20-Poly1305 and how it was used, it is suitable to acknowledge that this AEAD are one of the primitives of our application protocol. And an AEAD was chosen because it combines two different algorithms - cipher and

[8]A stream cipher is a symmetric key encryption/decryption algorithm that encrypts 1 bit or byte of text at a time and are combined with a keystream.

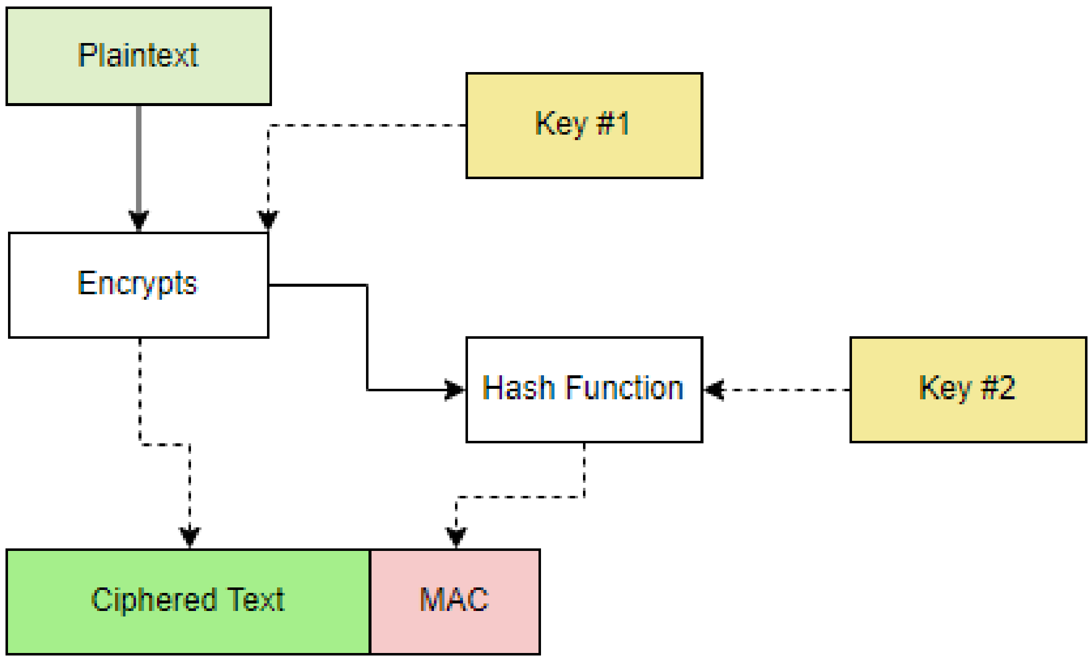


Figure 6.10: Data flow representation of EtM within the ChaCha20-Poly1305 implementation used where the first 32 bytes are the ciphered text and 16 bytes are the MAC tag.

MAC - into a single primitive with provable security guarantees. This allows communication to be effectively safer and much harder to be compromised.

As it is depicted in Algorithm 6.1, as the final step of encryption, the text message is symmetrically encrypted and authenticated by using ChaCha20-Poly1305 (cipher stream and MAC, respectfully), with the output of the text having the ciphered text and the MAC being correctly placed byte by byte, outputting a fully encrypted message with size of 48 bytes, where the first 32 bytes have the ephemeral public key incorporated and the remaining 16 bytes correspond to the MAC tag.

Decryption

When compared to the encryption process, the decryption process is much more simpler, relying more in validating the integrity and length of the received payload. The process of decrypting is compatible with the process of decryption and it is roughly outlined in Algorithm 6.2.

Having had used an AEAD form of encryption, the authenticity of the message can be verified by the combination of the key and the tag, and check if both are not tempered with or forged. If inconsistencies are found, the message is rejected and considered insecure. One of these validations begins with the length of the payload received, which is made as the first step when receiving the encrypted message `em`.

The ephemeral public key created and the tag were sent within the payload, the first corresponding to the first 32 bytes and the remaining 16 bytes corresponding to the MAC tag.

Algorithm 6.2 Decryption algorithm

```
 1: Input: Our own secret key and the encrypted message payload
 2: Output: Decrypted plaintext
 3:
 4: procedure decrypt($sk$, $em$)
 5:
 6:     if $m$.length < 48
 7:      return ErrorMalFormed
 8:
 9:     $epb \leftarrow em[0..32]$                     ▷ The first 32 bytes are the ephemeral public key.
10:     $t \leftarrow em[32...48]$                     ▷ (The tag corresponds to the last 16 bytes.)
11:
12:     $symk \leftarrow Curve25519(sk, epb)$
13:     $pt \leftarrow ChaCha20Poly1305(symk, t, em)$  ▷ Decrypts with symmetric key and tag and errors if invalid.
14:
15:     return $pt$                                    ▷ Returns plain text
16: end procedure
```

A symmetric key `symk` is derived by combining the ephemeral public key `epb` with our own permanent secret key `sk` and this is made through a Curve25519 multiplication operation.

The ciphered text `em` is then decrypted with the symmetric key `symk` generated and the tag received. This is made through a decrypting ChaCha20-Poly1305 process. Within this implementation, the ciphered text's integrity is tested through the Poly1305 method.

Having passed all the integrity and length tests whilst decrypting the ciphered payload and comparing the MAC tag to verify the message's authenticity (if it does not, the error is bubbled up and the message is thus rejected), the plaintext `pt` output is returned.

After this procedure, if the application's logic were not applied, the secret key holder knows what got encrypted but would have no idea who it come from. However, since the recipient is aware of the sender's public key (because of the pre-requisite of them having to be friends), the recipient effectively knows who sent the message.

Chapter 7

Project Assessment

This chapter relates to the overall evaluation of the project and how it is assessed technically and by statistical methods. This stage refers to the later stages of development, when the prototype is functional. This chapter will outline the way the project was tested technically and the first planning made for the assessment of the application, which has suffered changes due to the ongoing COVID-19 pandemic. Nonetheless, this chapter will delve on the assessment of the solution and how it is planned to occur, what criteria are going to be under scrutiny, which hypothesis will be tested and the main evaluation methodology, as well as the testing and its types were made to demonstrate code reliability.

7.1 Testing

Testing the resilience and the inner-workings of an application is a normal and highly regarded occurrence within software engineering, as it allows spotting bugs and reconsidering options to increase the application's overall quality. These are fundamental to prove the application's correct behaviour on runtime and to prove requirement fulfilling.

The testing of the application at hand was made during the course of development, having had used a Test-Driven Development (TDD) approach. To maximize the probable situations to be covered and to maintain quality testing, it was commonly agreed to tackle the testing phase through three approaches: unit testing, integration testing and mutation testing.

7.1.1 Testing Procedure

Since the framework used for DHT interaction is Holochain, testing within it is indeed a peculiar case, since it is needed to see the effects of each function within the DHT to verify if functions are doing it correctly or not. This case is even more relevant in integration testing, when users post and retrieve data from it within an entire scenario.

Luckily for this, Holochain provides a tool to conduct end-to-end/scenario testing that is run on Javascript, called `Tryorama`. It is essentially an orchestrator that allows to create test suites about the behaviour of multiple nodes that share a common DHT. It essentially spawns a DHT and deals with the networking of each user connected to it.

Adding to this framework, we are using `Tape`, a testing harness that effectively allows us to have access to a wider range of assertions (e.g., it provides a way to compare two objects either on a surface level, or "deeply") and provide more meaningful feedback on which tests failed and where it panicked.

Having stated this, the technical testing of the application is done in Javascript, even though the application is written in Rust. Although it makes it possible to test each function and scenario, there is no way to obtain code coverage since there are no bindings from Javascript to Rust in these testing frameworks nor tools that could be attached that could potentially showcase this information.

Overall there were 119 unit and integration tests made with the combined `Tryorama` and `Tape` test suit, as it is demonstrated in Listing 7.1. This output stems from the Tape test harness.

```
21:08:40 [tryorama] info: conductor 'alice' exited with code 0
21:08:40 [tryorama] info: conductor 'bob' exited with code 0

1..119
# tests 119
# pass  119

# ok

Done in 539.30s.
```

Listing 7.1: Output of the Tape test harness showcasing the amount of unit and integration tests made.

Unit Testing

Unit testing refers to pieces of code that test a specific portion of a codebase within no particular context. A unit of code is a small testable part of an application and it can range from whole classes to mere methods. Unit tests are usually short code fragments that constitute a test case and are independent of other. Considering we are following the SOLID principles, each piece of code is intended to do a particular thing and unit testing allows us to make sure the responsibility of a specific method is fulfilled. Long-term wise, unit testing facilitates future changes that the application may encounter in a later stage and it confirms if a specific module performs its duty as intended in different scenarios.

For this project, Tryorama was used for both unit and integration testing. In the case for unit testing, Rust's built-in testing system was used exclusively to test the cryptographic protocol that was implemented, which was developed independently in Rust and detached from Holochain's framework. For the first case, a simple example is found in Listing 7.2, which showcases an example of creating a friend into the source chain. As it is depicted, two agents (Alice and Bob) are created from a conductor configuration that is defined on the test suite initialiser, where the `Tape` test harness is coupled as middleware.

```
scenario("CREATE A FRIEND", async (s, t) => {
    const {alice, bob} = await s.players({alice: conductorConfig, bob:
   conductorConfig}, true);

    const response = await alice.call('meletea', 'chat', 'add_friend', {
        address: alice.instance("meletea").agentAddress,
    });

    await s.consistency();

    t.ok(response.Ok);
    t.deepEqual(response.Ok.agent, alice.instance("meletea").
   agentAddress);
});
```

Listing 7.2: Simple unit test example for creating a friend.

Within this unit test, a zome function is called to add a friend and before making assertions, the DHT waits to be consistent across both agents. As it is shown, in each unit test, there are multiple assertions made possible by the `Tape` testing framework which allows to assert if the expected objects are deeply equal.

Integration Testing

Whilst unit testing focuses on specific code units and are frequently run in a single module and are self-contained, integration testing combines various software modules and tests these as a group. In other words, it encompasses different components within the system and verifies if these work effectively with each other. Integration tests are often used to simulate the process of specific use cases. Integration testing can make use of, for example, databases. In the case of this application, the DHT will serve such purpose and will serve as the middleman component between two agents and the possible interactions that can occur between these two parties

Similarly to unit testing, the combination of `Tryorama` and `Tape` was used to implement these scenarios and, besides the established goal use cases, there were a other scenarios that were tested. The list of scenarios that were included can be found in Table 7.1.

As it is shown in Table 7.1, integration tests were split within two paradigms: communication and cryptography. The latter was tested purely on Rust whilst the former was made with `Tryorama`. It is worth nothing that tasks such as, for example, fetching another user's profile, creating a new message to be saved locally, creating a new friend, removing said friend or even generating a public key pair, are not contemplated in the aforementioned table because these constitute unit testing procedures. A table demonstrating each unit test would be unfeasible to showcase because, as it is shown in Listing 7.1, a total of 119 unit/integration tests (with an additional six related to the Rust-exclusive cryptography counterpart) were conducted.

Table 7.1: List of scenarios which were verified through integration testing.

Component	Scenario
Communication	Sending a direct message to another user online.
	Sending a direct message to a non-friend and be barred.
	Establishing friendship and allowing communication.
	Receive friendship, accepting it, exchange dialogue and then remove. Check if communication is still possible
Cryptography	Generate key pair, create message, encrypt it, send it over to recipient and decrypt the message successfully. Generate faulty key pair **OR** corrupt ephemeral public key **OR** corrupt ephemeral secret key **OR** corrupt ephemeral symmetric key **OR** incorrect payload length and error the process when any of these occur, meaning the message was tempered with.

Mutation Testing

Although integration and unit testing together are usually enough to sufficiently test an application, is is crucial to ensure the quality of these tests are up to par and that these unit tests are not sloppily implemented. Mutation tests aid controlling the quality of the tests and make it easy to infer if these are correctly testing a test case or not.

In short, mutation testing involves modifying the test in small ways and mimic typical programming errors (e.g. switching variable names or operators or by dividing each expression by zero) and force the creation of higher quality tests. The purpose of this is to help the tester in charge to develop effective tests and locate such weaknesses that are usually overlooked (Budd et al. 1979).

Whilst tests were being created, a tool named `mutagen` was used to make sure the tests were reliable. All of the aforementioned tests passed this test, inclusively the cryptographic Rust-based ones. On Rust, to indicate wanting to mutate a test, a `#[mutate]` macro is placed on top of it to indicate it is needed to be mutated on runtime. All unit tests were target of this and, after checking their output, `mutagen` outputs a file showcasing all successful mutations that were created and on each unit test it occurred and in the specific line the source code was changed. This is depicted in Listing 7.3.

```
{"id":1,"mutator_id":1,"mutation":{"impl_name":null,"fn_name":"
    general_process","mutator":"lit_int","original_code":"32","
    mutated_code":"33","source_file":"src/utils/curve25519.rs","
    location_in_file":"93:71-93:73"}}
```

Listing 7.3: One line of the output of mutagen.

7.1.2 Further Cryptographic Testing Considerations

The implemented messaging protocol and the inherent cryptography has been tested for cases where the payload has been compromised by a foreign entity with probably malicious intents. Inherently to the methods that were used, a few other attacks are mitigated by nature of using the ECDHE flow sequence, respectively peer-reviewed.

Man-In-The-Middle Attack

Man-in-the-middle attacks are also known as "hijack attack" and refers to when an attacker secretly interferes within a communication and alters it, making it believe that both parties are communicating with each other but are, in fact, doing so with the third party. Attackers may be able to eavesdrop and control the entire conversation by impersonating the recipient of the message.

Since the attacker aims to evade mutual authentication, a man-in-the-middle attack may only succeed only when they impersonate each endpoint well enough to satisfy the protocol's expectations. To tackle this, the authentication counterpart of the protocol that was outlined in Section 6.7.2 is handled by the AEAD algorithm based on ChaCha20 and Poly1305 authentication code and it is used every time a message is sent, comparing the MAC tag once a message is received. TLS proves to be effective against man-in-the-middle attacks because of this very system and by using Elliptic Curve Cryptography (ECC) on key agreement in their handshakes and on authentication mechanisms (Nir, Josefsson, and Pegourie-Gonnard 2018). As Nir, Josefsson, and Pegourie-Gonnard (2018) states, this is the standardised way for implementing a secure TLS protocol, going further and stating the usefulness and security guarantees ECDHE provides to these systems that have authentication mechanisms like the ones implemented in our messaging cryptographic protocol.

Our protocol also has self-healing properties and resilience on a long-term perspective, mainly through Forward Secrecy. This is attained, as it was explained in Section 6.7.2 by implementing ephemeral public/private session key pairs that make it impervious for large scale attacks on conversations, even when one of the keys is compromised. Compromising one session will not affect data from other than that exchanged in the session protected by the particular compromised key.

Logjam Attack

The Logjam attack (and vulnerability) was first discovered in 2015 and changed the way TLS and most internet protocols were implemented. This security vulnerability is specific against DH key exchanges and enables the man-in-the-middle network attacker to downgrade a

TLS connection to utilize 512-bit DH cryptography. This effectively effects protocols such as Hypertext Transfer Protocol Secure (HTTPS), which are widely used on the Internet.

Notwithstanding, this attack is feasible even against 1024-bit DH primes. Therefore, according to the paper authors who have discovered this vulnerability Adrian et al. (2015), using primes of 2048 bits or more as a defense is recommended. Or, in alternative, switching to an ECDH-based protocol. This is one of the reasons why elliptic-curve cryptography was used as an approach to public-key cryptography in our protocol.

Timing Attacks on Padding

A timing attack is considered a side-channel attack [1] where the malicious party tries to break into the cryptosystem by analyzing the time taken to execute the cryptographic algorithm. This information can be leaked indirectly through variations in the running times of cryptographic devices and timing measurement and analysis might allow to recover a system's secret key.

An issue that plagued earlier TLS versions were the padding oracle attacks, which uses padding validation of a cryptographic message to decrypt a payload ciphertext. These were mitigated by including an Encrypt-then-MAC extension, which is included in the latest version of TLS (Rescorla and Dierks 2018) which mitigated these kind of attacks which, in turn, is also implemented in the AEAD mechanism used in our messaging cryptographic protocol (Gutmann 2014).

7.2 Experiment Assessment

This section will portray the initially planned procedure to assess the application and the main hypothesis this project attempts to address. Hypothesis testing and statistical analysis will ensue on the performance of the application when compared with other types of messaging applications, either centralized (orthodox) or blockchain-based.

7.2.1 Assessment Criteria and Plan

Similarly to what was discussed in Section 3.6, one of the main criteria and aspect of the application that is being assessed is performance. More specifically, the throughput of the application. This metric corresponds to the delay and the time interval between when the message is sent and when the message is received by the recipient. The scalability issue of a DHT and its storage capabilities is dependent on the number of users and it is safe-guarded by Holochain's framework, which orchestrates such load accordingly and already showcased performance increase with operations on the DHT as the number of users increase (Harris-Braun 2018).

Table 7.2 showcases the criteria that was planned to be evaluated after a functional prototype and their respective definitions. After this, a thorough evaluation of the data gathered and its statistical relevance is conducted in order to prove the scientific validation of the project and

[1] A side-channel attack regards those that are based on information and details obtained from a computer system in lieu of the implemented cryptographic algorithm itself.

the assessment stage. The performance criterion is more relevant to the topic in discussion in this report - the decentralization of the application - in lieu of user satisfaction that is more related with the other thesis inherent to this project related with the interaction between the user and interface.

Table 7.2: Set of criteria that were planned to be assessed upon completion of the project.

Criteria	**Definition**
Performance	Assessment of the throughput of the application (TPS, regarding the delay between messages from the sender and recipient.
User satisfaction	Evaluation of the overall user interaction using the Likert scale (Joshi et al. 2015). Factors within this criterion include: user comfort, interaction speed, accuracy and learning curve.

In addition to end-to-end testing within the framework and code covered by functional and unit testing to showcase code reliability, another criteria (besides the performance of the application) was meant to be covered as well: user satisfaction. This aspect was one of the aspects the application was thriving to obtain good results on, blending both the user interface and backend counterpart. Regarding the BCI evaluation process, it would consist of usability tests on how the platform and navigation could be improved, both being assessed on a typical five-level Likert Scale. This assessment is placed within the overall application goals and is related with the inclusivity aspect of the platform. However, it does not fall under the main research hypothesis nor goal this report strives for, which is providing an alternative with better performance than regular blockchains.

The user interface evaluation process initially was meant to be measured as a metric to check if the main goals of the application were met, in regards to synthetic telepathy counterpart of the application and if it is accurate enough for regular use. For this, a random group of people that have harm/hand impairments or disability would be selected to try out the application. Age is not a factor that is decisive for the experiments however it is important to verify if the solution developed is effective at any age range group.

This group would fill in a user questionnaire pertaining to the five factors referenced in Table 7.2 and a second experiment would be conducted with another sample group to check for improvements made with the platform. However, the user satisfaction experiment could not be conducted because of the rampant COVID-19 situation, having been made impossible to schedule meetings with these kind of individuals at hospitals with effective medical evaluation. Therefore, this criteria is not verified statistically in this project.

However, the performance criteria is possible without the pandemic's restrictions and experiments were conducted to study the performance of the developed application when compared to other techniques and applications. The studies conducted will be explained in the following section.

7.2.2 Statistical Hypothesis

As with any research-oriented project, there is a main hypothesis that one is trying to prove, and that has been mentioned previously, it being:

Hypothesis $\mathcal{H}$ - Decentralized environments and resource sharing based on DHT using a partial consensus protocol have better performance, scalability prospects and maintain a bottom line of data integrity akin to blockchains when compared with the latter with global consensus protocols.

Explaining the aforementioned hypothesis, blockchains usually have an underlying protocol that makes it so that any data that is added to the chain has to reach a global consensus across the nodes to be added. This, of course, can be expensive and slow, especially when users join the network. This is one of the reasons most blockchains do not scale (as stated in Section 4.2.2) (Zheng et al. 2017). The hypothesis at hand is trying to prove that having a decentralized system that works similar to a blockchain but is based on a DHT instead of a chain of blocks can have better performance and have more satisfying scalability prospects when comparing with blockchains. In addition to this, having a consensus that is not global but partial and local to the peers making the data transaction and their neighbouring nodes to validate the transaction, can make the network have a higher performance, throughput and transactions per second when compared with blockchains, and maintain data integrity.

When it comes to the methodology of evaluation after having the respective output and results from the application through the assessment period, in accordance with the criteria that are to be evaluated upon the completion of the project, the hypotheses being assessed are related to its performance, on both mean performance between different applications and on scalability as more users join the network.

The metric being tested regarding the performance between applications is the delay between the sender sending a message and the recipient receiving the same, measured in milliseconds (ms). This is essentially the throughput (TPS) of the application when compared to a blockchain-based messenger and a centralized application, verifying if there is a significant difference between the different approaches (centralized, blockchain-based and DHT-based).

Additionally, the scalability will be tested by performing the operation of sending and receiving a message on a network that boasts different number of users, increasing throughout and check if there is a performance gain as more users join the network. This is expected through Holochain's normal behaviour but it is going to be tested nonetheless.

Therefore, the main hypothesis that is going to undergo statistical evaluation will be, on both experimental processes regarding performance and scalability:

$\mathcal{H}_0$ **Null hypothesis** - Every group have the same performance.

$\mathcal{H}_1$ **Alternative hypothesis** - Not every group has the same performance.

7.2.3 Experimental Process for Performance

Any experiment designs begins with measurements and data retrieval. Three different applications were chosen to be under scrutiny, these being Meletea (a code-name to our developed project), WhatsApp (representing a classic centralized application) and Adamant (a blockchain-based messaging application). To conduct experiments on each, it entails a different approach on retrieving data for each one. However, it has been decided to conduct experiments on each approach (group) 150 times. This number was decided to obtain a more reliable and stronger statistical evidence, by having a large number sample's number of the dataset. In each experiment, a single message will be sent between the receiver and the recipient, this process being iterated 150 times.

When it comes to Meletea, testing the delay between sending a message and receiving it was easily tested through the `Tryorama` framework already provided by Holochain that was, inclusively, already used in creating unit and functional testing. To obtain the delay in milliseconds, the after the message was sent and the after the message was received were saved, their difference resulting in the wanted metric. This experiment was conducted 150 times and made use of the asynchronous messaging process, where the DHT is effectively used. Therefore, the metric being evaluated is the time to post data onto the DHT and retrieve it.

Regarding testing the WhatsApp application, a framework was used to send and listen for messages to make this process automated and more consistent. This framework named `Venom` makes it possible to set a chatbot on WhatsApp and automate the process of sending messages periodically and perform actions upon receiving them. Two different numbers were used to simulate a real case scenario and this method was repeated 150 times as well. The snippet of code in Listing 7.4 showcases the receiver component of the small Javascript application that was created for this experiment.

```javascript
const venom = require('venom-bot');
const fs = require('fs')

venom.create().then(async (client) => {
    start(client)
});

function start(client) {
    client.onMessage((message) => {
        let now = Date.now();
        let duration = Math.abs(now - message.body);

        // Write to file
        let row = "whatsapp," + duration.toString() + "\r\n";
        fs.appendFile('data.csv', row)
    });
}
```

Listing 7.4: Receiver component of the Venom experiment. The duration is printed to a file with the duration of the sending timestamp and the timestamp when the message is received.

For the third blockchain-based application, the Adamant platform was used for benchmarking and comparison application, being the most reliable and safe option currently. Adamant works on a fee-per-message model, which means that every message is securely sent for a small price. Luckily, Adamant provides an initial amount of their own cryptocurrency (which is the currency used for each fee) to give the platform a whirl. For this, a Python script was created to automate and scrape the Web-version of the platform, making use of `BeautifulSoup` to scrape the website for new messages and, upon these, calculate the difference and output the results to a file. The snippet of code in Listing 7.5 outlines an example of logging into the platform and maintaining the session to a chat URL.

```
from bs4 import BeautifulSoup
import requests

# Start the session
session = requests.Session()
# Create the payload
payload = {'_passphrase':'<passphrase>'}
# Post the payload to the site to log in
s = session.post("https://msg.adamant.im/", data=payload)
# Navigate to the next page and scrape the data
s = session.get('https://msg.adamant.im/chats/U8957503396672188351')
```

Listing 7.5: Snippet of the scraper script. This piece of code shows logging in the Adamant platform with a specified account and navigating to a chat URL where the scraping ensues.

Similarly to the previous two applications, this experiment was conducted 150 times. Since Adamant only provides 90 messages for free, two accounts had to be created for this to be made possible. The decision to have a high amount of samples on each application was to prove the hypothesis more reliably.

Exploratory Data Analysis

Having had the samples for each type of application, it is often useful to do exploratory data analysis. The metric used is performance, more specifically the delay between sending and receiving a message. This was measured in milliseconds (ms). A script written in R was used to get all of this information and was used throughout the whole experimental process.

Table 7.3: Exploratory data analysis on each application and their corresponding kurtosis, with the minimum, maximum, median, mean and standard deviation also being contemplated, all of these values being in milliseconds (ms).

Application	**Min**	**Max**	**Median**	**Mean**	**SD**	**Kurtosis**
Meletea	3023.36	3659	3366.9	3370.17	109.95	3.08
WhatsApp	1564.04	3822.83	2824.66	2745.66	413.94	2.78
Adamant	5506.64	6629.78	6011.565	6018.16	188.15	3.40

The data shown in Table 7.3 showcases some data regarding each application. By the mean's value of each one, we can assume that WhatsApp (centralized applications) have a better performance. However, the standard deviation of this approach is significantly higher compared to the other ones. This might be explained since Meletea was tested locally and through testing while WhatsApp and Adamant were tested on their own platforms which might have to deal with a workload from other pre-existent users (and possibly other factors related to internet connection). This is further demonstrated in the minimum and maximum values. WhatsApp could have been effected by factors such as heavy network use or periods of increased traffic that may create congestion within the network. This might explain the time samples that were obtained during experimentation. Adamant and Meletea do not have such problem because of their low user base and local testing environment, respectfully.

Additionally, since each group was tested in a different environment, each one in their own platform, the samples are considered as unpaired, or in other words, independent.

The kurtosis values signify, intuitively, the degree of flatness of the distribution curve and to what extent the distribution is flat. A normal curve would be equal to three. As it is shown in Table 7.3, Adamant (blockchain-based application) and Meletea are more than three (which refers to a normal distribution curve). This means that, unlike WhatsApp's distribution that is platykurtic (meaning it is lower than three and is flatter), Adamant and Meletea have a leptokurtic distribution.

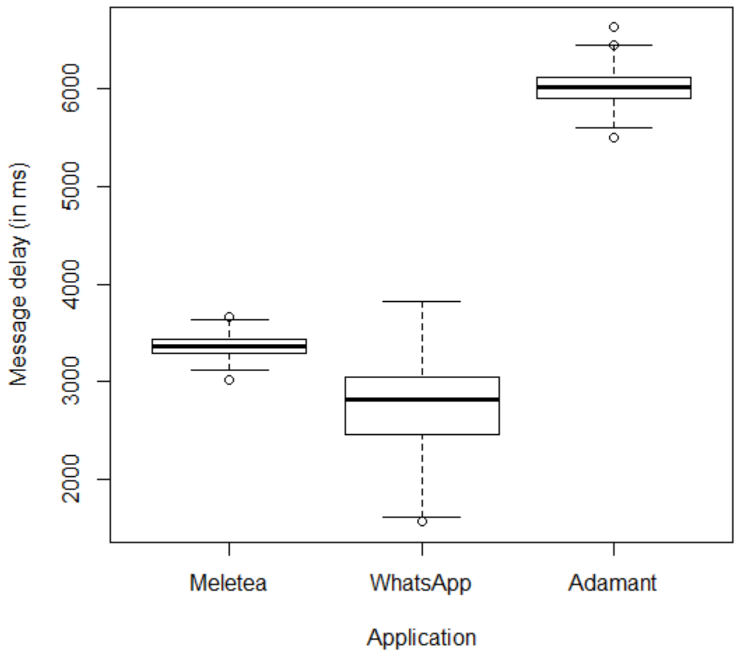


Figure 7.1: Boxplot of each platform compared to each other.

In Figure 7.1, a box plot comparing the performance of the three platforms is shown side-by-side. It is quite noticeable that the centralized-based approach (WhatsApp) has a slightly better performance when compared with Meletea, albeit the samples being much more widely distributed. In spite of this, checking if there is a significant difference is crucial and therefore statistical testing has to be conducted to reach a more conclusive answer. The blockchain-based approach (Adamant), however, fairs much more poorly when compared with their peers, even with a lower standard deviation when compared with WhatsApp.

These boxplots provide further information on the range of distribution of the samples within the experiments that were conducted. It is clearly shown that Adamant has a higher number of outliers which, in turn, prove the value of the kurtosis of the application. Since it is much more weighted on the center of the distribution, tailing values will be considered outliers and, therefore, abnormal to the overall experiment. In the figure it is also shown that WhatsApp has an outlier under the 2000 milliseconds mark, showcasing that this occurrence is abnormal and effectively less common. In Meletea's case, being explained by the testing environment, there are two outliers; however these closely tied in a narrow distribution space.

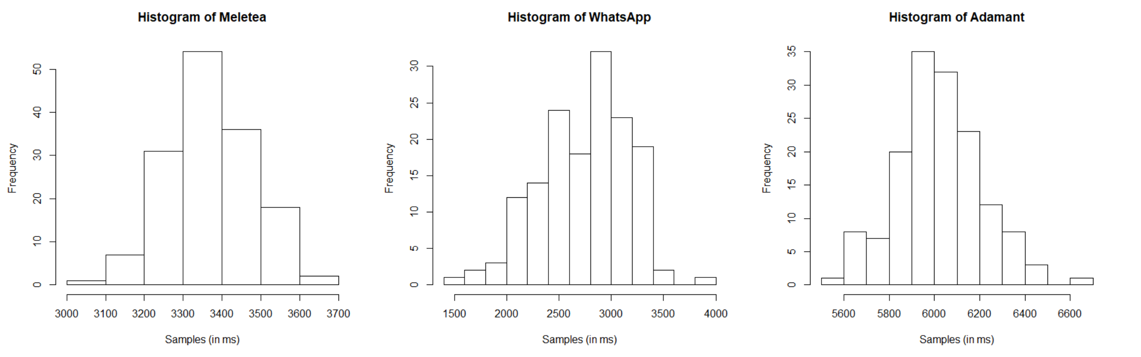


Figure 7.2: Histogram of the distribution of samples of each platform, compared side by side.

These assumptions are further evidenced in Figure 7.2. By first glance, one might assume these distributions are all normal but these will be promptly tested for empirical proof afterwards. The higher sample deviation is clear on WhatsApp. By comparing these histograms, it is possible to assume that a centralized-based approach provides a better performance when compared to a DHT-based or blockchain-based one.

Samples' Distribution

Before proceeding with proving if either the samples are normally distributed or not, it is important to set a good significance level. The significance level is, in other words, the probability of rejecting the null hypothesis when it is true. The choice of this parameter is arbitrary and highly differs from field to field. For this report, the significance level is 0.001, as suggested by Colquhoun (2014), which grants this experiment a false discovery probability below 5%.

Knowing the distribution of the samples is mandatory and an assumption depending on the statistical tests wanting to be used. Therefore, the hypothesis for this assumption to be tested on each platform are:

$\mathcal{H}_0$ **Null hypothesis** - Samples follow a normal distribution.

$\mathcal{H}_1$ **Alternative hypothesis** - Samples do not follow a sample distribution.

Although the Central Limit Theorem[2] could be applied to each group, normality tests will be conducted to prove the null hypothesis. As recommended by Sheskin (2020), the D'Agostino-Pearson normality test is a highly recommended, versatile and powerful normality test as it provides results with better confidence when compared with its peer, such as Kolmogorov-Smirnov. In fact, the latter is not recommended, as per D'Agostino (1986), since it does not take into account the discrepancies at all parts of the cumulative distribution curve.

Depicted in Table 7.4 are the *p-values* yielded by the showcased normality tests. As it was said previously, only the D'Agostino-Pearson one is relevant to this situation but the other

[2]The Central Limit Theorem states that the sampling distribution of the sample mean approaches a normal distribution as the number of samples increases.

Table 7.4: *P-values* of each normality test conducted. Only the D'Agostino-Pearson test is taken into account, the other two are shown to strengthen the normality assumption and showcase difference between normality tests.

Normality Test	Meletea	WhatsApp	Adamant
D'Agostino-Pearson	0.9938	0.06614	0.3585
Shapiro-Wilk	0.9758	0.07175	0.687
Kolmogorov-Smirnov with Lilliefords correction	0.8242	0.01877	0.3574

ones are shown here to showcase that the distributions are indeed normal. Since every *p-value* is above the chosen significance level of 0.001, the null hypothesis is not rejected, having the statistical evidence that the samples follow a normal distribution.

Furthermore, it is common that in all tests, the WhatsApp application scored, albeit with a *p-value* higher than the significance level, much lower than the others. The Q-Q plot depicted in Figure 7.3 makes it easier to discern the normality of the samples. As it is outlined, the sample and theoretical quantiles follow loosely the reference line, meaning that the samples are statistically enough normally distributed.

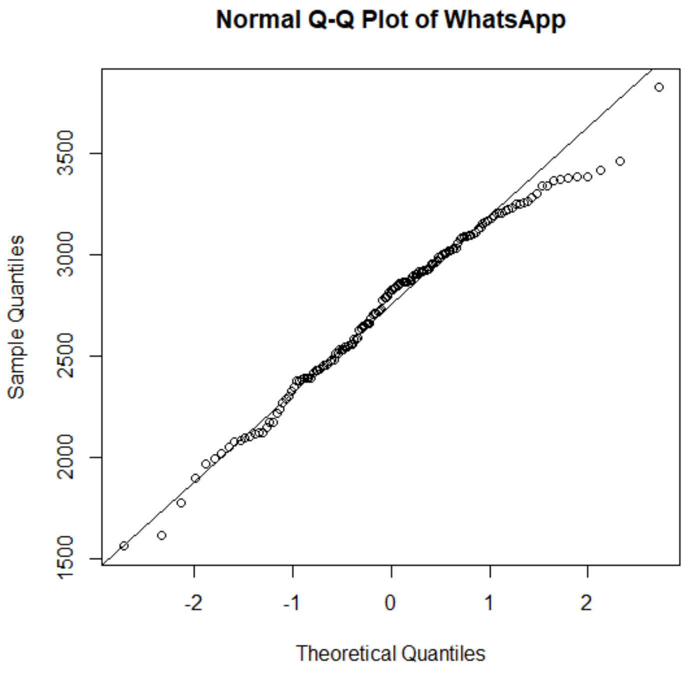


Figure 7.3: Normal Q-Q plot for the WhatsApp platform.

Homoscedasticity Between Groups

On a similar level of importance when compared to the distribution of the samples, seeing if there is a significant difference of variance between groups is also crucial to ultimately pick what statistical test it is going to be undertaken. For this section, homoscedasticity means that there is an homogeneity of variance between groups (platforms). On the contrary of this, if there is a difference between variances, it means there is heteroscedasticity between the platforms.

To this, the hypothesis for the variance of group arises, as such:

$\mathcal{H}_0$ **Null hypothesis** - Groups have a similar variance.

$\mathcal{H}_1$ **Alternative hypothesis** - Groups do not have a similar variance.

For this effect, Levene's variance test can be used to check the homoscedasticity of groups. Bartlett's Test may also be used to check the equality of variances (Snedecor and Cochran 1989), however it is sensitive to departures from normality. This means that if the samples stem from a non-normal distribution, it might not be as effective as Levene's.

Table 7.5: *P-values* of both Levene and Bartlett's variance tests.

Variance Test	**Levene**	**Bartlett**
P-value	2.2×10^{-16}	2.2×10^{-16}

Regardless of sensitivity from the distributions, both tests performed similarly and scored a *p-value* that is inferior to the significance level. Therefore, it is safe to assume that we have statistical evidence to refute the null hypothesis. This means that there is high confidence that the groups do not have homogeneity of variance between them, meaning that they are not homoscedastic.

Statistical Test

For this last part of the statistical analysis, the hypothesis for the statistical test refer to the main one, which is:

$\mathcal{H}_0$ **Null hypothesis** - Every solution have the same performance.

$\mathcal{H}_1$ **Alternative hypothesis** - Not every solution has the same performance.

The One-Way ANOVA test is processed on the assumption that the samples are independent, are normally distributed and have equal variances. As it was proved in the last section, the groups are not homoscedastic, therefore such test can not be used empirically.

In these cases where it is assumed that the groups were sampled from populations of different variances, two pertinent statistical tests arise: Welch ANOVA and Brown-Forsythe. The Kruskal-Wallis test may also be considered. However, the latter is considered a non-parametric statistical test. This type of tests have weaker assumption but are generally less powerful than the parametric ones.

According to McDonald (2014), "the other assumption of one-way ANOVA is that the variation within the groups is equal (homoscedasticity). While Kruskal-Wallis does not assume that the data are normal, it does assume that the different groups have the same distribution, and groups with different standard deviations have different distributions. If your data are heteroscedastic, Kruskal–Wallis is no better than one-way ANOVA, and may be worse. Instead, you should use Welch's ANOVA for heteroscedastic data". Furthermore, there are recommendations to use the Welch ANOVA counterpart when compared with Brown-Forsythe are also supported by Glantz and Slinker (2001), who suggest using the Welch test in most situations, as it has more power and maintains alpha at its desired level, whereas Brown-Forsythe is more relevant when the data is skewed (not normal), which is not the case in our experiment.

Table 7.6: *P-values* yielded from the Welch ANOVA test.

P-value	2.2×10^{-16}

Being showcased in Table 7.6, the *p-value* is significantly inferior to the significance level 0.001, which means that null hypothesis can be rejected and, therefore, conclude, with statistical relevancy, that not every solution does not have the same performance. This is especially true between Meletea (our DHT-based approach) and Adamant (blockchain messaging application).

Post-hoc Analysis

Normally, for a post-hoc analysis, a Tukey's test is used. However, it requires a set of assumptions that our samples do not possess, namely the lack of equal variances. With this in mind, the Games-Howell post-hoc test is the best approach to compare combinations of groups. Even though it is rather similar to Tukey's test in its formulation, Games-Howell's test does not assume homogeneity of variance and sample sizes. This non-parametric test was actually designed with Welch's test output in mind.

This test answers to the following hypothesis, which can indirectly correlate with a significant difference of performance between platforms:

$\mathcal{H}_0$ **Null hypothesis** - Pair-wise groups have a similar mean.

$\mathcal{H}_1$ **Alternative hypothesis** - Pair-wise groups do not have a similar mean.

Table 7.7: Output of the Games-Howell post-hoc non-parametric test, the numbers shown being the *p-values* of pair-wise comparisons with each group.

	Adamant	**Meletea**
Meletea	2.2×10^{-16}	-
Whatsapp	2.2×10^{-16}	2.2×10^{-16}

Table 7.7 showcases the *p-values* of the performed Games-Howell pair-wise comparison test on the normally distributed data with unequal group variances. Since in all three scenarios

(less so between Meletea and WhatsApp) the *p-value* is less than the significance level 0.001, the null hypothesis can effectively be refuted and we have strong statistical evidence that the means between our DHT-based application and a normal centralized messaging application are different, the latter having a better performance.

7.2.4 Experimental Process for Scalability

Similarly to the previous experimental process, this experiment was iterated on a high amount of times. The data retrieval process made use of `Tryorama` and a related library called `tryorama-stress-utils`. This library is "a small API to help with writing `Tryorama` tests involving many conductors and instances" (Dougherty and Mutinta 2019). This is useful is particularly handy to spawn a high amount of users and create test scenarios with networks with a high amount of agents.

```
const { Batch} = require('@holochain/tryorama-stress-utils');
const R = require('ramda')

const NUMBER_AGENTS = 10;
module.exports = (scenario, configBatch, N, C, I) => {
    const totalConductors = N*C;
    scenario('DDOS an agent', async (s, t) => {
        // Every agent will message agent0
        const players = R.sortBy(p => parseInt(p.name, NUMBER_AGENTS), R
    .values(await s.players(configBatch(totalConductors, I), false)))
        await Promise.all(players.map(async player => {
            return player.spawn()
        }));

        const batch = new Batch(players).iteration('series')
        // Lets collect all agents, and use this to reliably enumerate
    an agent by agentAddress
        const agents = await batch.mapInstances(
            instance => new Promise( (resolve, reject) => resolve(
    instance.agentAddress ))
        );
        await s.consistency();

        let agent0Address = agents[0]

        let before = Date.now();
        const response = await alice.call("app", "chat", "send_message",
     {
            to_agent : bob.instance('app').agentAddress,
            message: "lmao"});

        t.ok(response.Ok)
        let after = Date.now();
        let execution_time = Math.abs(after − before);

        writeToFile(execution_time, NUMBER_AGENTS, "./data_scale.csv"));
        t.ok(messageResult.every(r => r.Ok()))
    })
};
```

Listing 7.6: Piece of code in the testing environment with an amount of 10 users. The tools allow batches of users to be spawned.

As it is shown in the piece of code in Listing 7.6, the `Batch` middleware provided by `tryorama-stress-utils` is used to create a batch of users that are spawned within the network and connected on a single mock instance of the network. After this, the function to send the message is called from the backend, recording the timestamps before and after the operation, as well as waiting for the system to be consistent across all users (from the line of code `s.consistency()`). The difference between the time is then recorded and saved to a file to later be analyzed.

This process is repeated 50 times and for a different amount of users. Because of computational limitations, the maximum number of users that it could withstand was 200. Therefore, there are four groups of samples: 10 users, 50 users, 100 users and 200 users, with each one having 50 samples of the test conducted in Listing 7.6.

Having had the data retrieved, the same process as the one made in Section 7.2.3 is made: exploratory data analysis and applying statistical tests to see if there are differences between the groups.

Exploratory Data Analysis

Each sample was measured in milliseconds (ms) and, similarly to the other experimental process, a script written in R was used to obtain information from the samples that were created. Having had obtained 50 samples of each group, it can be considered as an high amount of samples for each group.

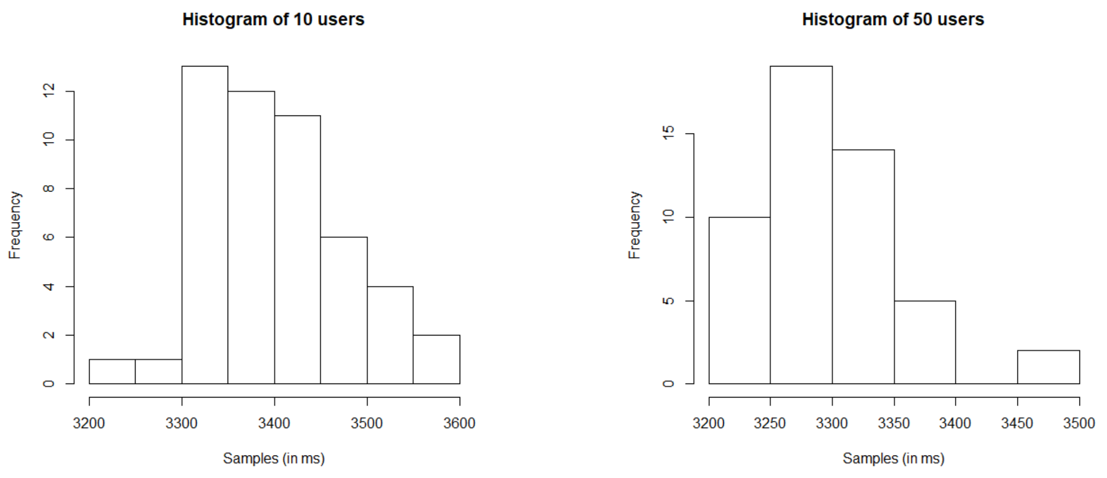


Figure 7.4: Histogram of the distribution of samples of each user groups, 10 users and 50 users respectively.

By observation in both Figures 7.4 and 7.5, it is possible to deduce that the groups related to 10, 50 and 100 users follow a fairly normal distribution. However, the 200 user group shows signs of having a bimodal distribution, albeit this claim is slightly dubious because overall, it may seem that it follow a rough normal distribution.

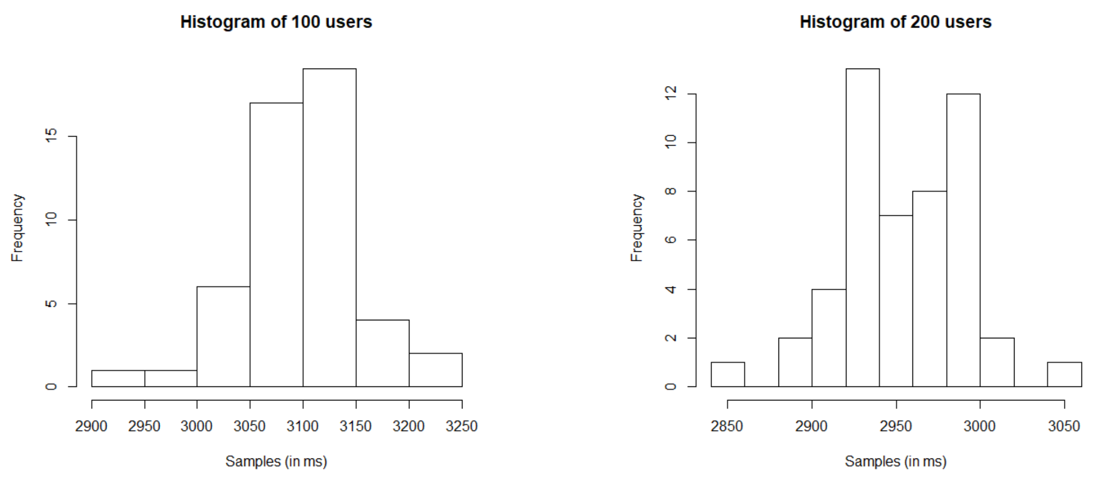


Figure 7.5: Histogram of the distribution of samples of each user groups, 100 users and 200 users respectively.

Statistical Testing

When it comes to statistical testing, choosing a test assumes a set of requirements regarding the distribution, equality of variance and independence of samples. The tests that were overseen in the previous Section 7.2.3 have a similar process to this current one. The tests for the scalability prospects were conducted in the four groups of users, having had concluded that, with a 0.001 significance level, the samples are independent (because they do not stem from the same population and were conducted on different testing iterations), the distributions are normal (the bimodal distribution was rounded down to a normal distribution because it roughly followed a normal distribution and according to D'Agostino-Pearson test, it was above the *p-value*) and the variances are not equal amongst the groups, according to Barlett's test).

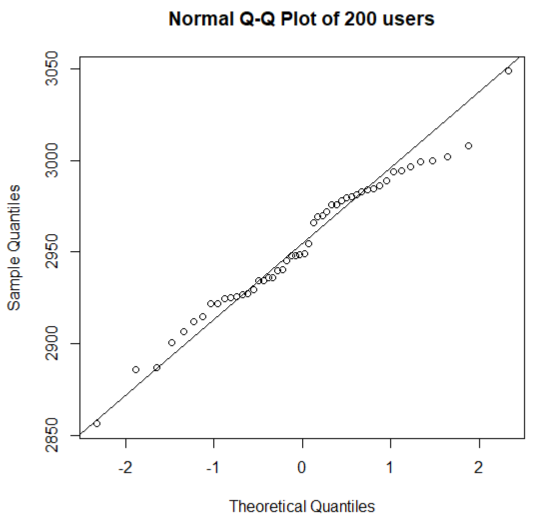


Figure 7.6: Q-Q plot on the group of 200 users performance.

The diagram in Figure 7.6 shows that, although the distribution of samples within the 200 user group is at glance bimodal, it roughly follows the reference line. Therefore, it is not far-fetched to consider the distribution as normal on the statistical experiments being conducted.

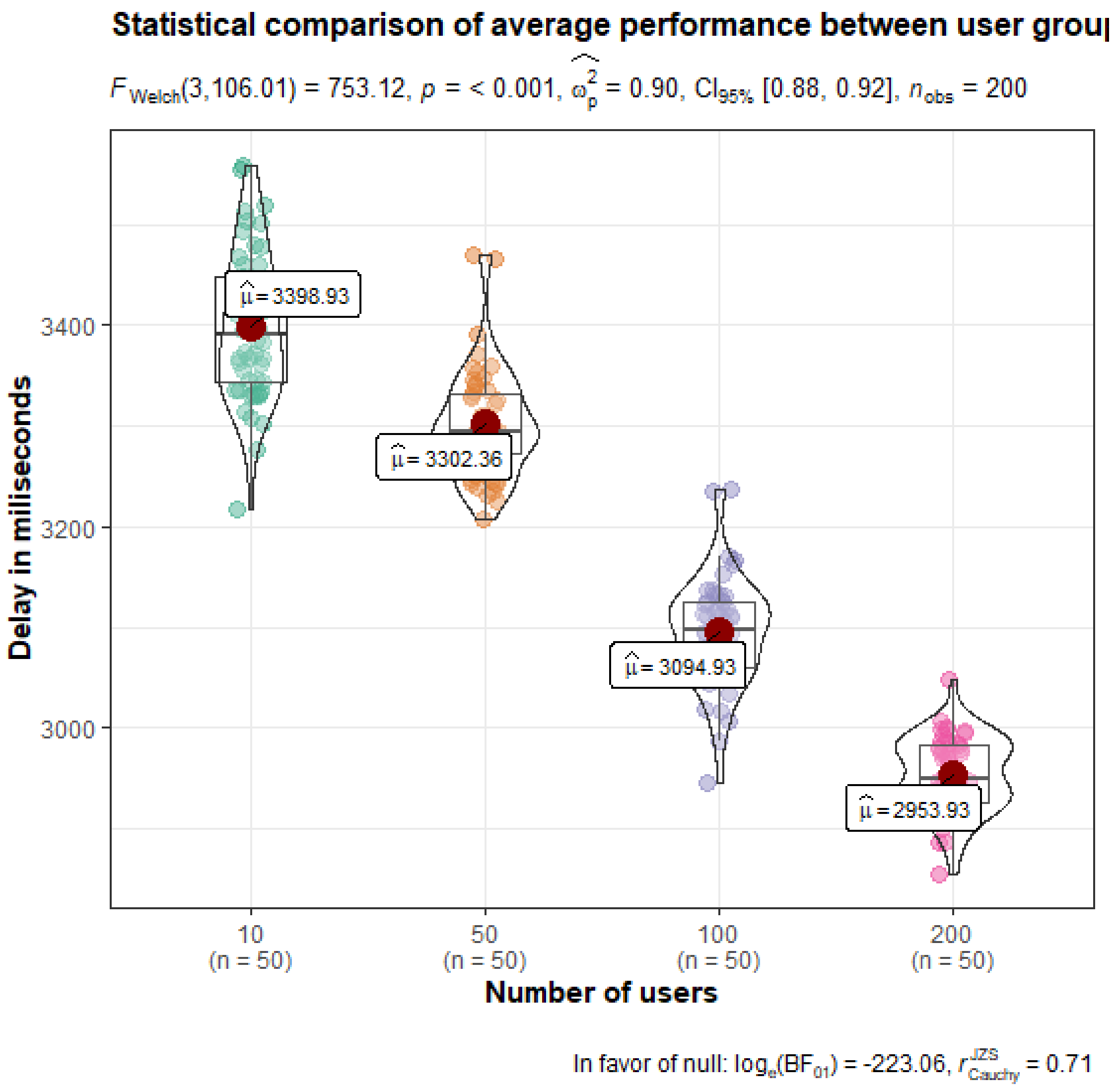


Figure 7.7: Violin plot showcasing each group, their median, the result of the Welch ANOVA test and the number of samples in each group. The plot shows the performance (in milliseconds) dependent on the amount of users and how it slowly increases as more users join the network.

The results showcased in Figure 7.7 show the result of, due to the requirements being the ones aforementioned, the Welch ANOVA test that was conducted on each group of users. This figure also depicts a violin plot and the respective mean of the samples and it is noticeable that there is a different in each group. Although the scale of the plot is in milliseconds and rather small, it shows there is an apparent difference in the performance between the different groups, regarding the interaction between the sender and the DHT. Even if it is a matter of milliseconds, there are performance gains as the number of users grow.

The Welch ANOVA test indicates that the *p-value* is below the significance level and, therefore, there is statistical evidence that there is a difference in the means and performance of each group of users.

7.2.5 Drawn Conclusions

With all of these tests being performed, some conclusions can be drawn. The developed application has statistical evidence that, regarding TPS and performance, even though it slightly fares worse when compared to the well established centralized messaging application WhatsApp, it is still considerably better than a blockchain-based application, while retaining most of the qualities of the latter: integrity and redundancy of data and keeping the history of the records without any possible tampering. It is worth noting that WhatsApp tests showcase a high standard deviation, which can be explained by being tested in an environment that is not controlled and can be affected by internet connection and traffic.

Orthodox applications such as WhatsApp maintain a higher performance/throughput when compared to their peers, most likely due to the level of maturity and longstanding development tradition it witholds. Adamant, being a pure blockchain-based approach, suffers from one of its main issues, which is scalability and performance degradation as more peers join the network. This is evident in the experiments conducted. This does not mean, however, that techniques can not be employed to improve performance, even on the short-term. For example, Layer 2 solutions can be implemented to address lower TPS levels when blockchains scale (Decker, Russell, and Osuntokun 2018), similar to the Lightning Network off-chain operation that was discussed in Section 4.1.2.

On the other hand, the scalability prospects of the application corroborate with the hypothesis that the application increases on the lookup time and interaction with the DHT as more users join the network. Even though this was proved already by Harris-Braun (2018), the results obtained after experimentation reinforce the idea that a higher amount of users allow higher accessibility levels on the data that resides within the DHT. Although a higher amount of users could not be tested due to computational constraints, the data showed showcase a clear tendency of performance improving.

Therefore, the statistical hypothesis that was set out in the beginning of the report is statistically proven with the conducted experiments and tests, with a probable chance of false discovery of under 5%, according to Colquhoun (2014).

7.3 Requirements Completion

Some objectives were outlined in Section 1.4 pertaining to the overall goals of the application and some specific ones related with the decentralized backend component. This section will delve into the achieved goals and what goals were successfully reached whilst developing the project, on an application, functional requirement and non-functional requirement level.

7.3.1 Application Goals

As it was stated in Section 1.4, one of the main issues that the whole application was meant to address was the privacy issue on decentralized system, all the while having a performance that is roughly scalable on a long-term basis. The connection between an interface based on an EEG headset with the frontend and the latter with the dedicated shared backend was successful and has shown results on navigation within the application. The SSR counterpart is still being developed but it is not the focus on this report, which in turn centers on the decentralization part of the project.

Specific to the backend part, one of the main goals pertains to the scalability of the application. This is mostly handled by how users connect to the network and how Holochain orchestrates these connections. As more users join the network, their devices make available storage that can be used to store redundant information from the DHT. Therefore, as the user-base grows, the amount of storage increases and so does data availability. When a user attempts to fetch a specific data from the DHT, it will yield such data from a neighbouring user that holds such data. This is handled by the CAS system appended to Holochain.

Another main goal relates with the privacy component and exchange of data. As it was showcased in Section 6.7, the implemented messaging protocol provides a security that is widely used in current platforms that provide resilience to a myriad of different attacks. This algorithm being influenced by Signal Protocol provides sufficient security for two parties to privately talk with each other, even through a regulated, semi-public medium such as the DHT.

7.3.2 Functional Requirements

The functional requirements of this project were outlined in Section 5.1.1 and being that this project is research-oriented and focuses on technical aspects, the number of use cases is quite limited. Regardless, all of the use cases were successfully implemented, both on the frontend and backend counterparts of the application.

The information depicted in Table 7.8 demonstrates how the implementation of the application correlates with the successful completion of each use case showcased, planned beforehand and marked as functional requirement goals. It is shown that the use cases that were previously planned were completed successfully and that the functional requirements were completed in accordance to the implementation of the application and their entry types.

On the encryption part, the communication is fully encrypted and secure between both users and it is granted by the details that are referred in Section 6.7.

7.3.3 Non-functional Requirements

Regarding the non-functional requirements that are stated in Section 5.1, the usability parameter is non-related to this thesis as it deals with the frontend of the application and that is out of scope of the backend and decentralization part of the project, even if the latter accommodates to the unorthodox navigation and interaction techniques employed.

Interoperability between different systems was also mentioned in the dawn of the project and it is assured by Holochain's framework. Holochain has tools that provide different

Table 7.8: Table showing the implementation concepts that are related with the use case that verifies their complete status.

Use Case	Description
Create contact **Remove contact** **List all contacts**	Contacts can be equated to the Friend entry type. The Friend entry type is also a mechanism to avoid spamming from unknown addresses. Each user has a profile which allows to attach more data to an address and have different, for example, profile pictures and nicknames.
Choose contact/chatroom	This use case is related with the frontend counterpart and does not have an effect on this thesis.
Send message **Receive message**	Both of these use cases make use of the Message and Async Message entry type that are used to maintain an history of messages exchanged with the other users and send messages when the recipient is offline, respectfully. Both of these use cases are also encompassed within the cryptographic messaging protocol that was implemented.

applications built on top of it to interact with other systems, one of these being called Holoscape. Holoscape lets users to install different Holochain-based applications in a single environment and maintain a common profile across different systems within the Holochain ecosystem. Each hApp network are natively interoperable (Holochain 2018).

Reliability and accessibility of the system is guaranteed by the decentralized nature it has. By having no dependency to central servers, the only factor that makes it unavailable for a user to not interact with the DHT is if either the latter has either a poor internet connection or the user-base is so small that data is not accessible and is not made redundant enough. Regarding reliability, network traffic is not an issue that is relevant because connections are not made to server but to a shared DHT that is not present in a centralized location.

Performance-wise, the experiments conducted in Section 7.2.2 showcase that the performance of the application is comparable to current centralized solutions, albeit slightly slower. Even so, when compared to blockchain-based solutions, the performance fares satisfactorily well and maintains a bottom-line as users scale.

Finally, referring to the supportability non-functional requirement, the provided solution is compiled to a WASM format that can be fully supported from a Javascript-based frontend. For this project, the React framework was utilized which indicates support for normal current web browsers. However, if it is needed, React Native could also be employed, therefore making this platform accessible from mobile devices similarly.

The scalability prospects of the system are mostly guaranteed by the behaviour that Holochain has with the DHT, themselves having had inclusively benchmarked the scalability of their applications, corroborating the assumption that their systems scale linearly as more users join the network (Harris-Braun 2018). The more users join the network, more storage is available and a higher amount of addresses near a user that wants to query the data are closer to him, making DHT lookup time not degrade overtime.

One of the aspects that was referred in the hypothesis pertains to this solution resembling blockchain solutions. When it comes to data that is within the DHT, the data is tamper-proof (just like in blockchains) because a random selection of users is chosen to store the same piece of data on their devices and make it accessible and redundant. This makes it so that data has to be changed on multiple places, similarly to a blockchain. This behaviour is handled by the Holochain framework alongside the CAS system, where when data is posted to the DHT, a random selection of users (based on their address within the network) are selected to save the shard of the DHT.

7.4 Project Promotion

One of the main objectives of the project was to receive feedback from the scientific community, thus having scientific validation of the project regarding its capabilities and performance. To promote the project and obtain scientific feedback through peer-reviewed assessment, as the time of writing, three conferences were considered and papers were submitted to these.

7.4.1 14th International Conference on Interfaces and Human Computer Interaction

The Interfaces and Human-Computer Interaction Conference (IHCI) 2020 is a conference which focuses on the main issues concerning interface culture and design, as well as aspects of design, development and implementation of interfaces and their possible applications and implications for human and technology interaction. Therefore, it aims to venture innovative approaches and their applications in interface-related work. This year's edition was to be held in Zagreb, Croatia.

With the help of both supervisors Paula Escudeiro and Carlos Ferreira, a short paper was submitted to this conference showcasing the project context and design, as well as evaluation methods. This paper can be found in Appendix B. The short paper was accepted and was due to be presented between the 23rd and 25th of July. However, due to the ongoing COVID-19 situation, the on-site event was canceled and was replaced with a virtual version. Despite this, the group had to decline the invitation for the presentation virtually because after the cancellation of the original conference and subsequent change to a virtual format, the dates overlapped with the conference that is going to be outlined in the following section.

7.4.2 22nd International Conference on Human-Computer Interaction

The Human-Computer Interaction International Conference (HCII) 2020 is a conference encompassing 21 distinguished international boards from 49 different countries, with two main thematic areas: Human-Computer Interaction and Human Interface and the Management of Information. Regarding the former, it addresses innovative topics regarding

Human-Computer interaction theory, methodology and practice, with the development of novel methods and technologies regarding interface and UI and related backends. This year's edition was to be held in Copenhagen, Denmark.

Similarly to IHCI 2020, an abstract was submitted for this conference detailing its uses and new approach of interaction on the messaging panorama and on decentralized systems to be showcased on poster presentations. This abstract can be consulted in Appendix C. Similarly to the previous conference, the paper was submitted and was promptly accepted for poster presentation and publication on the proceedings of the conference. In spite of that, due to the current COVID-19 pandemic situation, the physical venue for the conference was cancelled and replaced with a virtual version that is being held between the 19th and 24th of July. A poster presentation will ensue during this period to gain feedback on the project and possible future prospects.

Moreover, as it was previously stated, the paper is being published in this year's conference proceedings in the HCI International 2020 - Posters, Chapter Number 1 book published by Springer (Stephanidis and Antona 2020).

7.4.3 36th TERENA Networking Conference

TERENA Networking Conference (TNC), is one of the largest and most renown research and education networking conferences conducted in European soil, allowing to present and network with different universities, industry representatives and organizations. The 2020's edition was to be hosted by the United Kingdom's National Research and Education Network organization and be held in Brighton, United Kingdom. This project was nominated and supported by the Future Talent Programme (FTP20) by Fundação para a Ciência e a Tecnologia (FCT).

The application was made through FCT and, after being reviewed against other applicants, it was accepted to be presented in the conference lightning talks under FCT's alias (the application short paper can be found in Appendix D. Lightning talks consist of five minute presentations to the whole audience of TNC to present and promote the project to a wider audience. For this, three meetings, alongside other presented, were scheduled with Nadia Sluer, a representative of GÉANT, a pan-European data network encompassing research and education networks across Europe, and Michael Koenka, founder of MDK//Amsterdam. These meetings had the goal to further perfect the delivery and fine-tune the content of the presentations that were to be given during the conference and do so effectively.

Even so, because of the rampant COVID-19 dire situation, the conference had to be cancelled and could not be held virtually. Despite of this conundrum, the lightning talks were recorded and sent to the organization to be compiled. The presentation which focused on this paper has been delivered and is awaiting the editing process to later be published by the organizing body of TNC20.

Chapter 8

Conclusion

This chapter is meant to wrap up what was learnt and the future prospects this project may have on a commercial level. It will delve on the key takeaways of using the approach that was undertaken. Additionally, the goals that, due to lack of time or other reasons, were not ultimately met, will be stated and provide reasoning for such occurrence. A final appreciation of the application and work will also be conducted.

8.1 Accomplished Goals

On an overview perspective, all the use cases that were set out to be completed were, indeed, implemented successfully on the backend counterpart. Each user can create their own profile, add people as friends, choose chatrooms and effectively communicate with anyone that is within the network.

The focus of this project laid on outlining a decentralized framework and system that maintains tamper-proof abilities and data records immutability, while providing a better performance and throughput that regular blockchain based applications. These non-functional requirements were similarly met with success with the backbone of the application being based on the Holochain network protocol, which directly makes it that records that are posted in the DHT are made redundant by random user selection that holds these pieces of data on their devices.

The performance of the network grows alongside the number of users that partake in the network. This is because the more users join the network, more storage is provided. And since the information within the DHT is split with a random selection of users, when a specific agent wishes to retrieve a specific piece of information from it, there are more users that can be closer to the agent's CAS system address. Therefore, higher levels of accessibility of data are advantageous for users that are querying the DHT.

The aforementioned are handled with a combination of Holochain and a corresponding CAS system. Work that was presented throughout this report also make use of software engineering patterns to further boost the performance for DHT lookup and general interactions. Having different entry types, global entry address stores, anchors and anchor trees makes the organization of the DHT much more manageable and cut querying times considerable.

Regarding the security concerns that were explained throughout this document, the implemented messaging protocol showcases an example of success for peer communication in a private manner that is resilient to most common current attacks and has self-healing properties, mainly through forward secrecy. The cryptographic employed follow recommendations from Internet Research Task Force (IRTF) and can be considered a successful implementation of a key-exchange-based protocol for communication between agents within the network and follows the guidelines that were first drafted for the project.

8.2 Failed Goals

Albeit the use cases of the project were successfully completed, some secondary goals were initially drafted that were not unfortunately met, mostly because of lack of development time. One of these being the ability to update messages after these are being sent. The messaging process can be split into two different scenarios: the recipient is online, therefore the message being sent by normal TCP/IP connection between both parties; or the recipient is offline, the message therefore being posted on the DHT to later be retrieved.

Dissection this use case, for the first scenario, a way to approach this scenario would be to send a signal (through the same TCP/IP connection) for the recipient to change the message locally and the latter to change his own record of the conversation by updating his history of Message Entry types. This would make it so that the message history would be correctly synchronized on both devices.

For the situation where the recipient is offline, the sender could update the posted Async Messages in the DHT before the recipient goes back online and fetches these. That way, the recipient only receives the updated version of the Async Messages and receives the updated messages accordingly. In this process, the sender would also update their own message history to reflect these changes on their device as well.

Moreover, even though it is an intentional design choice that every user's data is local to their device, there is no current tangible way for users to backup their data to later be used on another device. Since the identity of the user is made through its device address, the implementation of this use case is tricky because it would have to render one of the devices associated to a user obsolete.

Finally, employing post-quantum resilience methods to the protocol was a feature that was wanted but due to being a research area that is still rather abstract and theoretical, resources for implementing these techniques were scarce and was infeasible given the time frame and circumstances of the project.

8.3 Limitations and Future Projects

Although all of the main goals of the project were indeed completed and some secondary *post-hoc ones* were failed, there are some limitations and future prospects to this project that can be outline to extend its usage, range of features and boost its performance and usability.

Regarding the direct interaction with the user and interface, the biggest limitation that imposes regards the accessibility of hardware that is compatible with this project. Having had used an EPOC+ headset, it narrows down extremely the supportability for users who are disabled that want to partake in the network. Additionally, although there have been state-of-the-art strides to make the SSI as accurate as possible, the level of accuracy displayed by most platforms, this one included, is not enough to make text input by muscle activity reliable on a commercial level. However, it is not far-fetched to assume that in a reasonable amount of years this is feasibly possible.

Referring to the backend and more specifically its technical details, one of the big constraints that poses refers to the dependency of the system's performance with the number of users that partake in the network. As it was stated throughout the report, the more users join the network, the more storage they provide and make the data, therefore, more accessible and redundant. This, of course, increases performance and lookup time for users when fetching a specific snippet of data. However, when the user base shrinks, there is a higher payload for each user to bare with the data that is created and this is not sustainable on a long-term basis.

In fact, the accumulation of data throughout time of use is another concern that arises. Although performance is not affected, storage is limited and as time progresses, so does storage need. Although this has been fought by discarding unnecessary data and creating side-channels for direct communication between users, techniques such as throwaway DHTs are useful to tackle this issue.

Another issue that could have been addressed regards the overall privacy of the communications between the users. Having had used Holochain and their CAS system for entries, this means that each entry that is posted to the DHT has the information of its author and its location. This means that every entry shows the address of the author who authored it, therefore leaving a trail of metadata that allows third parties to build a profile of social connections. Although the payload of the message is secure, external users can map out who is talking to who, albeit not knowing what is being exchanged with each other.

On another hand, one interesting prospect for the application would be having a file storage system. Going beyond message exchange, a file system could be developed to share files whilst sending a message. A dedicated zome can be mixed in the DNA of the application and provide basic file storage capabilities. A way to implement this could be setting a file size limit and chunk files to provide uniform data distribution in the DHT and implement a validation function that verifies that the order of the chunks on the recipient counterpart is correct. Each file would have a manifest appended that describes the file as a whole and the process of retrieving all the chunks, which would be the recipient's job to compile these together. This manifest ought to contain an ordered list of the hashes/addresses of the chunks and validation logic.

Another requirement that, although not being specifically planned on the designing phase of the project, would be worthwhile would be a throwaway DHT. This directly correlates with the long-term storage of historical data within the DHT and, subsequently, on user's devices. Currently, the application adopts an *a priory* approach on storage of data, leaving only data that is stored on user's devices that are absolutely necessary. But if, for example, ephemeral data would to be used in the application, this could pose serious long-term storage concerns.

For this, a child DHT can be created for temporary content storage. This could hold data that is considered ephemeral (e.g. a user's activity status) and it can be purged or deleted on an *ad-hoc* basis. Offloading such content from the more persistent DHT and referencing its content by address makes it possible that if people lose interest in the data, they can simply leave the DHT. Whenever the last person leaves, the data disappears, consequently clearing users' devices storage.

8.4 Final Appreciation

On an ending note, this project, although having been research-oriented and its output yielding a product that even though not commercially viable, it can be a great starting point to sparkle the debate between blockchain systems and DHT-based networks in terms of performance and security and as a platform to develop applications on top of it.

There are clear trade-offs when shifting between the three paradigms of centralized applications, blockchain-based and DHT-based. When comparing our approach with, for example, a centralized one such as Signal or WhatsApp, the former being a staple for secure communication between peers, when a user joins our network, they are trading the independence from central servers with leaving a trail of metadata that allows third-party users to see social connection between users. All in all, it is up for the user to decide if they prefer their data to be on a centralized location or on a decentralized environment that leaves a metadata trail. Blockchain does the latter transparently but at a cost of performance.

The question then arises: Is storage and decentralization worth not having central points of failure and a prospect of servers being attacked? There is a clear anonymity and security trade-off to have with such decision that can not be quantified; one either wants data transparency or data anonymity.

When relating to the development and implementation of this solution, it was noticeable how the Holochain framework was very much unripe and in its early stages of development. Being in alpha stage, much concerns related with software engineering and data behaviour had to be fixed or re-implemented to fit the project's needs and were including through pull requests.

Moreover, the addition of WASM to the project highly restricted the potential usage of libraries that relied on low-level, known target machine interactions. This was more explicit whilst developing the cryptographic messaging protocol with the lack of access to machine-level randomness.

Similar to Holochain, the Rust programming environment is also less mature when compared with its longer standing peers such as C++ or Javascript. With this being said, some compilation hiccups did happen throughout the course of development that affected the development time. However, these were not at the same extent as the Holochain ones.

Even though the time to develop did not go as exactly planned, as well as the testing phase due to the abnormal COVID-19 situation, the group aims to have the product in the best condition available for the assessment of the application, even after this document is submitted. Therefore, some features that are not mentioned in this document might be ultimately present in the application later on.

Bibliography

Aaker, D.A. (2002). *Building Strong Brands*. Free. isbn: 9780743232135. url: https://books.google.pt/books?id=Dtq0QgAACAAJ.

Abbott, Martin L and Michael T Fisher (2009). *The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise*. Pearson Education.

Abu-Salma, Ruba et al. (2017). "The security blanket of the chat world: An analytic evaluation and a user study of telegram". In: Internet Society.

Adrian, David et al. (2015). "Imperfect forward secrecy: How Diffie-Hellman fails in practice". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 5–17.

Akbari, Azadeh and Rashid Gabdulhakov (2019). "Platform surveillance and resistance in Iran and Russia: The case of Telegram". In: *Surveillance & Society* 17.1/2, pp. 223–231.

Alexander, Roman (2018). *Iota-introduction to the tangle technology: Everything you need to know about the revolutionary blockchain alternative*. Independently published.

Androulaki, Elli et al. (2018). "Hyperledger fabric: a distributed operating system for permissioned blockchains". In: *Proceedings of the Thirteenth EuroSys Conference*, pp. 1–15.

Armour, Brian S. et al. (2016). "Prevalence and Causes of Paralysis—United States, 2013". In: *American Journal of Public Health* 106.10, pp. 1855–1857. doi: 10.2105/ajph.2016.303270.

Arteiro, Luís (2020). *Rust Crypto WASM repository fork*. url: https://github.com/LuchoTurtle/rust-crypto-wasm.

Baird, Leemon (2016). "The swirlds hashgraph consensus algorithm: Fair, fast, byzantine fault tolerance". In: *Swirlds Tech Reports SWIRLDS-TR-2016-01, Tech. Rep.*

Baird, Leemon, Mance Harmon, and Paul Madsen (2018). "Hedera: A governing council & public hashgraph network". In: *The trust layer of the internet, whitepaper* 1.

Balbaert, Ivo (2012). *The way to Go: A thorough introduction to the Go programming language*. IUniverse.

Bellare, Mihir and Chanathip Namprempre (2000). "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, pp. 531–545.

Benčić, Federico Matteo and Ivana Podnar Žarko (2018). "Distributed ledger technology: Blockchain compared to directed acyclic graph". In: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, pp. 1569–1570.

Béraud, Adrien (2019). *OpenDHT: API Overview*. url: https://github.com/savoirfairelinux/opendht/wiki/API-Overview.

Bernstein, Daniel (2011). *Irrelevant patents on elliptic-curve cryptography*.

Bez, Mirko, Giacomo Fornari, and Tullio Vardanega (2019). "The scalability challenge of ethereum: An initial quantitative analysis". In: *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, pp. 167–176.

Blin, Sébastien (2019). *Jami Technical Documentation*. url: https://git.jami.net/savoirfairelinux/ring-project/wikis/home (visited on 12/12/2019).

Budd, Timothy A et al. (1979). *Mutation Analysis.* Tech. rep. YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE.

Buterin, Vitalik (2013). “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform”. In:

– (2019). “On Sharding Blockchains”. In: url: https://github.com/ethereum/wiki/wiki/Sharding-FAQ#what-are-some-trivial-but-flawed-ways-of-solving-the-problem.

Cadwalladr, Carole and Emma Graham-Harrison (2018). “Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach”. In: *The guardian* 17, p. 22.

Clement, J. (2019). *Most popular global mobile messaging apps 2019*. url: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/.

Cohn-Gordon, Katriel et al. (2017). “A formal security analysis of the signal messaging protocol”. In: *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, pp. 451–466.

Colquhoun, David (2014). “An investigation of the false discovery rate and the misinterpretation of p-values”. In: *Royal Society open science* 1.3, p. 140216.

Cormen, Thomas H et al. (2009). *Introduction to algorithms 3rd ed.*

Cosmina, Iuliana (2017). “Spring microservices with spring cloud”. In: *Pivotal Certified Professional Spring Developer Exam*. Springer, pp. 435–459.

D'Agostino, Ralph B (1986). *Goodness-of-fit-techniques*. Vol. 68. CRC press.

Decker, Christian, Rusty Russell, and Olaoluwa Osuntokun (2018). “eltoo: A simple layer2 protocol for bitcoin”. In: *White paper: https://blockstream. com/eltoo. pdf*.

Dillenberger, Donna Eng and Gong Su (2019). *Parallel execution of blockchain transactions*. US Patent 10,255,108.

Divya, M and Nagaveni B Biradar (2018). “IOTA-next generation block chain”. In: *International Journal Of Engineering And Computer Science* 7.04, pp. 23823–23826.

Dobbelaere, Philippe and Kyumars Sheykh Esmaili (2017). “Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations: Industry Paper”. In: *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*. ACM, pp. 227–238.

Dobkin, Bruce H. (2005). “Clinical practice. Rehabilitation after stroke”. In: *The New England journal of medicine* 352.16. 352/16/1677[PII], pp. 1677–1684. issn: 1533-4406. doi: 10.1056/NEJMcp043511. url: https://www.ncbi.nlm.nih.gov/pubmed/15843670.

Domingues, Michael AP (2017). “Performance testing of open-source HTTP web frameworks in an API”. In: *DSIE'17*, p. 8.

Dougherty, Michael and Ashanti Mutinta (2019). *Tryorama Stress Utils Library*. url: https://github.com/holochain/tryorama-stress-utils.

Durov, Pavel (2019). *200,000,000 Monthly Active Users*. url: https://telegram.org/blog/200-million.

Edspresso (2019). *What is a distributed hash table?* url: https://www.educative.io/edpresso/what-is-a-distributed-hash-table.

El Ioini, Nabil and Claus Pahl (2018). “A Review of Distributed Ledger Technologies: Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, October 22-26, 2018, Proceedings, Part II”. In: pp. 277–288. isbn: 978-3-030-02670-7. doi: 10.1007/978-3-030-02671-4_16.

Enberg, Jasmine (2019). "Global Messaging Apps 2019". In: eMarketer. url: `https://www.emarketer.com/content/global-messaging-apps-2019#page-report`.

Evgenov, Pavel et al. (2017). *White paper: Adamant messaging application*. Tech. rep. url: `https://adamant.im/whitepaper/adamant-whitepaper-en.pdf`.

Facebook (2019). *First Quarter 2019 Results Conference Call*. url: `https://s21.q4cdn.com/399680738/files/doc_financials/2019/Q1/Q1-'19-earnings-call-transcript-(1).pdf`.

Galuba, Wojciech and Sarunas Girdzijauskas (2009). "Distributed hash table". In: *Encyclopedia of database systems*, pp. 903–904.

Gennaro, Rosario (2006). "Randomness in cryptography". In: *IEEE security & privacy* 4.2, pp. 64–67.

Glantz, SA and BK Slinker (2001). *Primer of Applied Regression & Analysis of Variance, ed*. McGraw-Hill, Inc., New York.

Gutmann, Peter (2014). "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)". In: *Request for Comments* 7366.

Haas, Andreas et al. (2017). "Bringing the web up to speed with WebAssembly". In: *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 185–200.

Harris-Braun, Eric (2018). *Holochain Benchmarks*. url: `https://github.com/holochain/benchmarks`.

Harris-Braun, Eric, Nicolas Luck, and Arthur Brock (2018). "Holochain: scalable agent-centric distributed computing". In: *GitHub*. url: `https://github.com/holochain/holochain-proto/blob/whitepaper/holochain.pdf`.

Holochain (2018). *Holochain Holo 101*. url: `https://holo.host/wp-content/uploads/holochain-reorientation-from-blockchain.pdf`.

Hoyle, Roberto et al. (2017). "Was my message read? Privacy and signaling on Facebook messenger". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 3838–3842.

Hur, Christian and Laura Ubelhor (2017). *Developing Business Applications for the Web: With HTML, CSS, JSP, PHP, ASP. NET, and JavaScript*. MC Press, LLC.

Informationstechnik (BSI), Bundesamt Für Sicherheit in der (2020). *Cryptographic Mechanisms: Recommendations and Key Lengths*. Tech. rep. TR-02102-1 v2020-01. Federal Office for Information Security (BSI, p. 86. url: `https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile`.

International Telecommunication Union (2019). *Measuring digital development: Facts and figures 2019*. ITU Publications.

ISO/IEC (2009). *Information technology – Security techniques – Authenticated encryption*. Tech. rep. International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). url: `https://www.iso.org/standard/46345.html`.

Jackson, Richard M (2012). "Audio-supported reading for students who are blind or visually impaired". In: *Wakefield, MA: National Center on Accessible Instructional Materials*.

Jaeger, Erica (2014). "Facebook Messenger: Eroding User Privacy in Order to Collect, Analyze, and Sell Your Personal Information, 31 J. Marshall J. Info Tech. & Privacy L. 393 (2014)". In: *J. Marshall J. Info. Tech. & Privacy L.* 31, p. i.

Jakobsen, Jakob and Claudio Orlandi (2016). "On the CCA (in) Security of MTProto." In: *SPSM@ CCS*, pp. 113–116.

Johnson, Khari (2017). *Facebook Messenger passes 1.3 billion monthly active users*. url: `https://venturebeat.com/2017/09/14/facebook-messenger-passes-1-3-billion-monthly-active-users/`.

Josep, Anthony D et al. (2010). "A view of cloud computing". In: *Communications of the ACM* 53.4.

Joshi, Ankur et al. (2015). "Likert scale: Explored and explained". In: *Current Journal of Applied Science and Technology*, pp. 396–403.

Kaliski, Burt (2000). *PKCS# 5: Password-based cryptography specification version 2.0*. Tech. rep. RFC 2898, september.

Kan, Luo et al. (2018). "A multiple blockchains architecture on inter-blockchain communication". In: *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, pp. 139–145.

Katona, Jozsef and Attila Kovari (2015). "EEG-based computer control interface for brain-machine interaction". In: *International Journal of Online Engineering (iJOE)* 11.6, pp. 43–48.

Koen, Peter A., Heidi M. J. Bertels, and Elko Kleinschmidt (2014). "Managing the Front End of Innovation—Part I: Results From a Three-Year Study". In: *Research-Technology Management* 57.2, pp. 34–43. doi: `10.5437/08956308X5702145`. eprint: `https://www.tandfonline.com/doi/pdf/10.5437/08956308X5702145`. url: `https://www.tandfonline.com/doi/abs/10.5437/08956308X5702145`.

Koen, Peter A et al. (2002). "Fuzzy front end: effective methods, tools, and techniques". In: *The PDMA toolbook 1 for new product development*.

Koen, Peter et al. (2001). "Providing Clarity and A Common Language to the "Fuzzy Front End"". In: *Research-Technology Management* 44.2, pp. 46–55. doi: `10.1080/08956308.2001.11671418`. eprint: `https://doi.org/10.1080/08956308.2001.11671418`. url: `https://doi.org/10.1080/08956308.2001.11671418`.

Kuber, Ravi and Franklin P Wright (2013). "Augmenting the instant messaging experience through the use of brain–computer interface and gestural technologies". In: *International Journal of Human-Computer Interaction* 29.3, pp. 178–191.

Kuznetsov, Arkadiy (2017). *A rust password generator on wasm*. url: `https://arkada38.github.io/2017/12/22/a-rust-password-generator-on-wasm/`.

Kwon, J and E Buchman (2019). *Cosmos Whitepaper*.

Langley, A, M Hamburg, and S Turner (2016). "Elliptic Curves for Security (RFC7748)". In: *Internet Engineering Task Force*, pp. 1–22.

Lanning, Michael J (1998). *Delivering profitable value*. Perseus Books Group Cambridge.

Lapaire, Jean-Rémi (2018). "Why content matters. Zuckerberg, Vox Media and the Cambridge Analytica data leak". In: *ANTARES: Letras e Humanidades* 10.20, pp. 88–110.

Lapierre, Jozee (2000). "Customer-perceived value in industrial contexts". In: *Journal of business & industrial marketing*.

Leavy, Thomas Michael and Gerard Ryan (2018). *Decentralized authoritative messaging*. US Patent 10,142,300.

LeBlanc, Maurice (2008). *"Give Hope - Give a Hand" - The LN-4 Prosthetic Hand*.

Lee, Jeeun et al. (2017). "Security analysis of end-to-end encryption in Telegram". In: *Proc. 34th Symposium on Cryptography and Information Security (SCIS 2017), Naha, Japan*.

Mazzara, Manuel et al. (2020). "Size matters: Microservices research and applications". In: *Microservices*. Springer, pp. 29–42.

McDonald, John H (2014). "Kruskal–Wallis test". In: *Handbook of biological statistics*, pp. 157–164.

Millán, J. D. R. et al. (2010). “Combining Brain-Computer Interfaces and Assistive Technologies: State-of-the-Art and Challenges”. In: *Frontiers in neuroscience* 4. 161[PII], p. 161. issn: 1662-453X. doi: `10.3389/fnins.2010.00161`. url: `https://www.ncbi.nlm.nih.gov/pubmed/20877434`.

Nakamoto, Satoshi (2008). *Bitcoin: A peer-to-peer electronic cash system*. Tech. rep.

Nir, Yoav, Simon Josefsson, and Manuel Pegourie-Gonnard (2018). “Elliptic curve cryptography (ecc) cipher suites for transport layer security (tls) versions 1.2 and earlier”. In: *Internet Requests for Comments, RFC Editor, RFC* 8422.

Obondo, Barry (2018). *Lightning Network (beta) launched on bitcoin mainnet for the first time*. url: `http://theblockpro.com/cryptocurrency/bitcoin/lightning-network-beta-launched-bitcoin-mainnet-first-time/`.

O’Neill, Melissa E (2014). “PCG: A family of simple fast space-efficient statistically good algorithms for random number generation”. In: *ACM Transactions on Mathematical Software*.

Osterwalder, Alexander and Yves Pigneur (2013). *Business model generation a handbook for visionaries, game changers, and challengers*. Wiley & Sons.

Poon, Joseph and Thaddeus Dryja (2016). *The bitcoin lightning network: Scalable off-chain instant payments*.

Popov, Serguei (2016). “The tangle”. In: *cit. on*, p. 131.

Prince Sales, Tiago et al. (2017). “An Ontological Analysis of Value Propositions”. In: doi: `10.1109/EDOC.2017.32`.

Raval, Siraj (2016). *Decentralized applications: harnessing Bitcoin’s blockchain technology*. " O’Reilly Media, Inc."

Reeve, Amy, Eve Simcox, and Doug Turnbull (2014). “Ageing and Parkinsons disease: Why is advancing age the biggest risk factor?” In: *Ageing Research Reviews* 14, pp. 19–30. doi: `10.1016/j.arr.2014.01.004`.

Rescorla, Eric and Tim Dierks (2018). “The transport layer security (TLS) protocol version 1.3”. In:

Rhea, Sean et al. (2005). “OpenDHT: a public DHT service and its uses”. In: *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 73–84.

Ristic, Ivan (2013). *Deploying Forward Secrecy*. url: `https://blog.qualys.com/ssllabs/2013/06/25/ssl-labs-deploying-forward-secrecy`.

Rosello, Pol, Pamela Toman, and Nipun Agarwala (2017). “End-to-end neural networks for subvocal speech recognition”. In:

Saaty, Thomas L (1988). “What is the analytic hierarchy process?” In: *Mathematical models for decision support*. Springer, pp. 109–121.

– (2008). “Decision making with the analytic hierarchy process”. In: *International journal of services sciences* 1.1, pp. 83–98.

Schrittwieser, Sebastian et al. (2012). “Evaluating the Security of Smartphone Messaging Applications.” In: *NDSS*. Citeseer.

Sheffer, Yaron, Ralph Holz, and Peter Saint-Andre (2015). “Recommendations for secure use of transport layer security (tls) and datagram transport layer security (dtls)”. In: *RFC 7525*. RFC.

Sheskin, David J (2020). *Handbook of parametric and nonparametric statistical procedures*. pg. 46. crc Press.

Shi, Zhenqing et al. (2012). “Improved key recovery attacks on reduced-round salsa20 and chacha”. In: *International Conference on Information Security and Cryptology*. Springer, pp. 337–351.

Sigman, Ben (2019). *Sense.Chat Tech - Whitepaper Part 1*. url: `https://medium.com/sensechat/sense-chat-tech-c91295caf632`.

Snedecor, George W and William G Cochran (1989). "Statistical Methods, eight edition". In: *Iowa state University press, Ames, Iowa*.

Stephanidis, Constantine and Margherita Antona, eds. (2020). *HCI International 2020 - Posters*. Vol. 1224. Communications in Computer and Information Science. HCI International. Springer. isbn: 978-3-030-50726-8. doi: `10.1007/978-3-030-50726-8`.

Tatum, William, Benedetto Diciaccio, and Kirsten Yelvington (2016). "Cortical processing during smartphone text messaging". In: *Epilepsy Behavior* 59, pp. 117–121. doi: `10.1016/j.yebeh.2016.03.018`.

Tencent (2019). *Tencent Q3 2019 Results, page 4*. url: `https://cdc-tencent-com-1258344706.image.myqcloud.com/uploads/2019/11/13/8b98062831f2f28d9cb4616222a4d3c3.pdf`.

Thakkar, Parth, Senthil Nathan, and Balaji Viswanathan (2018). "Performance benchmarking and optimizing hyperledger fabric blockchain platform". In: *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, pp. 264–276.

Tikhomirov, Sergei (2017). "Ethereum: state of knowledge and research perspectives". In: *International Symposium on Foundations and Practice of Security*. Springer, pp. 206–221.

Turland, Connor (2018). *Summary of Holochain Architecture*. url: `https://github.com/holochain/holochain-proto/wiki`.

Ulaga, Wolfgang and Andreas Eggert (2006). "Relationship Value and Relationship Quality: Broadening the Nomological Network of Business-to-Business Relationships". In: *European Journal of Marketing* 40, pp. 311–327. doi: `10.1108/03090560610648075`.

Verna, Allee (2008). "Value network analysis and value conversion of tangible and intangible assets". In: 9.1, pp. 5–24. issn: 1469-1930. doi: `10.1108/14691930810845777`. url: `https://doi.org/10.1108/14691930810845777`.

Walls, Craig (2016). *Spring Boot in action*. Manning Publications.

Wand, Michael, Matthias Janke, and Tanja Schultz (2014). "The EMG-UKA Corpus for Electromyographic Speech Processing". In:

Welch, Chris (2016). *Google Keyboard for Android now has a one-handed mode*. url: `https://www.theverge.com/2016/5/2/11568836/google-keyboard-android-app-one-handed-mode-new-features`.

Witzig, Pascal and Victoriya Salomon (2018). *Cutting out the middleman: a case study of blockchain-induced reconfigurations in the Swiss Financial Services Industry*. Tech. rep. Université de Neuchâtel.

Wood, Gavin et al. (2014). "Ethereum: A secure decentralised generalised transaction ledger". In: *Ethereum project yellow paper* 151.2014, pp. 1–32.

Wright, Franklin Pierce (2010). *EmoChat: Emotional instant messaging with the EPOC headset*. University of Maryland, Baltimore County.

Wu, Caesar, Rajkumar Buyya, and Kotagiri Ramamohanarao (2018). "Cloud Computing Market Segmentation". In: *ICSOFT*, pp. 922–931.

Wu, Di, Ye Tian, and Kam-Wing Ng (2006). "Analytical study on improving DHT lookup performance under churn". In: *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*. IEEE, pp. 249–258.

Xu, Ling et al. (2019). "Task Assignment Algorithm Based on Trust in Volunteer Computing Platforms". In: *Information* 10, p. 244. doi: `10.3390/info10070244`.

Zhang, Hui, Ashish Goel, and Ramesh Govindan (2003). "Incrementally improving lookup latency in distributed hash table systems". In: *ACM SIGMETRICS Performance Evaluation Review*. Vol. 31. 1. ACM, pp. 114–125.

Zheng, Zibin et al. (2017). "An overview of blockchain technology: Architecture, consensus, and future trends". In: *2017 IEEE international congress on big data (BigData congress)*. IEEE, pp. 557–564.

Ziegler-Graham, Kathryn et al. (2008). "Estimating the prevalence of limb loss in the United States: 2005 to 2050". In: *Archives of physical medicine and rehabilitation* 89.3, pp. 422–429.

Appendix A

Meetings

Table A.1: Table showcasing a briefing of each meeting with the supervisors and experts in different fields.

Date	Briefing	Participants
31/10/2019	Overall objectives of the project Guidance in project formalization	Fábio Lourenço Luís Arteiro Dr. Paula Escudeiro Dr. Carlos Ferreira
09/12/2019	Discussion of decentralized approaches	Fábio Lourenço Luís Arteiro Dr. Carlos Ferreira Dr. Miguel Areias (FCUP/INESC TEC)
10/02/2020	Report submission P1 and project progress update	Fábio Lourenço Luís Arteiro Dr. Carlos Ferreira
23/03/2020	Progress report and implementation details	Fábio Lourenço Luís Arteiro Dr. Carlos Ferreira Dr. Paula Escudeiro

Appendix B

IHCI Paper Submission

SYNTHETIC TELEPATHY ON DECENTRALIZED MESSAGING APPLICATIONS

ABSTRACT

Peer-to-peer communication support for individuals with hand/arm impediments is currently scarce and poorly developed, with these users having heightened difficulty to utilize current mainstream applications. Furthermore, concerns for privacy and authorship of data has gained relevancy throughout the years. This paper outlines a new approach on human-machine interaction that is inclusive to people with these disabilities, with additional support for the vocally impaired by using an electroencephalography headset and electromyography surface electrodes respectively, all on a decentralized application that is developed to guarantee privacy on communication between users.

KEYWORDS

Brain-computer interface, silent-speech recognition, peer-to-peer communication, decentralization, distributed hash table

1. INTRODUCTION

With the advent of messaging applications, there is an apparent lack of support for disabled individuals to communicate on these platforms and to interact efficiently with other people. There has been significant progress addressing such issues mainly through electroencephalography (EEG), capable of recording brain activity that can be translated into actions (Vanacker *et al.*, 2007). Similarly, electromyography (EMG) can measure activity from muscles responsible for speech, enabling silent speech recognition (SSR) (Rosello, Toman and Agarwala, 2017).

Moreover, as centralized systems have come into scrutiny regarding privacy and security (Bouhnik, Deshen and Gan, 2014), development of alternative decentralized solutions have increased, a movement pioneered by Bitcoin (Nakamoto, 2008) that culminated in the blockchain technology. The latter has shown tremendous potential and applicability on a wide range of scenarios, boosted by the introduction of smart contracts by Ethereum (Wood and others, 2014), going beyond currency exchange. The surge of this technology paved the way for variants that maintain blockchain's main advantages but applied to different scenarios, having set a precedent and strides for decentralization across many fields (Wan *et al.*, 2017).

This solution’s main purpose is to provide a new approach for communication with support for users with hand/arm movements impairments, supported by contactless interaction with the device, where SSR is used in lieu of conventional speech-to-text features, consequently allowing oneself with speech disorders to also input text without clearly speaking or manually inserting it. It aims for the exchange of messages within a community where data privacy and integrity are crucial and each data is cryptographically stored with partial consensus, making it resilient against central points of failure with no scaling limits.

1.2 Related work

Since this project is a messaging platform, the text input is a key feature and it should be inclusive for people with disabilities. Experiments have been made using EEG to input text using virtual keyboards, like allowing a user to make binary decisions, iteratively splitting a keyboard in half until one letter remains, having a spelling rate of about 0.5 char/min (Birbaumer *et al.*, 1999). Another approach, which instead of binary choices, allows the selection of one between six hexagons, enabling the input of about 7 chars/min (Williamson *et al.*, 2009). Besides this, another research was conducted using a matrix of 6 by 6 independent cells, reaching 7.8 chars/min (Donchin, Spencer and Wijesinghe, 2000). Although input speed is not the primary focus of these projects, it must be user-friendly for instant messaging to be conceivable. Another studied approach to allow text input was through a silent speech interface (SSI), which also enables input by people with speech impairments (e.g. laryngectomy). Researches on this field still have a high word error rate (WER), as some achieved around 50% WER for their SSR attempt, for a vocabulary of 108 words (Wand, Janke and Schultz, 2014). Another group proposed working in phoneme recognition, using the same dataset. Despite the versatility, it only attained around 30% char accuracy (Rosello, Toman and Agarwala, 2017). Studies for this approach do not yield great results for a wide-range vocabulary, although appearing to be more promising when compared to EEG-based interfaces.

Regarding messaging platforms and their dedicated backend, there have been attempts to develop decentralized messaging applications that aim to address many issues present in centralized applications that are usually perceived as secure (e.g. Telegram) and allow decentralized authoritative messaging (Leavy and Ryan, 2018). Applications that follow the classic blockchain approach towards the messaging panorama include *Adamant*, which showcases an example of a fully anonymous, private and secure messaging between peers. Relying on its own token that is used for each transaction, such application relies on having no personal or sensitive data being extracted from the user (Evgenov *et al.*, 2017). A decentralized solution which follows a distributed hash table (DHT) approach is Jami, where the DHT is used to locate peer addresses and establish peer-to-peer connection without any personal sensitive data associated, with undelivered messages being stored locally.

2. OUR SOLUTION

The proposed solution encompasses SSI and BCI for the interaction of an application that partakes in a decentralized ecosystem that is user-centric, with high throughput and maintaining a bottom-line of privacy of data and message exchange. It envisions a way for people with arm/hand and speech impairments to be able to communicate with anyone. It is, therefore, divided into two distinctive counterparts: human-machine interaction and decentralized communication between peers.

2.1 Human-Computer Interaction

Aiming for a hands-free-controlled application with text input by silent speech, this project purposes a synthetic telepathy-based solution. Not only does it allow individuals with speech problems and arm motor disabilities to interact with the platform, but also creates a multitude of use cases for people without these limitations (e.g. privacy and mitigation of background noises when using hands-free mode in public since it uses silent speech as an alternative to the common text-to-speech and speech-to-text solutions; multitasking in everyday tasks while navigating with mental commands).

For the BCI component in this project, the Emotiv EPOC+ is being used for EEG and the Emotiv BCI software for the training and classification of the data gathered from a user. This device and related software are of great advantage since it speeds-up the project development cycle, considering it already has predefined mental commands for a subject to train, which can be applied in the user interface (UI) navigation. Considering the UI must be accessible and intuitive for both users with or without impairments using BCI and SSI or usual interfaces, a new design idea had to emerge to accomplish these goals.

Thus, besides from regular click/tap to interact, the user can also navigate through different chat rooms using four different commands: pull, push, left and right. Simulating a three-dimensional space, the pull and push commands pull closer to and push away from the user view, respectively, a whole screen related to a single chat room (Figure **1**). The remaining left and right commands are used to slide through a carousel-type view with each chat room, always fixing a chat room in the middle, until the user pulls the one selected (Figure 1).

Regarding SSI, the first samples are being gathered for the prototype of this application. For data acquisition, a pair of EMG surface electrodes are being used, placed on the throat of the user, measuring myoelectric activity using bipolar configuration. For this first implementation, only data from a set of predefined words will be included, for validation and testing of different approaches. After feature extraction, several machine learning classification algorithms will be tested and compared for efficiency.

Other approaches more diverse than classification for only a small set of words are being studied for future work. For this feature, it is considered and tailored to the English language in order to reach to a broader audience.

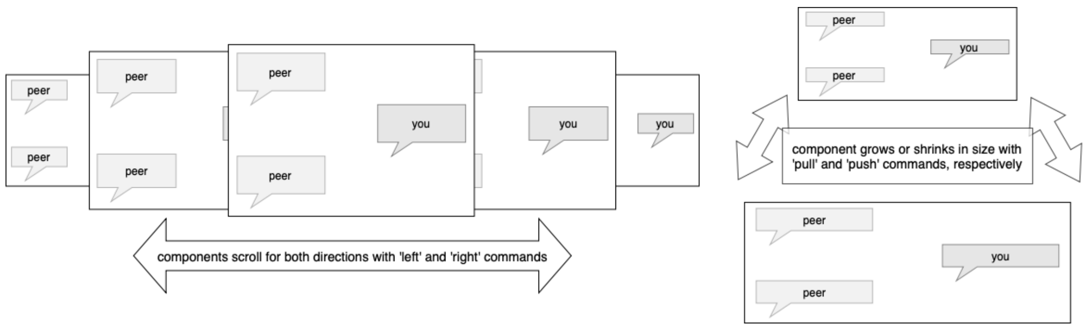

peer
you
components scroll for both directions with 'left' and 'right' commands
component grows or shrinks in size with 'pull' and 'push' commands, respectively

Figure 1. Navigation screens compatible with mental commands from the EEG headset.

2.2 Decentralized backend

The decentralized system is user-centric, that is, each user runs a bundle of the backend and frontend, has their own identity and their private and public shared data. Being on the same encrypted peer-to-peer network means each user can communicate with each other. Each node within the network runs the same application and, subsequently, the same logic and rules in which data exchanged must abide by. Each peer will validate their own data but also other's, making the system tamper-resistant and have data integrity.

Data resilience is also important in any messaging platform. That is, data must not get lost when users go offline. To address this issue, each piece of public data is witnessed, validated according to the system's logic that is present in every user and redundantly stored by a random selection of peers.

Users communicate with each other about data through gossiping. The latter is crucial for the well-being of the system because peers will gossip with each other to detect invalid data, such evidence of its author and take action to deal with malice users. This is a direct application of peer data replication and validation.

Inclusively, the system shares similarities with Jami in a way that it is based on a distributed hash table (DHT). This table, contrary to blockchain-based applications, is not replicated in its entirety each node. This decision of using a DHT is made in order to attain a higher TPS (transaction per second), throughput and lookup speed, in which each peer stores a segment/shard of the table.

2.2.1 User record

Every user has their own chain signed with their rightful private key. This chain can be thought of as a record of all the messages/data the user has produced and exchanged within the app and is stored on their own device. Each user has their own digital identifier using the public/private key pair cryptographic method. The combination of these two keys is imperative for the user online identity and for communication with other peers. The public key is shared with other participants who use it validate data and know who created it. Each user proves authorship of their data through digital signatures stemming from their private key. Any tampering is promptly detected by simply using the correspondent public key to verify the signature. If such validation fails, the data is considered invalid and this finding is gossiped and broadcasted to the rest of the network.

The DHT is where all public data resides in. The word public is used very loosely. Every entry is hashed so as to make it untraceable, so it is not necessarily public as in it is available for everyone to see. Each and every entry that resides within the DHT are essentially user chain entries that are merely marked as public, hashed with the peer's address and signed with its private key.

2.2.2 Validation and direct peer-to-peer communication

The security, state and integrity of the whole system is maintained by the logic that is replicated amongst every node. Data integrity is ruled by these rules which, in turn, uphold the security of the system. Besides normal messaging, users that want to share secret or sensitive data directly with another user that can later be destroyed can use direct node-to-node messaging through an end-to-end encrypted channel by resolving the specific agent's IP address.

The application is built on top of the Holochain framework, who handles a deal of network concerns and cryptography. Additionally, Holochain provides a way for the system to be interoperable with other networks build on the same Holochain network, making the application have more expandable prospects.

2.3 Evaluation

BCI evaluation process will consist of usability tests to gather feedback on how the platform improves their communication and overall navigation compared to similar apps. The factors that will be analysed are comfort, interaction speed, accuracy, and learning curve, and each will be evaluated on a typical five-level Likert scale. For SSI, the data sets used for each model created will be divided into smaller sets of training, test and cross-validation sets. These test sets will examine the accuracy of classification, improving our models to obtain the lowest error rate possible.

The Holochain framework provides support for both unit and functional tests. These will be useful to provide a measurement of code coverage and reliability of the written code at this level. End-to-end tests will ultimately leverage a way to measure performance as the solution scales. The throughput is tested through stress tests and assess if the former is consistent. The resilience of the system is tested through man-in-the-middle attacks, at cryptography level, and denial-of-service attempts.

3. CONCLUSION

The showcased solution mitigates issues from centralized communication, regardless of impairment status of individuals, applying methods of machine learning coupled with SSI and BCI. Moreover, it depicts a new proposal on the decentralized messaging platform, either from human-machine interaction standpoint or the distributed ecosystem counterpart. Taking a more user-centric approach towards the problem, utilizing a distributed hash table with partial consensus instead of data replication in its entirety with global consensus is more adequate to the messaging panorama. These design choices have a direct effect on performance and throughput of the system; more transactions per second can occur whilst the exchange of data is still secure and private and resilient against tampering attacks, while removing scalability limitations.

The limitations of this application comprise of the number of options for navigation and control over the application and the limited vocabulary and language in which the user can utilize subvocalization. Furthermore, the application is yet to support file exchange, although this issue could be addressed through a separate DHT or parallel channels.

REFERENCES

Birbaumer, N. *et al.* (1999) 'A spelling device for the paralysed', *Nature*, 398, pp. 297–298. doi: 10.1038/18581.

Bouhnik, D., Deshen, M. and Gan, R. (2014) 'WhatsApp goes to school: Mobile instant messaging between teachers and students', *Journal of Information Technology Education: Research*, 13(1), pp. 217–231.

Donchin, E., Spencer, K. M. and Wijesinghe, R. (2000) 'The mental prosthesis: assessing the speed of a P300-based brain-computer interface', *IEEE Transactions on Rehabilitation Engineering*, 8(2), pp. 174–179. doi: 10.1109/86.847808.

Evgenov, P. *et al.* (2017) *White paper: Adamant messaging application*. Available at: https://adamant.im/whitepaper/adamant-whitepaper-en.pdf.

Leavy, T. M. and Ryan, G. (2018) 'Decentralized authoritative messaging'. Google Patents.

Nakamoto, S. (2008) 'Bitcoin: A peer-to-peer electronic cash system'.

Rosello, P., Toman, P. and Agarwala, N. (2017) 'End-to-end neural networks for subvocal speech recognition'.

Vanacker, G. *et al.* (2007) 'Context-Based Filtering for Assisted Brain-Actuated Wheelchair Driving', *Computational intelligence and neuroscience*, 2007, p. 25130. doi: 10.1155/2007/25130.

Wan, Z. *et al.* (2017) 'BKI: Towards accountable and decentralized public-key infrastructure with blockchain', in *International Conference on Security and Privacy in Communication Systems*, pp. 644–658.

Wand, M., Janke, M. and Schultz, T. (2014) 'The EMG-UKA Corpus for Electromyographic Speech Processing', in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*.

Williamson, J. *et al.* (2009) 'Designing for uncertain, asymmetric control: Interaction design for brain–computer interfaces', *International Journal of Human-Computer Studies*, 67, pp. 827–841. doi: 10.1016/j.ijhcs.2009.05.009.

Wood, G. and others (2014) 'Ethereum: A secure decentralised generalised transaction ledger', *Ethereum project yellow paper*, 151(2014), pp. 1–32.

Appendix C

HCII Paper Submission

Brain-Computer Interaction and Silent Speech Recognition on Decentralized Messaging Applications

Luís Arteiro(✉), Fábio Lourenço, Paula Escudeiro, and Carlos Ferreira

Polytechnic of Porto - School of Engineering (ISEP), 4200-072 Porto, Portugal
`{1150625, 1150434, pmo, cgf}@isep.ipp.com`

Abstract. Peer-to-peer communication has increasingly gained prevalence in people's daily lives, with its widespread adoption being catalysed by technological advances. Although there have been strides for the inclusion of disabled individuals to ease communication between peers, people who suffer hand/arm impairments have scarce support in regular mainstream applications to efficiently communicate privately with other individuals. Additionally, as centralized systems have come into scrutiny regarding privacy and security, development of alternative, decentralized solutions has increased, a movement pioneered by Bitcoin that culminated on the blockchain technology and its variants.
Within the inclusivity paradigm, this paper aims to showcase an alternative on human-computer interaction with support for the aforementioned individuals, through the use of an electroencephalography headset and electromyography surface electrodes, for application navigation and text input purposes respectively. Users of the application are inserted in a decentralized system that is designed for secure communication and exchange of data between peers that are both resilient to tampering attacks and central points of failure, with no long-term restrictions regarding scalability prospects. Therefore, being composed of a silent speech and brain-computer interface, users can communicate with other peers, regardless of disability status, with no physical contact with the device. Users utilize a specific user interface design that supports such interaction, doing so securely on a decentralized network that is based on a distributed hash table for better lookup, insert and deletion of data performance. This project is still in early stages of development, having successfully been developed a functional prototype on a closed, testing environment.

Keywords: Brain-computer interface · Silent speech recognition · Peer-to-peer communication · Decentralization · Distributed hash table.

1 Introduction

With the advent of messaging applications, there is an apparent lack of support for disabled individuals to communicate on these platforms and use such applications to interact efficiently with other people. There has been significant progress addressing such issues mainly through electroencephalography (EEG) sensors capable of recording brain activity that, powered by machine learning, can translate it into actions [6]. Similarly, electromyography (EMG) electrodes can measure activity from muscle responsible for speech, enabling silent speech recognition (SSR) [11], that can be used for private and silent text input into the application.

Additionally, privacy and security of data within centralized systems have been under the spotlight, especially on mobile applications [2], contributing to the rise of the development of alternative, decentralized solutions have increased, a movement first created by Bitcoin [8] that effectively stapled blockchain as a growing technology. The latter has shown tremendous potential and applicability on a wide range of scenarios, boosted by the introduction of smart contracts by Ethereum [13], going beyond currency exchange. These assets allow trustless, secure, peer-to-peer value transfer that can easily be applied to the messaging paradigm, thus allowing immutable, secure messaging between nodes. The surge of this technology paved the way for variants that maintain blockchain's main advantages but change their scope. These have set a precedent and strides for decentralization across many fields [10].

2 Related Work

Regarding brain-computer interfaces (BCI) using non-invasive EEG, research projects have emerged allowing users to control virtual keyboards and mouses, empowering individuals with motor disabilities in their interaction with computers. Regarding app's navigation using this approach, there are experiments on cursor movement that allow a user to move it by a BCI. Tests with different approaches within movements for one or two dimensions showed an accuracy of approximately 70% [5]. A similar project utilizes a hybrid near-infrared spectroscopy-electroencephalography technique trying to extract four different types of brain signals, decoding them into direction symbols, namely, "forward", "backward", "left", and "right". The classification accuracy for "left" and "right" commands was 94.7%, for "forward" was 80.2% and 83.6% for "backward" [6]. Later in this document, it will be shown that our system uses a similar approach for the set of commands used for navigation in the platform.

Since this project concerns a messaging platform, text input is a key feature that must offer support for the target audience - disabled subjects. Experiments have been made using EEG to input text using virtual keyboards, allowing a user to make binary decisions and iteratively splitting a keyboard in half until one letter remains, having a spelling rate of about 0.5 char/min [1]. Another approach, which instead of doing binary choices, allows the selection of one between six hexagons, enabling the input of about 7 chars/min [12]. Although input speed is not the central focus of these projects, it must be user-friendly for instant messaging to be conceivable. Yet another studied approach to allow text input was through a silent speech interface (SSI), which also enables input by people with speech impairments (e.g. laryngectomy). Researches on this field still have a high word error rate (WER), as some have achieved around 50% WER for their SSR attempt, using a vocabulary of 108 words through the EMG-UKA corpus [11]. Another group proposed another solution, using the same corpus, attaining around 30% char accuracy [9]. Studies for this approach do not yield satisfactory results for a wide-range vocabulary, although appearing to be more promising when compared to EEG-based interfaces.

Regarding messaging platforms and their dedicated backend, there have been attempts to develop decentralized messaging applications that aim to address many issues present in centralized applications that are usually perceived as secure (e.g. Telegram) and allow decentralized authoritative messaging [7]. Applications that follow the classic blockchain approach towards the messaging panorama include Adamant, which showcases an example of a fully anonymous, private and secure messaging between peers. Relying on its own token that is used for each transaction, such application relies on having no personal or sensitive data being extracted from the user [4]. A decentralized solution which follows a distributed hash table (DHT) approach is Jami, where the DHT is used to locate peer addresses and establish peer-to-peer connection without any personal sensitive data associated, with undelivered messages being stored locally.

3 Our Solution

The proposed solution encompasses a new approach to the interaction of an application that partakes in a decentralized ecosystem that is user-centric, with high throughput and maintaining a bottom-line of privacy of data and message exchange. It envisions a way for people with arm/hand to be able to communicate with anyone. It is, therefore, divided into two distinctive counterparts: human-machine interaction, through BCI and SSI, and decentralized communication between peers. This is further outlined in Figure 1.

3.1 Human-Machine Interaction

Aiming towards a hands-free controlled application with text input through silent speech, this project proposes a synthetic telepathy-based solution to messaging. It allows individuals with arm motor disabilities to interact with the platform and also creates a multitude of use cases for people without these limitations (e.g. privacy and mitigation of background noises when using speech-to-text features in public; multitasking while navigating with mental commands).

Brain-Computer Interface For the BCI component in this project, Emotiv EPOC+ is being used for EEG recordings and the Emotiv BCI software for training and classification of the data gathered from the user. This device and related software are of great advantage since it accelerates the overall project development considering it already has predefined mental commands for a subject to train, which can be applied for the user interface (UI) navigation. Considering the UI must be accessible and intuitive for both users with or without

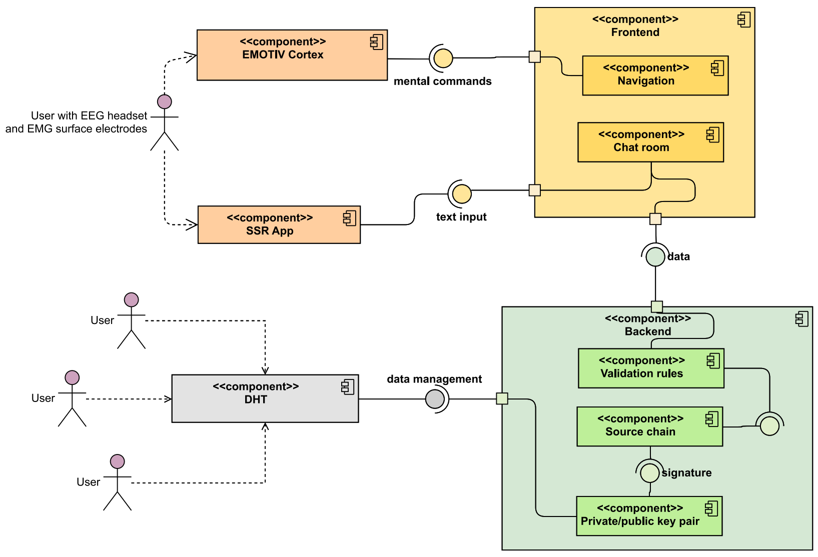


Fig. 1. Overall conceptual view of the platform, outlining the interaction between the user and interface, as well as showcasing the decentralized system where the user is inserted into (DLT - Distributed Ledger Technology), outlining the different components that make up the backend.

impairments, a new design idea had to emerge to accomplish these goals. Thus, besides from regular click/tap to interact, the user can also navigate through different chat rooms using four different commands, ”pull”, ”push”, ”left”, and ”right”. Simulating a two-dimensional space, the ”pull” and ”push” commands pull closer to and push away from the user view, respectively (see Figure 2), and the remaining ”left” and ”right” commands are used to slide through a carousel-type view with each chat room, always fixing a chat room in the middle, until the user pulls the one selected (see Figure 3). Furthermore, more commands may be added for others minor tasks, or these already implemented commands can correspond to multiple actions depending on the state of the application or feature that is being used.

Silent Speech Interface Regarding SSI, it is being used the aforementioned EMG-UKA corpus for implementation and testing of the system in its early stages. Approaches for this system are still being studied, and the first experiments are being conducted on a session-dependent model attempting to classify phones (speech sounds) from EMG data. Further developments will allow users to input full words into the system. In order to reach a broader audience and using the exiting corpus, the system is tailored for the English language.

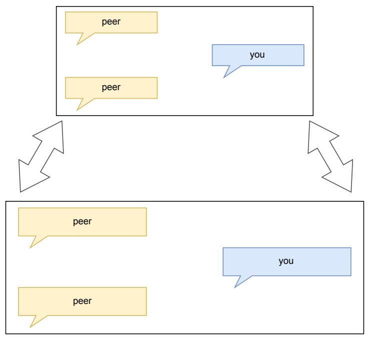


Fig. 2. "Pull" and "push" commands effect the UI, allowing the user to enter and leave chatrooms.

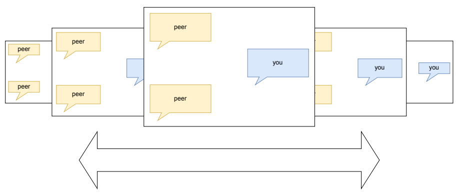


Fig. 3. The user can slide right or left and navigate through his/her chatrooms through "right" and "left" mental commands.

3.2 Decentralized Backend

As it was aforementioned, individuals who make use of the platform, whether these suffer from any impairment or not, will partake in a decentralized ecosystem that is intended to provide secure peer-to-peer communication with data integrity. The system is agent-centric, meaning each user runs a copy of the backend, have their own identity and their private and public shared data. Being on the same encrypted peer-to-peer network entails each user can communicate with each other directly to maintain its integrity. Having each peer hold the same application bundle with the respective logic, it is possible to verify other peer's transactions and data created. Each data has proof of authorship (i.e. a signature). Since every data is recorded and validated by the peers within the system, it is tamper-resistant - this showcases data integrity.

Data resilience is also important in any messaging platform. That is, data must not get lost when users go offline. To address this issue, each piece of public data is witnessed, validated according to the system's logic that is present in every user and stored by a random selection of peers. This makes it that the community and cooperating parties detect invalid data, gossip such evidence of harmful agents and take action to deal with malice data/users. This is a synonym to peer data replication and data validation.

The system shares similarities with Jami in a way that it is based on a distributed hash table (DHT). This table, contrary to blockchain-based applications, is not replicated in each node. To attain a higher transactions per second (TPS), throughput and lookup, each peer stores a segment of the table. This DHT is where the data resides and, in cases where users go offline, is stored to later be retrieved by the recipient. The implementation of this table is based on top of the Holochain framework, allowing for easier data replication and validation between peers. As more users join in the network, more computational power is contributed to the environmnent and data replication is more redundant, allowing the system to scale as more users partake within the system.

User Record Each user has their own chain signed with their rightful private key. This chain can be thought of as a record of all the messages/data the user has produced and exchanged within the app and is stored on their own device. Each individual has their own digital identifier using the public/private key pair cryptographic method. The combination of these two keys is imperative for the user online identity and for communication with other peers. The public key is shared with other participants. Each user proves authorship of their data through digital signatures stemming from their private key. Any tampering is promptly detected by simply using the correspondent public key to verify the signature. If such validation fails, the data is considered invalid and this finding is gossiped and broadcasted to the rest of the network. The distributed hash table is where all public data resides in. The word ”public” is used very loosely. Every entry is hashed so as to make it untraceable so it is not necessarily public as in it is available for everyone to see. Each and every entry that resides within the DHT are essentially user chain entries that are merely marked as public.

Validation and Direct Peer-To-Peer Communication The security, state and integrity of the whole system is maintained by logic that is hard-coded and bundled in every node. Data integrity is ruled by these rules which, in turn, uphold the security of the system. All data, whether it is in the user private chain or in the public DHT, is validated according to these rules. Normal messaging occurs when two users are online and is made through direct peer-to-peer connection in an end-to-end encrypted channel by resolving each agent's IP address. This method only works when both peers are online. To circumvent this, private messages are encrypted using the recipient's public key and published to the DHT, to later be retrieved by the recipient when they are back online.

Most of these networking protocols and distributed computing scenarios are handled by the Holochain framework, from which the architecture is based from. Not only it resolves many concerns regarding encryption but also allows for interoperatibility with other systems that take part within the Holochain network.

4 Preliminary Evaluation

As this is a work-in-progress and with no tangible results yet yielded, it is possible to define an assessment strategy for it. BCI evaluation process will consist of usability tests performed by hand/arm impaired individuals to gather feedback on how the platform improves the user's communication and overall navigation compared with similar apps. Prior to testing, the subjects ought to train the BCI headset and tailor it to their mental activity for the model to recognize the commands from a specific user. Afterwards, subjects will be asked to perform a planned set of actions that entails navigating in the platform and interact with it. Lastly, a survey will be conducted following the System Usability Scale [3], with 10 items answered in a degree of agreement or disagreement with each statement on typical five-level Likert scale. For SSI, the datasets used for each model created will be divided into smaller training and testing sets, allowing to evaluate their accuracy, aiming to obtain the lowest WER possible.

The Holochain framework provides support for both unit and functional tests. These are useful to provide a measurement of code coverage and respective reliability. End-to-end tests will ultimately leverage a way to measure performance as the solution scales. The throughput is tracked as the number of users grows and its consistency is assessed. Early results have showcased successful data creation on the public DHT on a small number of users (five) with a step duration of 1000 miliseconds, where the period is halved at every stage. In this, the stress test is conducted indefinitely and increases pace at every stage. Additionally, a small increase of throughput was noticed when more users joined the network, from two peers to five. These experiments have not shown any sensible traces of data corruption nor delay, although more tests on a higher scale are be needed to form more conclusive results.

5 Conclusion

This project mitigates communication barriers between subjects with or without hand/arm impairments and allows the inclusion of everyone into a single messaging system, using a new strategy to this type of platform, applying a synthetic telepathy approach for the interaction with it. Additionally, if the project produces positive results, the range of applicability for both BCI and SSI approaches can extend beyond the messaging paradigm.

The system that was outlined showcases a new proposal on the decentralized messaging platform, either from human-machine interaction standpoint or the distributed ecosystem counterpart. Following an user-centric approach towards the problem, utilizing a DHT with ad hoc peer consensus instead of data replication in its entirety with global consensus is more adequate to the messaging panorama. These design choices have a direct effect on performance and throughput of the system: more TPS can occur whilst the exchange of data is still secure, private and resilient against tampering attacks. This essentially means that scalability does not pose as a problem on a long-term perspective, unlike many solutions that are based on the classic blockchain concept.

6 Future Work

With the drafted architecture and design choices, there is still room for improvement for future project prospects so as to be providable to a broader audience and maintain a performance bottom line. On the human-machine interface counterpart, for the BCI component, other commands can be mapped and used by the platform, giving the user more options for navigation and control over the application, for a wider set of features, but keeping the interaction as intuitive as possible. SSI-wise, aiming for a more natural and reliable text input system for any user, efforts will be made for a session and user-independent system, enabling the its usage by multiple users and sessions without much accuracy fluctuations.

One of the objectives regarding the decentralized backend would be to go beyond message exchange and also extend to file sharing. Approaches could be developed similar to InterPlanetary File System (IPFS). However, this topic is sensitive since file storage takes up more space than common messages. Replication of data between peers would need to be addressed differently, perhaps through parallel channel or DHT altogether. Furthermore, voice calling could be another feature present within the application that could co-exist and be made through node-to-node channels.

References

1. Birbaumer, N., Ghanayim, N., Hinterberger, T., Iversen, I., Kotchoubey, B., Kübler, A., Perelmouter, J., Taub, E., Flor, H.: A spelling device for the paralysed. Nature **398**, 297–8 (04 1999). https://doi.org/10.1038/18581
2. Bouhnik, D., Deshen, M., Gan, R.: Whatsapp goes to school: Mobile instant messaging between teachers and students. Journal of Information Technology Education: Research **13**(1), 217–231 (2014)
3. Brooke, J., et al.: Sus-a quick and dirty usability scale. Usability evaluation in industry **189**(194), 4–7 (1996)
4. Evgenov, P., Lebedev, A., Soloduhin, D., Pikhtovnikov, M., Vorobev, A., Lebedev, S., Anokhov, P.: White paper: Adamant messaging application. Tech. rep. (2017), https://adamant.im/whitepaper/adamant-whitepaper-en.pdf
5. Fabiani, G.E., McFarland, D.J., Wolpaw, J.R., Pfurtscheller, G.: Conversion of eeg activity into cursor movement by a brain-computer interface (bci). IEEE Transactions on Neural Systems and Rehabilitation Engineering **12**(3), 331–338 (Sep 2004). https://doi.org/10.1109/TNSRE.2004.834627
6. Khan, M.J., Hong, M.J., Hong, K.S.: Decoding of four movement directions using hybrid nirs-eeg brain-computer interface. Frontiers in Human Neuroscience **8**, 244 (2014). https://doi.org/10.3389/fnhum.2014.00244, https://www.frontiersin.org/article/10.3389/fnhum.2014.00244
7. Leavy, T.M., Ryan, G.: Decentralized authoritative messaging (Nov 27 2018), uS Patent 10,142,300
8. Nakamoto, S., et al.: Bitcoin: A peer-to-peer electronic cash system (2008), https://bitcoin.org/bitcoin.pdf
9. Rosello, P., Toman, P., Agarwala, N.: End-to-end neural networks for subvocal speech recognition (2017), http://www.pamelatoman.net/wp/wp-content/uploads/2018/06/subvocal_speech_recognition_paper.pdf, from Stanford University
10. Wan, Z., Guan, Z., Zhuo, F., Xian, H.: Bki: Towards accountable and decentralized public-key infrastructure with blockchain. In: International Conference on Security and Privacy in Communication Systems. pp. 644–658. Springer (2017)
11. Wand, M., Janke, M., Schultz, T.: The emg-uka corpus for electromyographic speech processing. In: INTERSPEECH-2014. pp. 1593–1597
12. Williamson, J., Murray-Smith, R., Blankertz, B., Krauledat, M., Müller, K.R.: Designing for uncertain, asymmetric control: Interaction design for brain–computer interfaces. International Journal of Human-Computer Studies **67**, 827–841 (10 2009). https://doi.org/10.1016/j.ijhcs.2009.05.009

13. Wood, G., et al.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**(2014), 1–32 (2014)

Appendix D

TNC Paper Submission

Synthetic Telepathy on Decentralized Messaging Applications

FTP TNC20 Lightning Talk

Luís Arteiro[1(✉)], Fábio Lourenço[1], Paula Escudeiro[1], Carlos Ferreira[1]

[1] Instituto Superior de Engenharia do Porto
1150625@isep.ipp.pt, 1150434@isep.ipp.pt, pmo@isep.ipp.pt, cgf@isep.ipp.pt

(✉) Presenter

Description. Peer-to-peer communication has been increasingly gaining prevalence in people's daily lives, with its widespread adoption being catalysed by technological advances. Although there have been strides for the inclusion of disabled individuals to ease communication between peers [1], people who suffer arm/hand impairments have little to no support in regular mainstream applications to efficiently communicate with other individuals. Additionally, as centralized systems have come into scrutiny [2] regarding privacy and security, development of alternative, decentralized solutions have increased, a movement pioneered by Bitcoin [3] that culminated on the blockchain technology and its respective variants.

Within the inclusivity paradigm, this presentation aims to showcase an alternative to conventional messaging software applications but also on human-computer interaction with support for users the aforementioned disabilities, something that is scarce on common messaging platforms. This is made possible through the use of an electroencephalography headset and electromyography surface electrodes, for application navigation and text input purposes respectively. Effectively, the former is responsible of monitoring electrical brain activity in a noninvasive manner. Such signals are collected and digitized and, after data processing and subsequent application of machine learning algorithms, patterns of this activity can be translated into action. On the other hand, electromyography electrodes will perform a similar procedure, with the exception of detecting muscle activity.

Users of the application will be inserted in a decentralized system that is designed for secure communication and exchange of data between peers that are both resilient to tampering attacks and central points of failure, and with no long-term restrictions regarding scalability prospects. Therefore, being composed of a silent speech and brain-computer interface, together under the umbrella term of 'synthetic telepathy', users are able to communicate with other peers, regardless of disability status, with no physical contact with the device. Users utilize a specific user interface design that supports such interaction, doing so securely on a decentralized network that is based on a distributed hash table for better lookup, insert and deletion of data performance, roughly based on the Holochain framework.

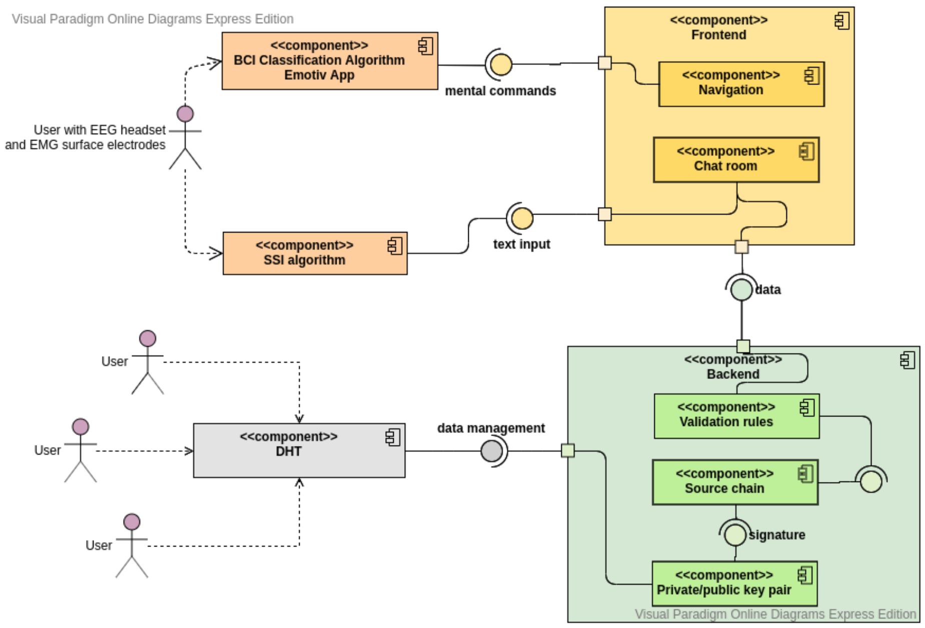
Keywords— Brain-computer interface, silent speech recognition, peer-to-peer communication, decentralization, distributed hash table

References

[1] Richard M Jackson. Audio-supported reading for students who are blind or visually impaired. *Wakefield, MA: National Center on Accessible Instructional Materials*, 2012.

[2] Dan Bouhnik, Mor Deshen, and R Gan. Mobile instant messaging between teachers and students. *Journal of Information Technology Education: Research*, 13(1):217–231, 2014.

[3] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.



Visual Paradigm Online Diagrams Express Edition
<<component>> BCI Classification Algorithm Emotiv App
mental commands
<<component>> Frontend
<<component>> Navigation
<<component>> Chat room
User with EEG headset and EMG surface electrodes
<<component>> SSI algorithm
text input
data
User
User
User
<<component>> DHT
data management
<<component>> Backend
<<component>> Validation rules
<<component>> Source chain
signature
<<component>> Private/public key pair
Visual Paradigm Online Diagrams Express Edition

Figure 1: High-granularity diagram showcasing how each component/concept interacts with each other across the application. The frontend and the backend counterpart both reside within each user's device. Each user has their own record of their data, all pieces of dat being chained; a private/public key pair and a set of validation rules that are present in every node within the network.