DEMEDIA – DECENTRALIZED SOCIAL MEDIA PROTOCOL

Perera B. B. S.

IT20254698

B.Sc. (Hons) Degree in Information Technology

Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

PEER-TO-PEER COMMUNICATION PROTOCOL

Perera B.S.S.

IT20254698

Dissertation submitted in partial fulfillment of the requirements for the Bachelor of

Science specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2023

DECLARATION

I declare that this is our own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature:

IT20254698	Perera B.S.S.	[illegible]

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

…………………………..   …………………………
Signature of the supervisor: Date

(Mr. Kavinga Yapa Abeywardena)

ABSTRACT

Social media has become widely used in modern society, and almost everyone uses it. It is now an essential part of how we live in the present. However, it is important to note that these social networks maintain on centralized servers to store user data. This approach has been shown to have several disadvantages. For instance, users have limited control over their data or the platform they use. This proposal presents a decentralized social network that utilizes the user's device as storage, while ensuring that data remains secure and authentic, with complete control over user data residing with the user.

Furthermore, the proposed solution consists of two central components such as the implementation of a user data decentralization protocol and the design and deployment of a client application that facilitates smooth communication with other network peers through this decentralized protocol. The user data decentralization protocol serves as the fundamental change, moving data distribution away from the centralized server model and toward a network of interconnected nodes or peers.
This protocol is designed to not only be used in the proposed social network, but also to be used in other decentralized systems, such as distributed applications, and other decentralized sharing platforms. RPC style is used to implement the protocol, and LibP2P is used as a base for the protocol which gives very robust and flexible P2P foundation for the P2P protocol. Finally, the proposed solution is evaluated by creating a demo social network application and testing the performance of the decentralized protocol against a traditional REST AP by creating benchmarks applications.

Keywords: *social media, decentralization, peer-to-peer, protocol, distributed systems*

ACKNOWLEDGEMENT

The success of this research is due to the efforts of my supervisor, Mr. Kavinga Yapa Abeywardena, as well as the rest of the team. Additionally, I'd like to express my gratitude to the Department of Information Technology at the Sri Lanka Institute of Information Technology, as well as to CDAP academics and staff, for enabling this research.

TABLE OF CONTENT

ABSTRACT.. iv
ACKNOWLEDGEMENT v
TABLE OF CONTENT vi
LIST OF FIGURES viii
LIST OF TABLES ix
LIST OF APPENDICES x
LIST OF ABBREVIATIONS i
1.0 INTRODUCTION 1
1.1 BACKGROUND LITERATURE 3
1.2.1. Skype 4
1.2.2. Gnutella 5
1.2.3. Kazaa 6
1.2.4. eMule 7
1.2.5. Torrent 8
1.2 RESEARCH GAP 10
1.3 RESEARCH PROBLEM 12
1.4 RESEARCH OBJECTIVES 13
1.4.1 Main Objective 13
1.4.2 Specific Objectives 13
2.0 METHODOLOGY 15
2.1 REQUIREMENT GATHERING 15
2.2 FEASIBILITY STUDY 15
2.2.1 Technical feasibility 15
2.2.2 Schedule feasibility 16
2.2.3 Economic feasibility 16
2.3 REQUIREMENT ANALYSIS 16
2.4 SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC) 17
2.5 PROPOSED SYSTEM DESIGN 18
2.5.1 SYSTEM OVERVIEW DIAGRAM (OVERALL) 18
2.5.2 System Overview Diagram (Individual) 20
2.6 COMMERCIALIZATION OF THE PROJECT 21
2.7 TESTING & IMPLEMENTATION 23
2.7.1 Background Establishment for Implementation 23

2.7.1.1 Programming Language Selection: Go Lang 23
2.7.1.2 Database System: PostgreSQL 24
2.7.1.3 Collaboration and Version Control: GitHub 24
2.7.1.4 Cloud Deployment: Amazon Web Services (AWS) 25
2.7.1.5 Containerization: Docker 26
2.7.2 Setting Up CI/CD Pipeline 28
2.7.2.1 Continuous Integration (CI) Pipeline 28
2.7.2.2 Continuous Deployment (CD) Pipeline 28
2.7.3 Development 30
2.8 TESTING 33
2.8.1 Unit Testing 33
2.8.2 Continuous Integration (CI) and Deployment (CD) 33
2.8.3 Smoke Testing 34
2.8.4 Validation using practical implementations. 35
2.8.4.1 Benchmarking with Infrastructure as Code (IaaC) Approach 35
3.0 RESULTS AND DISCUSSIONS 37
3.1 RESULTS 37
3.1.1 Demo Social Media Platform Results 37
3.1.2 Benchmarking Results 39
3.1.3 Results from overall practical implementations 39
3.2 RESEARCH FINDINGS 40
3.3 DISCUSSION 41
4.0 SUMMARY OF EACH STUDENT’S CONTRIBUTION 42
5.0 CONCLUSION 43
REFERENCES 44
GLOSSARY 47
APPENDICES 48

LIST OF FIGURES

Figure 1.2.1.1: Architecture of Skype 5
Figure 1.2.2.1: Architecture of Gnutella 6
Figure 1.2.3.1: Architecture of Kazaa 7
Figure 1.2.4.1: Architecture of eMule 7
Figure 1.2.5.1: Architecture of Torrent 8
Figure 1.2.5.2: Architecture of Torrent2 9
Figure 2.4.1: Software Development Life Cycle 17
Figure 2.5.1.1: System Overview Diagram 18
Figure 2.5.2.1: System Overview Diagram (Individual) 20
Figure 2.7.2.2.1: Logs from hub container 32
Figure 2.7.2.2.2: Docker composer file for hub and peer deployment 32
Figure 3.1.1: The panel of hubs 38
Figure 3.1.2: Tracing of p2p communication 38
Figure 3.1.2.1: Output from benchmark application 39

LIST OF TABLES

Table 1.2.1: Research Gap 11
Table 2.7.1.4.1.1: Testing & Implementation 25
Table 4.1: Student's contribution 42

LIST OF APPENDICES

Appendix A: Turnitin Report 48

LIST OF ABBREVIATIONS

MIT	Massachusetts Institute of Technology
TCP/IP	Transmission Control Protocol/Internet Protocol
TTL	Time To Live
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
SOAP	Simple Object Access Protocol
GRPC	Google Remote Procedure Call
P2P	Peer to Peer

1.0 INTRODUCTION

Social media platforms have become integral components in the modern lifestyles of most individuals. Social networks have established a global presence, with the initial recognizable social network, "Six Degrees," being launched in 1997, enabling users to establish profiles and connect with other users. Today, there are numerous social networks. Such as Facebook, WhatsApp, Instagram and Twitter, and their usage continues to grow [1]. These social networks are becoming popular among people due to their purpose and differences which make them compete. It is important to look into the causes behind the growing popularity of social networks. The primary reason is that users desire to keep connected with others while remaining up to date about global events. Physical interaction with others is often impossible due to limited time and responsibilities, so individuals communicate using technology, which proves more feasible. Additionally, social media is used for sharing photos and videos for entertainment and expressing opinions and ideas. And social media platforms serve as a means of researching products to purchase, prompting businesses to concentrate on online marketing and target audience acquisition. However, centralization of these social networks has led to various problems, which have recently become hot topics.

According to an article outlining several concerns in social media expressed at the social media Summit @ MIT [2], MIT Sloan professor Sinan Aral said “Social media is rewiring the central nervous system of humanity in real time” at the summit which brought professionals together to explore these challenges and focus on solutions, which ranged from new oversight committees to breaking up major corporations. Also, it mentions major concerns in present social media networks. The spread of false news and misinformation, the difficult balance between user privacy and platform transparency and a lack of regulation for social media companies are some of the areas that have been affected through social media at present.

With the emergence of web 3.0 in 2014 [3], people are inclined towards seeking solutions to the issues associated with web 2.0 technologies within the web 3.0 technology stack. As a result, decentralization has become a prominent factor.

Decentralized culture has been established, primarily due to the evolution of cryptocurrencies, particularly Bitcoin.

As mentioned in an article discussing Web 3.0 social media platforms [4], the growth of Web 3.0 is expected to persist, as individuals progressively recognize the advantages it offers in contrast to conventional social media websites. While searching for the most suitable web3 social media platforms, it is advisable to consider a range of factors such as traditional social media platforms, social media users, decentralized social networks, internet freedom, free speech, online networks, among other relevant considerations.

According to a review on the top decentralized social networks startups [5], Mastodon [6] and DeSo [7] are two of the most prominent decentralized social networking platforms currently under development. The fundamental goal of these social networks is to confront and overcome the issues connected with modern social media platforms by yielding the monopolistic control exerted by leading social media corporations and instead conceding power to the platform's actual users. Despite the proliferation of decentralized social media platforms presently, certain limitations and challenges still require attention. A majority of these platforms employ blockchain technology to achieve decentralization, which, as highlighted in a paper by R. Nourmohammadi and K. Zhang [8], can result in higher costs in terms of processing speed.

The main objective of this research endeavor is to construct a decentralized social network that utilizes the user's device as storage, while ensuring that data remains secure and authentic, with full control over user data residing with the user. The implementation of a user data decentralization protocol, along with the design and deployment of a client application capable of seamless communication with other network peers via the decentralized protocol, form the central components of this research project.

1.1 Background Literature

Developing a software application and deploying it on a single machine are the traditional approach to software development. This approach is not suitable for modern applications, which are distributed in nature and are deployed on multiple machines. These applications are called distributed applications and are very complex in nature making them very difficult to develop and maintain. Creating and managing distributed applications comes with various challenges. These challenges include the complexities of communication between applications, ensuring data consistency across distributed nodes and achieving high data availability in the face of potential disruptions.

Communication between two applications serves as the fundamental building block of distributed applications. These applications predominantly adopt one of two primary architectural approaches; client-server architecture and peer-to-peer architecture. [17]. In a client-server architecture, one application is the client and the other is the server. The client application sends a request to the server and the server application responds to the client. In peer-to-peer architecture, both the applications are peers. A peer can communicate with other peers directly and each peer has the same functionality as any other peer in the network. There is no central authority in peer-to-peer architecture.

Modern client-server architecture is based on RESTful web services [18]. RESTful web services are based on the HTTP protocol, which is stateless, meaning that the server does not remember the previous request and that it is the responsibility of the client to maintain the state of the request. Other than the HTTP protocol, there are other protocols like gRPC [19] and SOAP [20] which are used for communication between two applications.

There are a few protocols that are used for communication between two applications in peer-to-peer architecture such as the torrent protocol [15]. Most of them are specialized for a particular purpose and are therefore not used in general-purpose applications. Moreover, these protocols are very old and not maintained [11, 14]. The research component outlined in this proposal mainly focuses on developing a new, reusable protocol for communication between two peer-to-peer applications which is based on the TCP/IP protocol.

This area has seen several successful projects in the past, which include the following projects.

- Skype
- Gnutella
- Kazaa
- eMule
- Torrent

However, the majority of these projects have many usability issues, which will be discussed in the following sections.

1.2.1. Skype

Skype [9] Link is a proprietary peer-to-peer Internet telephony service that allows users to communicate with others via the Internet using voice calls, video calls, and instant messaging. It uses a centralized server which helps the system sign up new users as well as authenticating existing users with user ID and password information. The super node is a peer with higher bandwidth and CPU power and is used to improve scalability. The super nodes do the heavy lifting for the Skype service, and it is considered a disadvantage as the service relies on these super nodes, not a centralized server.

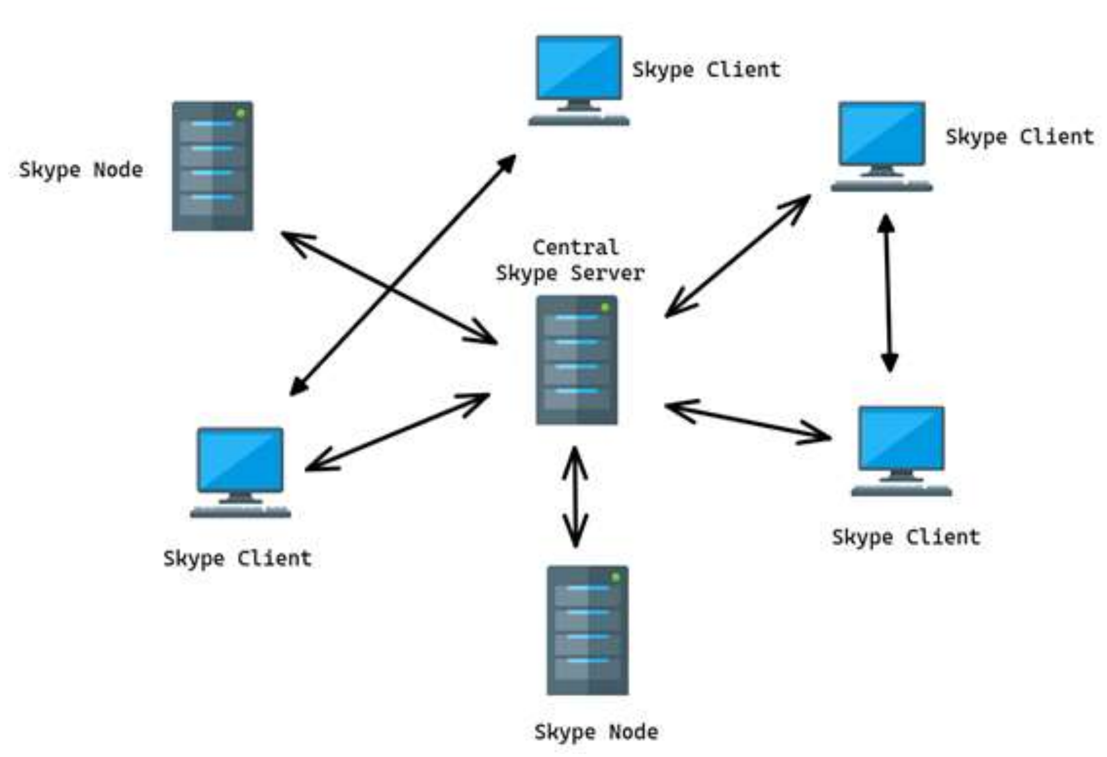


Figure 1.2.1.1: Architecture of Skype

According to Microsoft Store [10], the software release information are as follows:

- Initial release: August 29, 2003
- Latest release: 20 January 2023
- License: Proprietary

1.2.2. Gnutella

Gnutella [11] is a peer-to-peer file-sharing network using the TCP/IP stack for communication. Nodes on the network can be either clients or servers and four types of messages are used in the Gnutella protocol. Ping, Pong, Query, and Query Hit.

Flooding is used to propagate messages with each message has a Time To Live (TTL) field. When TTL reaches zero, the message is dropped. A drawback of flooding is that it floods the network with a large number of messages, this does not scale well.

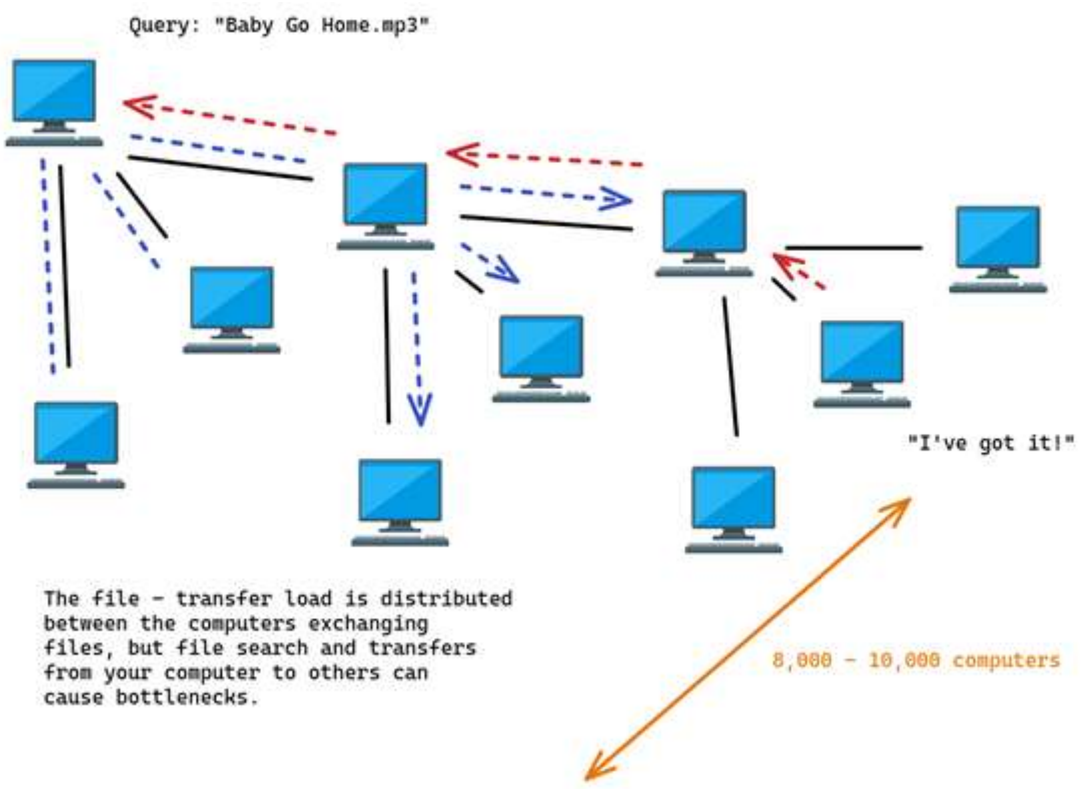

Query: "Baby Go Home.mp3"
"I've got it!"
The file - transfer load is distributed between the computers exchanging files, but file search and transfers from your computer to others can cause bottlenecks.
8,000 - 10,000 computers

Figure 1.2.2.1: Architecture of Gnutella

According to the last stable specification [12] Link, the last update was in 2001.

1.2.3. Kazaa

Kazaa [11] is a peer-to-peer file-sharing application for music, video, and other digital media files. FastTrack is the underlying protocol of Kazaa and uses the super node concept. Kazaa has the same problem as Skype, where it relies on a super node.

According to version details of Kazaa [13], the software release information are as follows:

- Initial release: March 1, 2001
- Latest release: 3.2.7 / 26 November 2006
- License: Proprietary

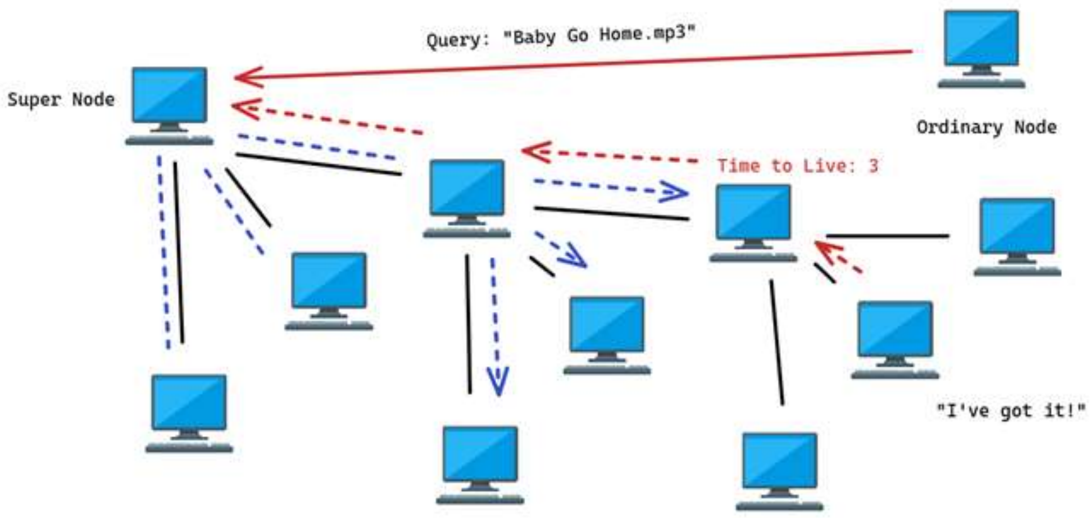


Figure 1.2.3.1: Architecture of Kazaa

1.2.4. eMule

eMule [14] is a peer-to-peer file-sharing application based on the eDonkey2000 network. eMule uses a hybrid network architecture, with a centralized server for indexing and a decentralized network for file sharing.

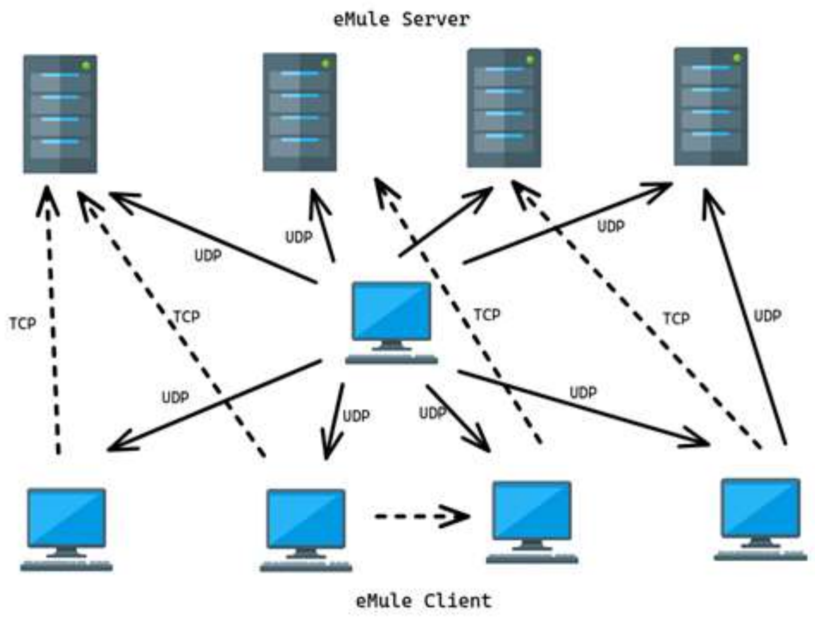


Figure 1.2.4.1: Architecture of eMule

1.2.5. Torrent

BitTorrent [15] is a peer-to-peer file-sharing protocol that is used to distribute data and electronic files over the Internet. It is one of the most common protocols for transferring large files, such as digital video files containing TV shows or video clips or digital audio files containing songs. When a user downloads a file using the BitTorrent protocol, the user becomes a seeder after the download is complete. The user can then upload the file to other users, thus becoming a leecher. With that, a file can have multiple seeders and leechers at the same time.

However, the BitTorrent protocol is specifically designed for file sharing, therefore it is not suitable for general-purpose communication.

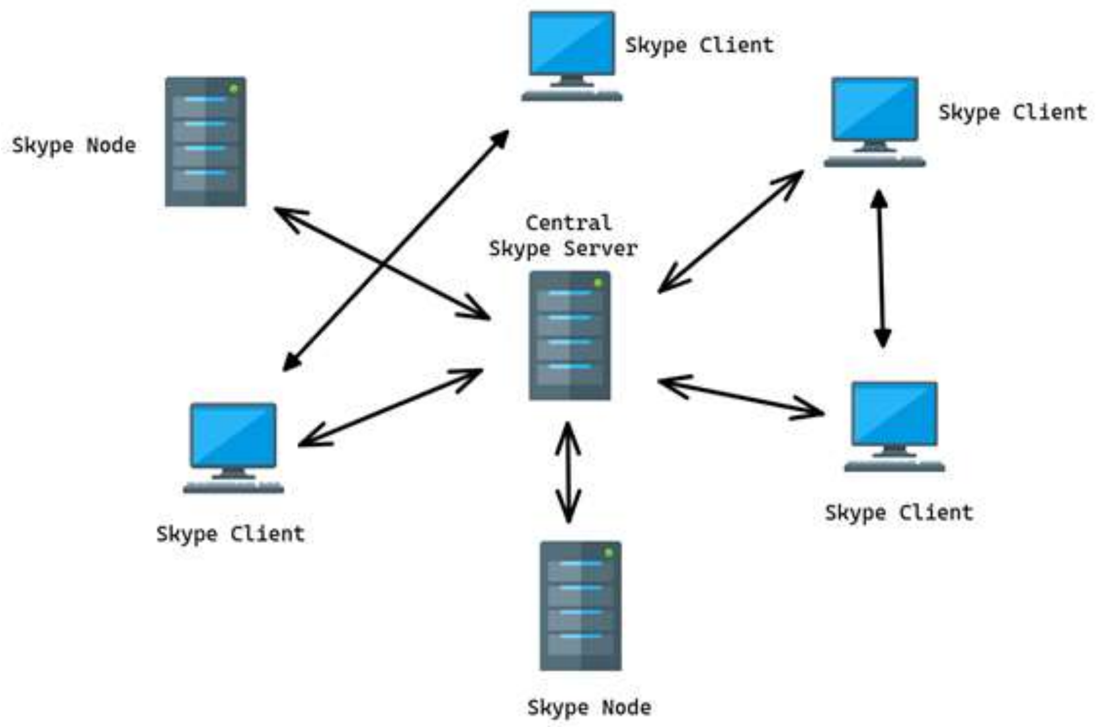


Figure 1.2.5.1: Architecture of Torrent

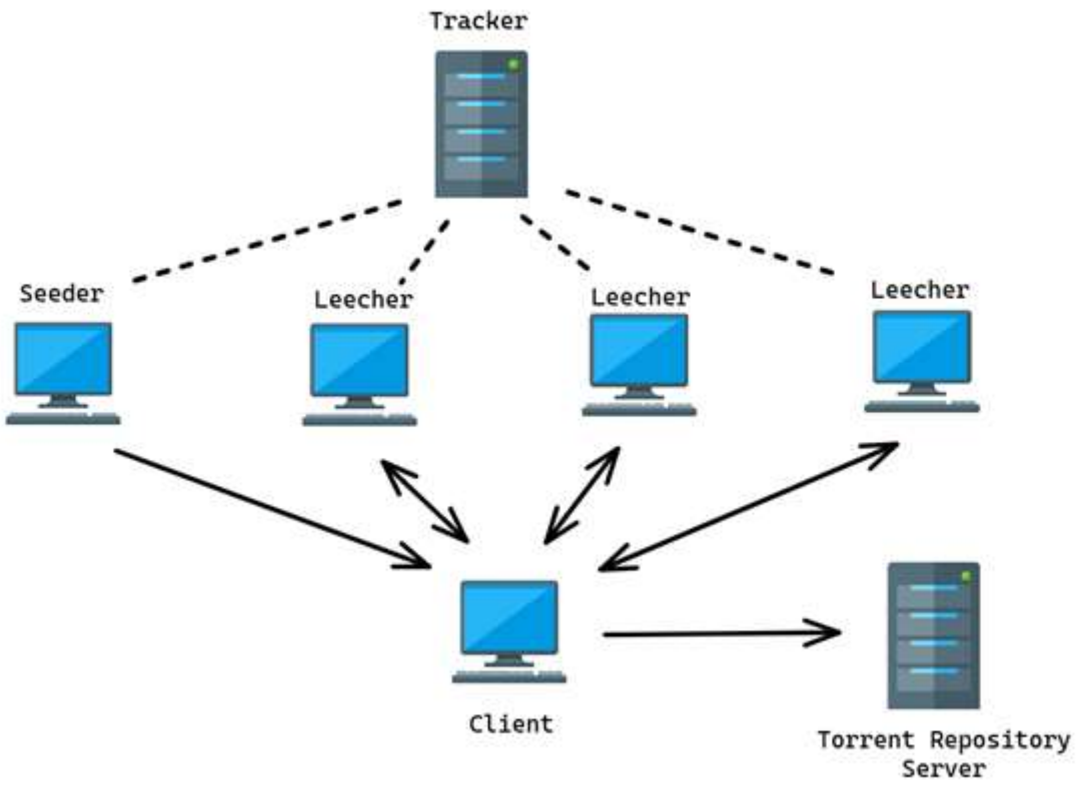


Figure 1.2.5.2: Architecture of Torrent2

In summary the problems discussed in existing projects can be condensed into the following set of issues.

- Lack of good documentation or none at all.
- Restrictive licensing or no license to be found.
- Old with the last update more than a decade away.
- No easy to reach point of contact.
- Closed source (product) or the source doesn't exist anymore.
- No specification.
- Implementation does not expose a friendly API.
- Tightly coupled with a specific use.

1.2 Research Gap

The primary aim of this research component is to create a new and reusable peer-to-peer communication protocol. Existing peer-to-peer protocols being designed for specific tasks, like file sharing, and are not suitable for general-purpose communication.

Skype is proprietary [10] and it relies on a super node [9]. Therefore, it is not suitable for general-purpose communication and the super node makes it centralized. Gnutella has open specification [12], however, it has problems such as flooding [11] and Time To Live (TTL), and according to the protocol's official sources [12], it is no longer maintained.

Like Gnutella, Torrent is also open source, but it is specifically designed for file sharing, and it is not suitable for general purpose communication. Kazaa [13] and eMule [16] are no longer maintained either, according to their respective sources.

The observations mentioned above show that although much research has been done in this area and applications have been developed, none of them are suitable for general-purpose communication, and where each attempt has failed to be suitable for this purpose can be simplified as shown in the table below.

	Skype	**Gnutella**	**Kazaa**	**eMule**	**Torrent**
Live	Yes	No	No	No	Yes
Open source	No	Yes	No	Yes	Yes
Documented	No	Yes	No	No	Yes
Scalable	No	No	No	No	Yes
General purpose	No	No	No	No	No

Table 1.2.1: Research Gap

A simplification of how existing solutions are not suitable as a general-purpose communication protocol. Therefore, developing a reusable, open-source, decentralized, peer-to-peer communication protocol is the main goal of this research component described in this proposal.

1.3 Research Problem

The Internet continues to be impacted by a number of fundamental problems after decades of protocol development. Most protocols are designed for client-server architecture and are not suitable for peer-to-peer architecture and peer-to-peer applications face scalability issues as well and while there are a few methods to scale peer-to-peer applications, they have their own drawbacks. Furthermore, most of the existing peer-to-peer protocols are designed for specific purposes, such as file sharing, and are not suitable for general-purpose communication. Other issues with such protocols include the fact that they are not open source and that the majority of them are no longer supported.

According on these findings, the following questions have been highlighted as problems with existing peer-to-peer protocols, which this research component attempts to address:

- What is the best way to communicate in a peer-to-peer network?
- How can a peer-to-peer communication network be scaled?
- How can a general-purpose peer to peer protocol be implemented?

1.4 Research Objectives

1.4.1 Main Objective

The primary goal of the proposed solution is to introduce a novel decentralized social media protocol. This protocol will prioritize keeping user data on their own devices while ensuring a smooth user experience. The component described in this proposal aims to develop a new, reusable peer-to-peer communication protocol that is suitable for general purpose communication and will be helpful for the communication between the user's device and the social media platform.

1.4.2 Specific Objectives

The objective of creating a reusable peer-to-peer communication protocol for general purpose communication would be achieved through the following sub-objectives.

- **Design an RPC (Remote Procedure Call) style full duplex communication protocol.**

In RPC, developers can define their own data structures and functions and then use RPC to call the functions remotely. RPC uses a binary format to exchange data and is a full duplex communication protocol, which means that both the client and the server can send and receive messages at the same time.

- **Use a key-value store to keep track of peers.**

Key-value store is a data structure that maps keys to values and this structure can be used to maintain a list of peers used to keep track of peers and their IP addresses. When a peer joins the network, it is added to the key-value store, and is removed from the store when a peer leaves the network.

- **Utilizing third-party storage providers for file sharing.**

Sharing files via RPC call is not efficient. Therefore, third-party storage providers can be used to share files and integrating third-party storage providers with the new peer-to-peer communication protocol will make it more efficient. The third-party storage providers can be used to share files, images, videos, and other types of data. This way, the peer-to-peer communication protocol and file sharing will be separated.

- **Implement protocol with flexibility for any kind of peer-to-peer communication.**

Developers should be able to change the protocol as they want and build their own applications on top of the protocol. The protocol should be flexible enough to be used for any kind of peer-to-peer communication.

2.0 METHODOLOGY

2.1 Requirement Gathering

The analysis is executed to identify the problems with the present investigation and the areas that need improvement. Reviewing previously released research articles on the subject usually provides the research criteria. In order to understand more about comparable publications, I also searched up a number of blogs, publications, and industry professionals. Defining and understanding the research problem as well as any past methodology, approaches, procedures, and methods is the main objective of requirement the collection.

The first stage of this research component is to collect the requirements for the new data decentralization protocol as well as the new client that will be created to use the protocol and connect to a peer-to-peer network. The requirements are derived from current peer-to-peer applications and peer-to-peer protocols. And previous research on existing data decentralization techniques and client programs for connecting to a peer-to-peer network have been reviewed and this analysis helps to identify the challenges in present research and the improvements that need to be made.

2.2 Feasibility Study

2.2.1 Technical feasibility

Peer-to-peer (P2P) communication integration is a technically feasible and strategically important step for any decentralized social media client application. The result encourages direct communication between users, enhancing their privacy and security. You may build a strong P2P social network by choosing an acceptable P2P protocol, implementing end-to-end encryption, developing an effective message routing scheme, and guaranteeing secure data storage on users' devices.

A smooth and secure user experience should also take user authentication, offline messaging, and scalability into consideration. By offering users more control over their data and interactions, this strategy not only improves the functionality of your program but also conforms to the principles of decentralized social media.

2.2.2 Schedule feasibility

This research component is expected to be completed in approximately seven months, and the work has been structured accordingly. The Gantt chart representation of how work is broken down to manageable tasks can be found in section 4.7 of this proposal. After considering the factors mentioned above, this component can be considered schedule feasible.

2.2.3 Economic feasibility

We took strategic decisions in achieving economic feasibility, choosing for open-source technology and tools. However, it is critical to recognize that expenses were made throughout the development period. These costs became because of having to use cloud service providers' services.

2.3 Requirement Analysis

The requirement analysis step was important in this research project since it allowed us to identify several issues that needed to be considered throughout the implementation phase. We carefully analyzed the data gathered from various sources throughout the requirement collection phase during this stage. This careful method enabled us to define the scope of the research and assess its feasibility.

Furthermore, the requirement analysis approach helped in identifying existing gaps in the current research environment while providing insight into the fundamental challenge of this study.

2.4 Software Development Life Cycle (Sdlc)

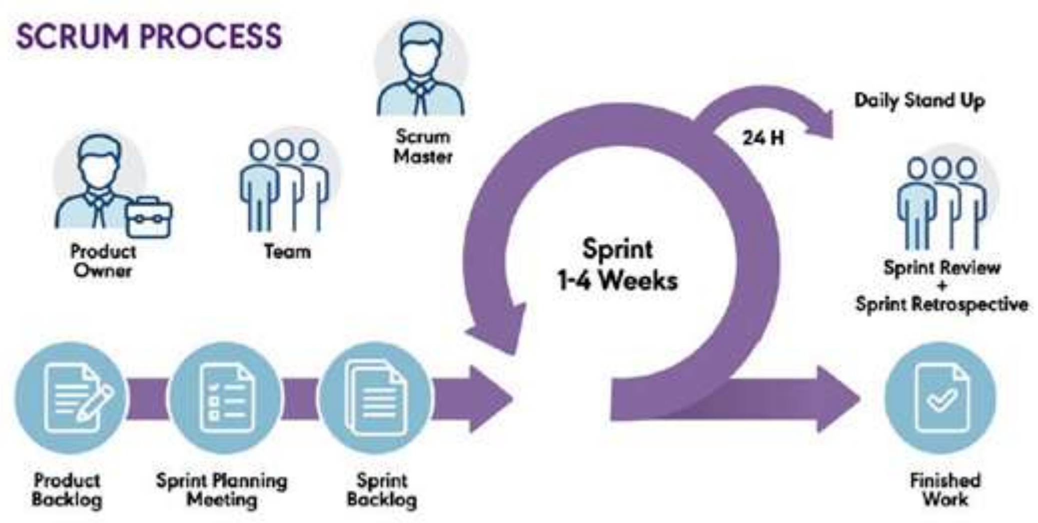


Figure 2.4.1: Software Development Life Cycle

The SCRUM framework, an Agile software development framework, will be used as the primary software project management framework throughout the research. The reason to choose agile methodology over other software project management methodologies such as Lean, Waterfall model, and Six Sigma is because it is best adapted for rapid and effective software development. According to the article [23], SCRUM is a popular agile framework because it defines the systems development process as a loose collection of activities combining the finest tools and techniques a development team can devise to create a system. According to additional information in the same article [23], SCRUM implies that the system's development process is unpredictable, complex and can only be described as an overall progression.

A systematic allocation and organization of the work have been used to achieve the research's outlined objectives and achieve the desired outcomes. A detailed schedule, complete with a Gantt chart, has been made to give each part of the research sufficient time to be finished on time. In addition, the selection of appropriate technologies to effectively implement the proposed solution and demonstrate the intended results of this research has been carefully considered. As evidenced by the detailed preparation and strategic decisions made throughout this research, each step has been taken to ensure a well-structured and systematic approach to achieving the research objectives.

2.5 PROPOSED SYSTEM DESIGN

2.5.1 SYSTEM OVERVIEW DIAGRAM (OVERALL)

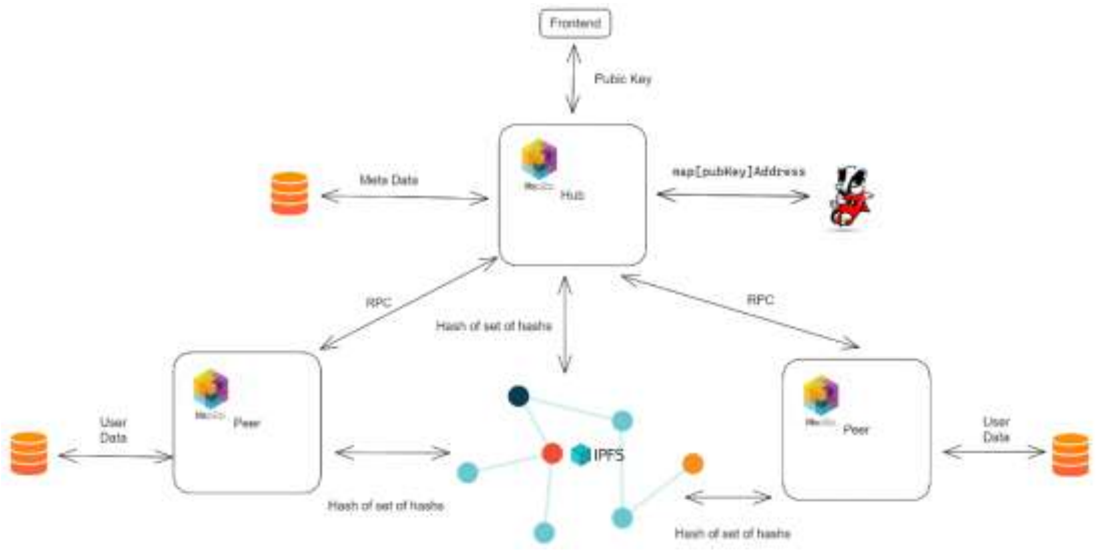


Figure 2.5.1.1: System Overview Diagram

The DeMedia architecture, which is a suggested infrastructure for a decentralized social media platform, is divided into its component parts in the description given. The architecture, which appears in the image that goes along with the explanation, can be broken down into four main parts: data decentralization protocol, peer-to-peer communication, decentralized data storage, and data integrity in a decentralized network. The data decentralization protocol, which enables the platform to keep user data within users' devices rather than on centralized servers, is the very first part of the DeMedia architecture.

Data decentralization will improve user control over their private information and improve the security and privacy of that information. Peer-to-peer communication, the second element of the DeMedia architecture, enables users to connect via a hub that serves simply as a communicator. As a result, communication will be faster and more effective while requiring fewer centralized servers. 14 Decentralized data storage, the third element of the DeMedia architecture, will be carried out using the InterPlanetary File System (IPFS). Decentralized file-sharing networks are more reliable and secure than centralized ones that can be built using the IPFS protocol. Data integrity in a decentralized network is the DeMedia architecture's fourth and final component.

This includes making sure the data kept on the decentralized network is trustworthy, safe, and immutable. Cryptography and other security measures are used to do this and guarantee that the data cannot be altered or corrupted. Overall, the DeMedia design intends to build a decentralized social media platform that provides greater user control over personal data and improved privacy, security, and efficiency. The high-level architecture diagram will show the network of hubs, peers, and a decentralized data storage system that will make up the infrastructure.

2.5.2 System Overview Diagram (Individual)

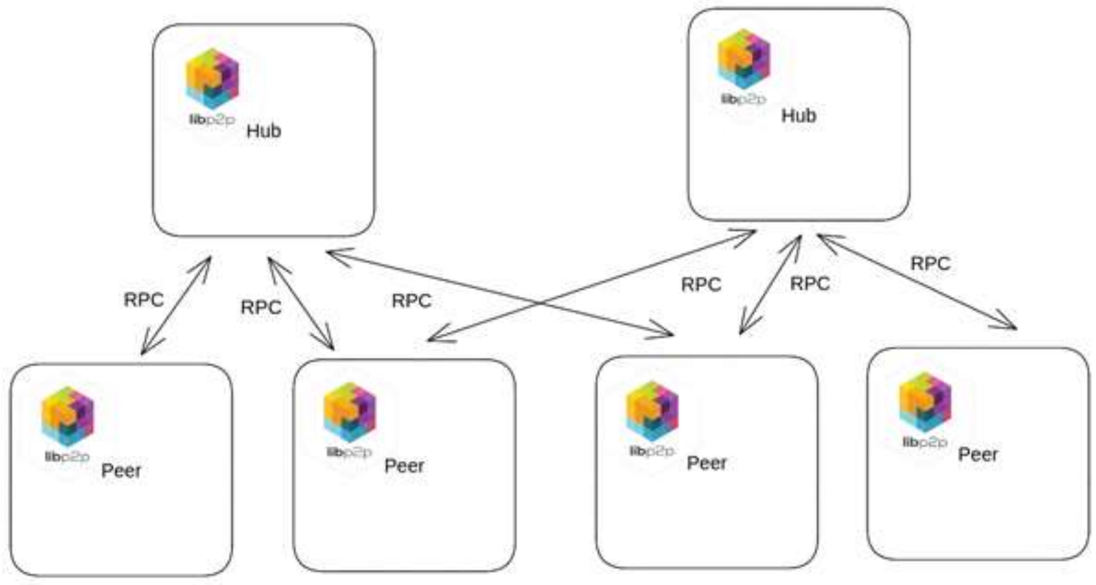


Figure 2.5.2.1: System Overview Diagram (Individual)

This component aims to develop the protocol to facilitate information decentralization along with a client program to demonstrate the protocol's capabilities. Primarily, it will be able to establish communication with hubs to connect to other peers in the network and securely store user data on the user's device. It will also display the connection information with hubs and peers.

2.6 Commercialization of the project

DeMedia is focused on developing an open-source protocol that facilitates the creation of decentralized social media platforms which can be self-hosted. The project aims to provide a base model for free, which can be used by anyone interested in creating a decentralized social media platform.

In addition to the free model, DeMedia will also offer two paid models.

- Subscription-based membership model
- Advertising-based revenue model

These paid models can be governed by the host, allowing them to generate revenue from their platform. Therefore, one could describe DeMedia as a research project focused on commercializing the development of decentralized social media platforms. The project is developing a range of monetization models that can be used by hosts to generate revenue from their platforms, thereby enabling commercialization of the technology.

To further elaborate on the commercialization aspect of DeMedia, it's important to understand that the project is focused on creating a technology that can enable the development of decentralized social media platforms. By providing a free base model, DeMedia is making it easier for individuals or organizations to create their own social media platforms that are not controlled by a centralized authority. However, to sustain and grow these platforms, there needs to be a way to generate revenue. This is where the two paid models offered by DeMedia come in.

The subscription-based membership model allows hosts to charge users for access to premium features or content on their platform. This revenue can be used to cover the costs of hosting and maintaining the platform.

On the other hand, the advertising-based revenue model enables hosts to generate revenue by displaying ads on their platform. Hosts can charge advertisers to display their ads on the platform, and this revenue can be used to cover the costs of hosting and maintaining the platform, as well as generating profits.

As a summary about this proposed system, DeMedia is enabling the commercialization of decentralized social media platforms by providing a technology that allows anyone to create their own platform, along with monetization models that enable hosts to generate revenue and sustain their platforms. This could lead to a more diverse and decentralized social media ecosystem, with a greater range of platforms catering to specific sectors and communities.

2.7 Testing & Implementation

2.7.1 Background Establishment for Implementation

The team went through a careful planning and decision-making process on the way to putting the decentralized social media protocol into implementation. This phase was thought to be essential for laying a strong basis for the next development and testing phases. In-depth discussions of the decisions made regarding the project's programming language, database system, communication tools, cloud provider, and containerization technologies will be discussed in this section. Each of these choices was crucial in determining the project's ultimate outcome.

2.7.1.1 Programming Language Selection: Go Lang

After thorough consideration, the decision was made to utilize the Go programming language, commonly known as Go Lang. There were multiple solid factors that led to this decision.

Go Lang is popular for its efficiency, simplicity, and scalability, making it an ideal choice for building decentralized systems. Its concurrency support allows multiple operations to be executed concurrently, enabling the protocol to handle a large number of users and interactions efficiently. Moreover, Go Lang offers a strong standard library and a rich ecosystem of packages, significantly expediting the development process.

The inherent support for robust error handling was considered advantageous, as it enhances the protocol's resilience and fault tolerance. In essence, Go Lang not only met the technical requirements but also aligned with the goal of creating a stable and reliable decentralized social media platform.

2.7.1.2 Database System: PostgreSQL

The selection of an appropriate database system is critical in ensuring the efficient storage and retrieval of data in any project, including the decentralized social media protocol. PostgreSQL was chosen as the database management system, a decision influenced by several key considerations.

First and foremost, PostgreSQL is recognized as a robust open-source relational database system. Its support for complex data types and advanced indexing mechanisms makes it well-suited to handling the diverse data structures prevalent in social media applications. Additionally, strong data integrity and security features offered by PostgreSQL ensure the privacy and reliability of user data.

Furthermore, PostgreSQL's extensibility through user-defined functions and a vibrant community of contributors made it adaptable to evolving project needs. The database's ability to perform efficiently even under high loads was a crucial factor in the decision, given the typically rapid and unpredictable user activity on social media platforms.

2.7.1.3 Collaboration and Version Control: GitHub

Effective collaboration and version control are indispensable for a team working on a complex project like a decentralized social media protocol. To achieve this, GitHub was adopted as the primary platform for code collaboration and integration. The reasons behind this choice were straightforward yet compelling.

GitHub provides an intuitive and user-friendly interface, simplifying the processes of code sharing, reviewing, and merging. This streamlined the development workflow, ensuring that team members could collaborate seamlessly. Its version control capabilities allowed changes to be tracked, conflicts to be managed, and a comprehensive history of the codebase to be maintained, enhancing transparency and accountability.

Moreover, GitHub's robust issue tracking system enabled effective task management and prioritization, particularly valuable in a project of this scale, where numerous features and components needed to be developed and integrated.

2.7.1.4 Cloud Deployment: Amazon Web Services (AWS)

In the modern era of software development, cloud computing has emerged as a game-changer, offering unparalleled scalability, reliability, and flexibility. The benefits of deploying development and production environments in the cloud were recognized, and after careful evaluation, Amazon Web Services (AWS) was selected as the cloud provider.

AWS's vast array of services and global infrastructure ensured that the decentralized social media protocol could scale seamlessly to accommodate increasing user loads. The pay-as-you-go pricing model allowed costs to be optimized while benefiting from world-class infrastructure and support.

Furthermore, AWS offered a range of tools and services for tasks such as server provisioning, load balancing, and auto-scaling. These features simplified the management of cloud infrastructure, freeing up valuable time and resources that could be redirected toward enhancing the protocol's functionality and performance.

2.7.1.4.1 Deployment Using Amazon EC2 Instances

For deploying Docker containers and the PostgreSQL database, an Amazon Elastic Compute Cloud (Amazon EC2) instance was utilized. Amazon EC2 instances offered several advantages, including scalability, flexibility, and ease of configuration. Below are the specifications of the Amazon EC2 instance used for this deployment:

Type	t2.micro
Number of vCPUs	1
RAM (GiB)	1.0
Storage (GiB)	30
Operating System	Ubuntu

Table 2.7.1.4.1.1: Testing & Implementation

2.7.1.5 Containerization: Docker

To achieve consistency in deployment and streamline the process, Docker, a containerization technology, was leveraged. Docker provided several advantages that significantly facilitated the implementation.

Docker's fundamental benefit lies in its ability to encapsulate applications and their dependencies within lightweight containers. Each component of the decentralized social media protocol was integrated and packaged as a Docker image. This approach not only simplified deployment but also ensured that each component could run consistently across various environments.

One of Docker's key strengths is its isolation capabilities, allowing conflicts between different components of the system to be avoided. This isolation reduced the risk of compatibility issues and contributed to the overall stability of the protocol.

Additionally, Docker's portability made it possible to develop and test components independently before seamlessly integrating them into the larger system. This modular approach enhanced development agility and minimized disruptions during the implementation phase.

2.7.1.5.1 Deployment as Docker Containers

The decision to employ Docker containers for deployment resulted in the successful deployment of the implemented demo social platform. This approach offered several tangible benefits:

- **Consistency**: Docker containers ensured that the demo platform ran consistently across different environments, eliminating the notorious "it works on my machine" problem.

- **Scalability**: Docker's scalability features allowed adaptation to varying levels of user demand effortlessly. As user activity increased, the application could be scaled horizontally by adding more containers.

- **Resource Efficiency**: Docker containers, being lightweight and sharing the host OS kernel, resulted in minimal overhead. This translated to efficient resource utilization, reducing infrastructure costs.
- **Quick Deployment**: Docker's quick start-up time enabled the rapid deployment of new updates and features, minimizing downtime and user disruption.

In conclusion, the strategic choices made in setting up the background for implementation were instrumental in ensuring the success of implementation of the decentralized social media protocol. The adoption of Go Lang, PostgreSQL, GitHub, AWS, and Docker laid a strong foundation for the subsequent phases of development and testing. These decisions, rooted in practical considerations and a deep understanding of the requirements unique to the project, enabled the creation of a robust, scalable, and efficient platform that met the goals and expectations.

2.7.2 Setting Up CI/CD Pipeline

Due to the frequent releases during the development process, the team realized the necessity for a CI/CD (Continuous Integration/Continuous Deployment) pipeline. As a result, it was decided to create a reliable CI/CD pipeline that incorporated both CI and CD components. This section describes the pipeline's implementation, which was crucial in accelerating the development and deployment procedures.

2.7.2.1 Continuous Integration (CI) Pipeline

For the CI portion of the pipeline, GitHub Actions was chosen as the preferred tool. This decision was well-founded for several reasons.

GitHub Actions offers a seamless and integrated approach to automating our software development workflows. It allowed us to define, customize, and automate various tasks, such as code compilation, testing, and code quality checks, directly within our GitHub repository.

Moreover, GitHub Actions integrates seamlessly with our codebase hosted on GitHub, making it a natural choice for our CI needs. The ease of configuration and extensive library of pre-built actions simplified the setup of our CI workflow.

2.7.2.2 Continuous Deployment (CD) Pipeline

In parallel with the CI pipeline, a CD pipeline was established using GitHub Actions and Watchtower. Each was chosen for distinct but complementary reasons.

GitHub Actions was extended to serve as our CD tool, ensuring the seamless deployment of our application. GitHub Actions' continuous deployment capabilities allowed us to automate the deployment process, ensuring that every successful CI build was automatically deployed to our production environment. This reduced manual intervention and minimized deployment errors.

Watchtower, on the other hand, played a crucial role in automating container updates. It continuously monitored our Docker containers for new image versions and automatically updated running containers to the latest versions when changes were detected. This ensured that our application was always running with the most up-to-date code, enhancing security and reliability.

Direct integration to the GitHub repository and the user-friendly configuration of GitHub Actions made them the best option for both CI and CD. It eliminated the need for third-party tools, simplifying our development workflow and ensuring that code changes were rapidly and reliably integrated and deployed.

Watchtower's selection was driven by its ability to automate container updates, reducing the need for manual intervention, and ensuring our application's consistent and secure operation.

The implementation of this CI/CD pipeline provided several benefits including:

- **Automation:** The CI/CD pipeline automated critical aspects of the development and deployment processes, reducing the manual effort required.
- **Speed and Efficiency:** Rapid and automated testing, integration, and deployment improved the overall speed and efficiency of the development cycle.
- **Consistency:** With CI/CD, every code change was subjected to the same automated testing and deployment process, ensuring consistency and reliability.
- **Reduced Errors:** Automation minimized the potential for human errors during deployment, enhancing the overall quality of our application.
- **Frequent Releases:** The CI/CD pipeline facilitated frequent and reliable releases, crucial for the development requiring rapid iterations and updates.

The establishment of the CI/CD pipeline using GitHub Actions and Watchtower significantly enhanced the development and deployment processes, aligning with the unique demands of the project. This automation not only improved efficiency but also contributed to the consistency and quality of our work, enabling to meet project milestones and deliver robust outcomes.

2.7.3 Development

This section describes the development work done for the component, focusing on the use of LibP2P, a well-known library used in well-known projects like IPFS and Ethereum for Peer-to-Peer (P2P) communication.

- **LibP2P: A Versatile Library**

LibP2P is a recognized, industry-standard library known for its adaptability and wide range of features. It is used to enable effective P2P communication in a number of blockchain projects, including IPFS and Ethereum. A strong P2P communication protocol was built for the decentralized social media network using LibP2P, which was utilized throughout the project.

Different implementations of LibP2P are available in numerous programming languages, including Go, JavaScript, Rust, and Java. LibP2P's Go implementation, considered to be the most developed and dependable option, was chosen for the project.

The DeMedia P2P communication protocol uses LibP2P's "go-libp2p-gorpc" RPC implementation to adopt an RPC (Remote Procedure Call) style approach. Building on LibP2P, this protocol adds features like streaming RPC calls via channels and Contexts, as well as asynchronous calls, to facilitate RPC communication.

- **Multiaddr: Identifying Peers**

Peer identification is an important component in P2P networking. The protocol used for this, multiaddr, consists of several layers of protocols and has a more readable encoding.

Peer information is first extracted from the multiaddr in the procedure. After that, use DNS resolution to discover the peer's IP address. "Go-multiaddr-dns" library was utilized to facilitate DNS resolving.

- **Establishing Connections and RPC**

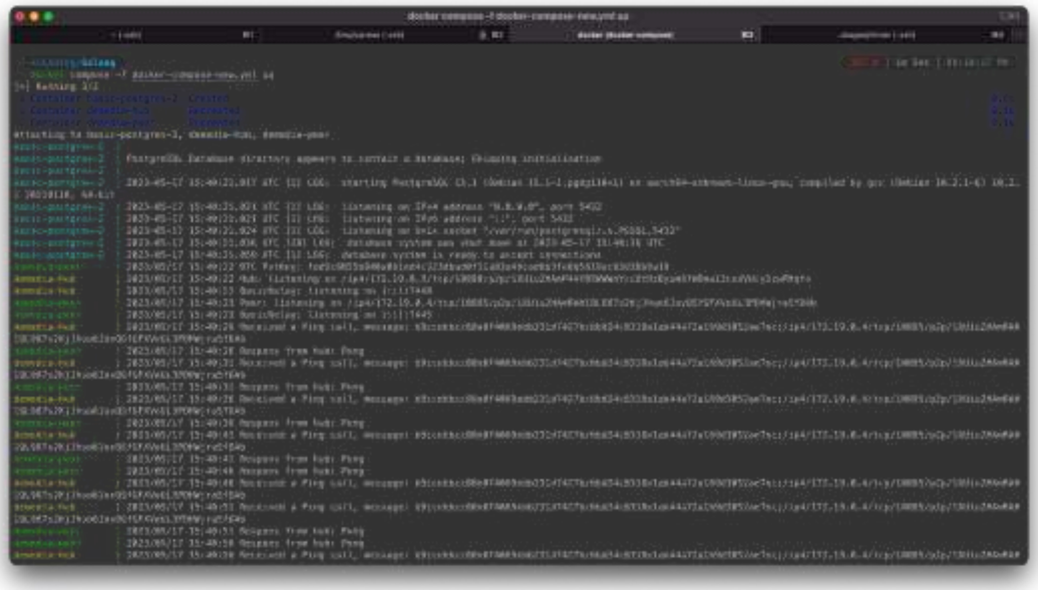
With the peer's IP address in hand, proceed to establish a connection with the target peer. This connection sets the stage for communication.

The development of an RPC client is then carried out in a similar manner, using the "go-libp2p-gorpc" library. The client is then ready to communicate a request payload to the chosen peer by converting it into a byte array.

- **Peer-Side Handling**

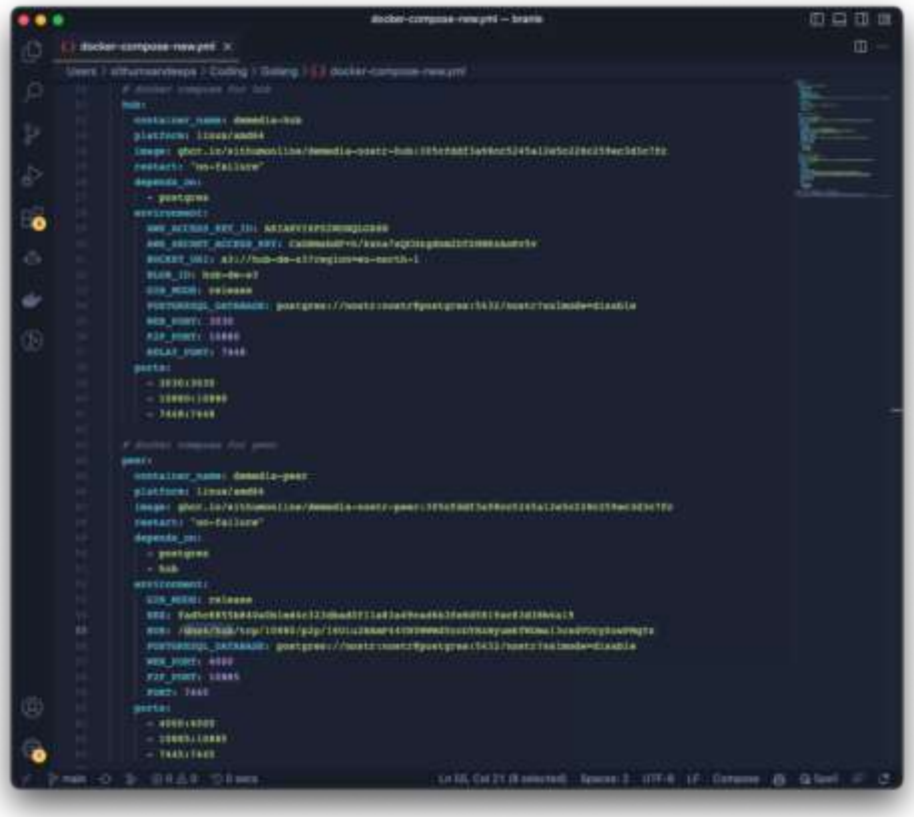
On the peer's side, RPC server is initiated, registering RPC methods as required. After that, the RPC server is launched and prepared to respond to incoming queries. When a request arrives, convert the request payload into the appropriate request structure and proceed to invoke the corresponding method.

In conclusion, the development work focused on utilizing LibP2P's capabilities and creating a strong P2P communication protocol for DeMedia. This protocol smoothly integrates with LibP2P's RPC capabilities and makes use of Multiaddr to identify peers, creating a productive and trustworthy P2P communication system.



```
docker compose -f docker-compose-new.yml up
[+] Running 3/3
 Container basic-postgres-2  Created
 Container demedia-hub       Recreated
 Container demedia-peer      Recreated
Attaching to basic-postgres-2, demedia-hub, demedia-peer
basic-postgres-2  |
basic-postgres-2  | PostgreSQL Database directory appears to contain a database; Skipping initialization
basic-postgres-2  |
basic-postgres-2  | 2023-05-17 15:49:21.817 UTC [1] LOG:  starting PostgreSQL [illegible]
basic-postgres-2  | 2023-05-17 15:49:21.821 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
basic-postgres-2  | 2023-05-17 15:49:21.821 UTC [1] LOG:  listening on IPv6 address "::", port 5432
basic-postgres-2  | 2023-05-17 15:49:21.824 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
basic-postgres-2  | 2023-05-17 15:49:21.836 UTC [28] LOG:  database system was shut down at 2023-05-17 15:48:18 UTC
basic-postgres-2  | 2023-05-17 15:49:21.850 UTC [1] LOG:  database system is ready to accept connections
demedia-peer      | 2023/05/17 15:49:22 UTC Pubkey: [illegible]
demedia-hub       | 2023/05/17 15:49:22 Hub: listening on /ip4/172.19.0.3/tcp/10880/p2p/[illegible]
demedia-hub       | 2023/05/17 15:49:23 QuicRelay: listening on [::]:7448
demedia-peer      | 2023/05/17 15:49:23 Peer: listening on /ip4/172.19.0.4/tcp/10885/p2p/[illegible]
demedia-peer      | 2023/05/17 15:49:23 QuicRelay: listening on [::]:7445
demedia-hub       | 2023/05/17 15:49:26 Received a Ping call, message: [illegible]
demedia-peer      | 2023/05/17 15:49:26 Respons from hub: Pong
demedia-hub       | 2023/05/17 15:49:31 Received a Ping call, message: [illegible]
demedia-peer      | 2023/05/17 15:49:31 Respons from hub: Pong
[illegible]
```

Figure 2.7.2.2.1: Logs from hub container



```yaml
# docker compose for hub
hub:
  container_name: demedia-hub
  platform: linux/amd64
  image: ghcr.io/[illegible]/demedia-nostr-hub:[illegible]
  restart: "on-failure"
  depends_on:
    - postgres
  environment:
    AWS_ACCESS_KEY_ID: [illegible]
    AWS_SECRET_ACCESS_KEY: [illegible]
    BUCKET_URL: s3://hub-de-s1?region=eu-north-1
    BLOB_ID: hub-de-s1
    GIN_MODE: release
    POSTGRESQL_DATABASE: postgres://nostr:nostr@postgres:5432/nostr?sslmode=disable
    WEB_PORT: 3030
    P2P_PORT: 10880
    RELAY_PORT: 7448
  ports:
    - 3030:3030
    - 10880:10880
    - 7448:7448

# docker compose for peer
peer:
  container_name: demedia-peer
  platform: linux/amd64
  image: ghcr.io/[illegible]/demedia-nostr-peer:[illegible]
  restart: "on-failure"
  depends_on:
    - postgres
    - hub
  environment:
    GIN_MODE: release
    SEC: [illegible]
    HUB: /dns4/hub/tcp/10880/p2p/[illegible]
    POSTGRESQL_DATABASE: postgres://nostr:nostr@postgres:5432/nostr?sslmode=disable
    WEB_PORT: 4000
    P2P_PORT: 10885
    PORT: 7445
  ports:
    - 4000:4000
    - 10885:10885
    - 7445:7445
```

Figure 2.7.2.2.2: Docker composer file for hub and peer deployment

2.8 Testing

In terms of testing, the approach used was intended to guarantee the accuracy and dependability of the research project. Each development effort was verified through a methodical testing process before it was integrated into the project's codebase hosted on GitHub and after the deployment.

2.8.1 Unit Testing

Unit tests played an important role in the testing process. These tests were conducted before code was pushed to GitHub, serving as a initial validation step. Unit tests are small, focused tests that verify the correctness of individual components or functions within the code.

Through the conduct of unit tests, potential issues were detected and rectified at an early stage, preventing them from propagating into the codebase. This approach ensured that each development effort underwent validation for correctness and functionality, contributing to the overall stability of the project.

2.8.2 Continuous Integration (CI) and Deployment (CD)

The testing process was tightly integrated with the CI/CD pipeline, streamlining the testing and deployment of developed features. Here's how it worked:

Continuous Integration (CI): Upon completion of development efforts, the CI pipeline automatically integrated these features. It compiled the code, ran unit tests, and ensured that the new code did not introduce regressions or errors.
Docker Image Building: Following successful CI, the pipeline proceeded to build Docker images. Docker images are like packaged containers that encapsulate the application and its dependencies.

Continuous Deployment (CD): A crucial aspect of the testing process was the CD pipeline. This pipeline utilized Watchtower, an automated container updating tool.

When new Docker images were created, Watchtower was triggered as part of the CD process.

Watchtower Deployment: Watchtower's role was to pull the latest Docker images with updated code and deploy them on Amazon Elastic Compute Cloud (Amazon EC2) instances. This automated process ensured that the application consistently ran the latest code changes.

2.8.3 Smoke Testing

After each deployment, a critical step was the execution of smoke tests. Smoke tests are a set of initial tests designed to verify that the newly deployed release is stable and functional. They are meant to ensure that the release is "smoke-free," indicating that it is ready for further testing and use.

The testing approach offered several notable benefits to the project:
Early Issue Detection: Unit tests allowed for the early detection and resolution of issues in the development process, reducing the likelihood of critical errors in the final product.

Quality Assurance: Through thorough unit tests and continuous integration, a high level of code quality and reliability was maintained.

Efficiency: Automation within the CI/CD pipeline reduced manual testing efforts, enabling faster development cycles and quicker deployment of new features.
Consistency: The automated deployment process with Watchtower ensured that the application consistently operated with the latest code changes, enhancing overall reliability.

Rapid Updates: The testing and CD approach facilitated swift updates and deployments, essential for an academic research project requiring frequent iterations and enhancements.

In conclusion, the testing approach, included unit testing, seamless integration with the CI/CD pipeline, and smoke testing, was important in maintaining the quality, reliability, and efficiency of the project. It enabled the validation of each development effort, early error identification, and the delivery of a robust and continuously evolving research platform.

2.8.4 Validation using practical implementations.

In order to assess the practical implementations of the project, the development of a demo social media platform and a separate benchmarking application was initiated. These efforts allowed valuable insights into the performance and functionality of the built protocol.

2.8.4.1 Benchmarking with Infrastructure as Code (IaaC) Approach

To deploy the benchmarking application, along with the required infrastructure and dependent applications, an Infrastructure as Code (IaaC) approach was followed. This approach brought significant advantages:

Reproducibility: IaaC allowed the definition and recreation of the entire infrastructure consistently, minimizing discrepancies between deployments.

Version Control: Infrastructure configurations were version-controlled, enabling the tracking of changes, effective collaboration, and maintenance of a comprehensive history.

Scalability: With IaaC, easy scaling of the infrastructure up or down in response to varying workloads was possible, ensuring optimal performance.

2.8.4.1.1 Terraform as the IaaC Tool

For implementing the IaaC approach, terraform was selected as the tool of choice. Terraform provided numerous benefits, including:

Infrastructure ambiguity: Terraform supports multiple cloud providers and infrastructure types, giving flexibility in choosing the best-suited resources.

Declarative Syntax: Terraform’ s declarative syntax made it easy to define and manage infrastructure configurations, enhancing readability and maintainability.

Modularity: Terraform allowed the creation of reusable modules, simplifying the deployment of complex infrastructure components.

Along with these 2 practical implementations, testing is carried out to validate the functionality, usability, and reliability of the protocol.

3.0 RESULTS AND DISCUSSIONS

3.1 Results

The implementation of both the demo social media platform and the benchmark application played an important role in validating various aspects of the project. Here, present the results obtained from these implementations, highlighting the project's key achievements and insights.

3.1.1 Demo Social Media Platform Results

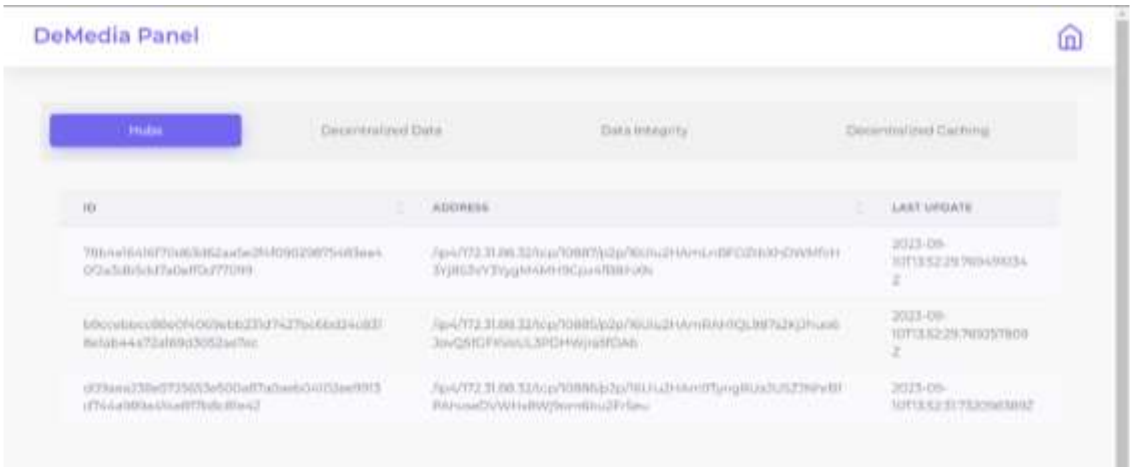
The development of the demo social media platform on top of the implemented decentralized social media protocol produced below noticeable results:

User Engagement: The platform successfully facilitated user engagement, demonstrating the protocol's effectiveness in creating a user-friendly social media environment.

Feature Integration: Various features, such as user profiles, posts, and interactions, were seamlessly integrated and validated, showcasing the versatility of the protocol.

User Experience: Users were provided a positive and intuitive experience while navigating and interacting with the demo social media platform, highlighting its user-centric design.

Stability: The platform established stability and robustness, with minimal downtime or disruptions during usage, ensuring a reliable user experience.



ID	ADDRESS	LAST UPDATE
[illegible]	/ip4/172.31.[illegible]/tcp/[illegible]/p2p/[illegible]	2023-09-10T13:52:29.[illegible]Z
[illegible]	/ip4/172.31.[illegible]/tcp/[illegible]/p2p/[illegible]	2023-09-10T13:52:29.[illegible]Z
[illegible]	/ip4/172.31.[illegible]/tcp/[illegible]/p2p/[illegible]	2023-09-10T13:52:31.[illegible]Z

Figure 3.1.1: The panel of hubs

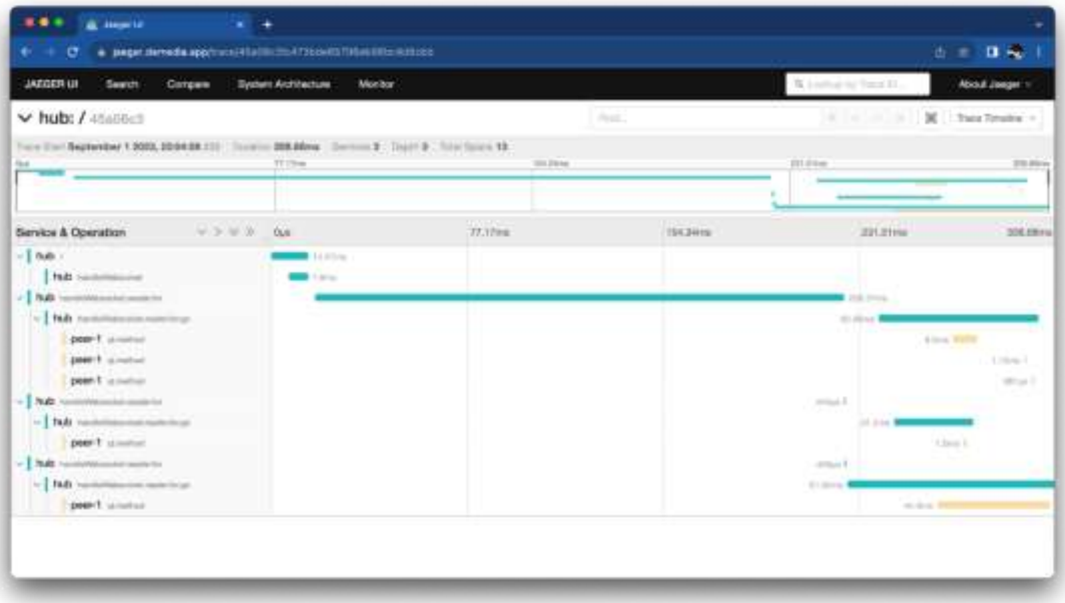
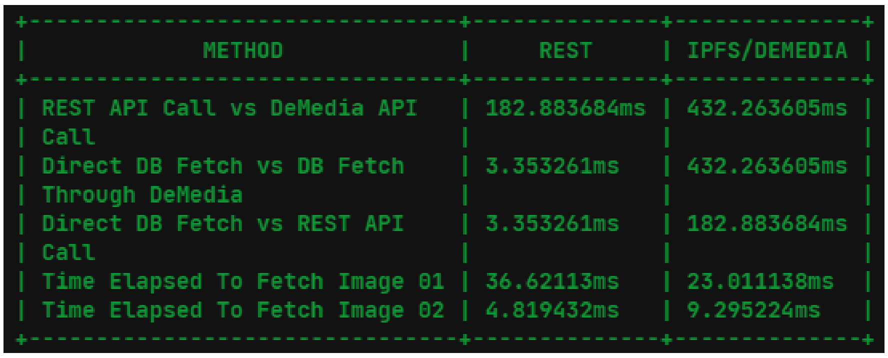


Figure 3.1.2: Tracing of p2p communication

3.1.2 Benchmarking Results

The benchmarking application provided insightful results on the performance and capabilities of the built protocol. These results, which are detailed below, are integral to the project's evaluation and future improvements:



```
+--------------------------------+-------------+--------------+
|             METHOD             |    REST     | IPFS/DEMEDIA |
+--------------------------------+-------------+--------------+
| REST API Call vs DeMedia API   | 182.883684ms | 432.263605ms |
| Call                           |             |              |
| Direct DB Fetch vs DB Fetch    | 3.353261ms  | 432.263605ms |
| Through DeMedia                |             |              |
| Direct DB Fetch vs REST API    | 3.353261ms  | 182.883684ms |
| Call                           |             |              |
| Time Elapsed To Fetch Image 01 | 36.62113ms  | 23.011138ms  |
| Time Elapsed To Fetch Image 02 | 4.819432ms  | 9.295224ms   |
+--------------------------------+-------------+--------------+
```

Figure 3.1.2.1: Output from benchmark application

3.1.3 Results from overall practical implementations

The development of the demo social media platform and the benchmark application allowed for the validation of several critical aspects of the project, including:

Functionality: Through rigorous testing and usage of the demo social media platform, validated the core functionality of the implemented decentralized social media protocol.

Scalability: By deploying the demo platform and benchmark application at scale, assessed the protocol's ability to handle a substantial user load and interactions effectively.

Security: Security features and measures were thoroughly examined and validated through the development of the demo platform, ensuring the protection of user data and interactions.

Performance: Performance metrics were gathered and analyzed to evaluate the speed and efficiency of the decentralized social media protocol in real-world scenarios.

3.2 Research Findings

According to recent research on decentralized social media protocols, peer-to-peer (P2P) communication has significant advantages in improving user privacy, reducing data latency, and increasing resistance to censorship. P2P systems spread data between nodes instead of centralized servers, making them less vulnerable to single points of failure or targeted censorship. However, issues remain, especially in terms of scalability and guaranteeing a consistent user experience. Maintaining efficient P2P communication gets more difficult as the number of users increases, demanding new approaches to ensure platform viability.

As mentioned in the results section, P2P communication protocol cannot give effective performance like rest API. It is because of the nature of the P2P communication protocol. It is not designed for high performance; it is designed for decentralized communication. It takes more time to establish a connection with the peer. In decentralized systems, it is not a big problem, because we don't need to establish a connection with the peer every time. We can establish a connection with the peer once and use that connection for the rest of the communication.

3.3 Discussion

Currently, we cannot find any reusable P2P communication protocol which is flexible enough to use in different decentralized systems. For that reason, we decided to build our own P2P communication protocol. We use LibP2P as a base for our P2P communication protocol. LibP2P is a very extensive library, and it is flexible enough to make different P2P communication protocols. Even DeMedia P2P communication protocol is under perform compared to the rest API, it is still a good solution for the decentralized systems. P2P communication protocol takes more time to route the request to the peer, but it is more secure and decentralized. Finding a peer in the network is the most challenging and complex part of any P2P communication protocol. DeMedia P2P communication protocol is not an exception, it is also challenging to find a peer in the network. If someone wants to make this protocol more efficient, they should focus on the network discovery part of the protocol. They can be using different techniques to make network discovery more efficient, such as DHT, gossip protocol, and so on.

An admin panel was created within the demo platform to showcase the inner workings of the protocol. This panel visually represented the decentralized communication architecture, demonstrating how hubs serve as intermediaries between peers and facilitate content sharing. Users could view hub details and the peer connections established through the hub. This visual representation enhanced user understanding of the protocol’s distributed nature and emphasized the absence of centralized control.

4.0 SUMMARY OF EACH STUDENT'S CONTRIBUTION

Student Number	**Name**	**Tasks**
IT29254698	Perera B.S.S.	• Implement RPC-style full duplex peer-to-peer communication protocol. • Implement a key-value store to keep track of peers. • Document the protocol. • Test the protocol.

Table 4.1: Student's contribution

5.0 CONCLUSION

The main goal of the research component that has been detailed in this proposal is to develop a new and reusable peer-to-peer communication protocol. Many of the existing peer-to-peer protocols are designed for specific purposes, such as file sharing. Consequently, they may not be well-suited for general-purpose communication. The aim is to address this limitation by developing a protocol that can be used effectively in a wide range of communication scenarios. This component also aims to overcome the problems seen in many of the existing solutions and projects in this field such as a lack of scalability and maintenance.

This research component also aims to address common issues identified in many existing solutions and efforts in this sector. Scalability and continuous maintenance concerns are among these challenges. The goal is to design a peer-to-peer communication protocol that not only offers flexibility but also overcomes the practical barriers that are frequently experienced in comparable projects by recognizing these problems and actively working to find solutions.

The overall objective is to contribute to a more efficient, flexible, and user-friendly peer-to-peer communication architecture that can satisfy a wide range of communication requirements.

REFERENCES

[1]. B. in Marketing, “The evolution of social media: How did it begin and where could it go next?,” Maryville Online, 03-Mar-2021. [Online]. Available: https://online.maryville.edu/blog/evolution-social-media/. [Accessed: 20-Mar-2023].

[2]. S. Brown, “Social media is broken. A new report offers 25 ways to fix it,” MIT Sloan, 30-Jun-2021. [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/social-media-broken-a-new-report-offers-25-ways-to-fix-it. [Accessed: 20-Mar-2023].

[3]. Sketchar, “What advantages does web 3.0 give to creators?,” Sketchar, 03-Mar-2023. [Online]. Available: https://blog.sketchar.io/what-advantages-does-web-3-0-give-to-creators/. [Accessed: 20-Mar-2023].

[4]. “Web3 social media platforms in 2023: Why they are the future,” GamesPad, 14-Mar-2023. [Online]. Available: https://gamespad.io/web3-social-media-platforms-in-2023-why-they-are-the-future/. [Accessed: 20-Mar-2023].

[5]. “Top decentralized social networks startups,” Tracxn. [Online]. Available: https://tracxn.com/d/trending-themes/Startups-in-Decentralized-Social-Networks. [Accessed: 20-Mar-2023].

[6]. “Decentralized social media,” Mastodon. [Online]. Available: https://joinmastodon.org/. [Accessed: 20-Mar-2023].

[7]. S. Freight, “The decentralized social blockchain,” DeSo. [Online]. Available: https://www.deso.com/. [Accessed: 20-Mar-2023].

[8]. R. Nourmohammadi and K. Zhang, "An On-Chain Governance Model Based on Particle Swarm Optimization for Reducing Blockchain Forks," in IEEE Access, vol. 10, pp. 118965-118980, 2022, doi: 10.1109/ACCESS.2022.3221419.

[9] Y. A.-N. R. M. N. Mehdi Jahanirad*, "Security measures for VoIP application: A state of the art review," Academic Journals, 2011.

[10] "Skype," [Online]. Available: https://apps.microsoft.com/store/detail/skype/9WZDNCRFJ364?hl=en-us&gl=us.

[11] S. K. M. G. Leonidas Lymberopoulos, "Deliverable D.6.1: ARGUGRID Platform Design," 2007.

[12] "GnutellaProtocol0," [Online]. Available: https://courses.cs.washington.edu/courses/cse522/05au/gnutella_protocol_0.4.pdf .

[13] "Kazaa," [Online]. Available: https://www.cvedetails.com/version-list/1550/2662/1/Kazaa-Kazaa-Media-Desktop.html.

[14] D. B. Yoram Kulbak, "The eMule/eDonkey protocol," 2005 17 January. [Online]. Available: http://pages.di.unipi.it/ricci/e-mule-report.pdf.

[15] B. Cohen, 22 May 2003. [Online]. Available: https://stuker.com/wp-content/uploads/import/i-1fd3ae7c5502dfddfe8b2c7acdefaa5e-bittorrentecon.pdf.

[16] "eMule," [Online]. Available: https://sourceforge.net/projects/emule/files/eMule/.

[17] Robin Jan Maly ETH Zurich, Switzerland, 2003 March. [Online]. Available: https://pub.tik.ee.ethz.ch/students/2002-2003-Wi/SA-2003-16.pdf.

[18] Alex Rodriguez, "Introduction to RESTful Web services," IBM, 5-Nov-2008. [Online]. Available: https://developer.ibm.com/articles/ws-restful/. [Accessed: 29-Apr-2023].

[19] Kasun Indrasiri and Danesh Kuruppu, "gRPC - Up and Running," Jan-2020. https://www.oreilly.com/library/view/grpc-up-and/9781492058328/

[20] XML Protocol Working Group, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," 27-Apr-2007. [Online]. Available: https://www.w3.org/TR/soap12/

GLOSSARY

Blockchain - A decentralized, distributed digital ledger that records transactions across multiple devices and networks.

Decentralized - Something is not controlled or managed by a single central authority or entity.

Hub - A central point or place where things come together or connect.

Peer - Entities that have equal or similar within a group or network.

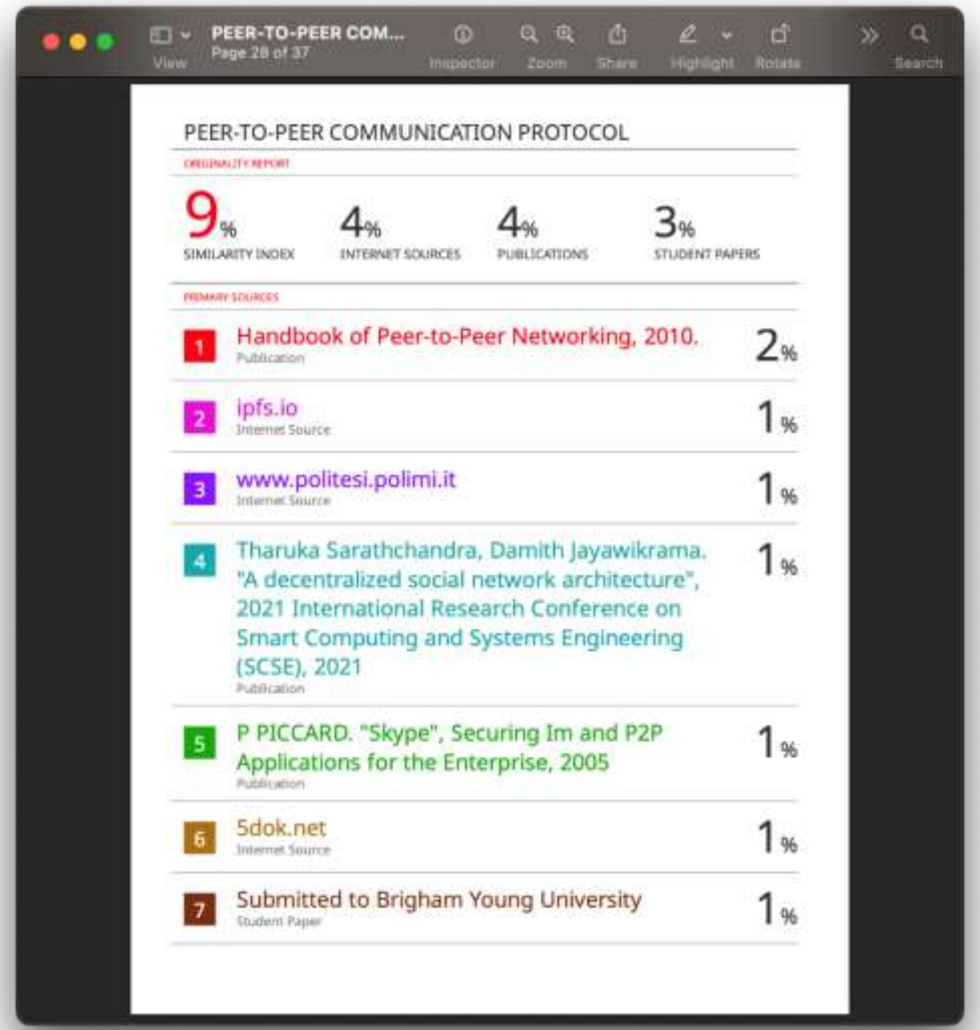
Peer-to-peer (P2P) caching - a decentralized caching mechanism in which nodes in a network cache data for each other, reducing the load on the network and improving performance.

Node - An object or computer in a P2P network that may communicate directly with other nodes to send and receive data.

Protocol - A set of instructions that specify how data is transferred between hardware or software.

Bandwidth - The quantity of data that can be sent through a network in a given amount of time.

APPENDICES



PEER-TO-PEER COMMUNICATION PROTOCOL

ORIGINALITY REPORT

9%	4%	4%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Handbook of Peer-to-Peer Networking, 2010. Publication	2%
2	ipfs.io Internet Source	1%
3	www.politesi.polimi.it Internet Source	1%
4	Tharuka Sarathchandra, Damith Jayawikrama. "A decentralized social network architecture", 2021 International Research Conference on Smart Computing and Systems Engineering (SCSE), 2021 Publication	1%
5	P PICCARD. "Skype", Securing Im and P2P Applications for the Enterprise, 2005 Publication	1%
6	5dok.net Internet Source	1%
7	Submitted to Brigham Young University Student Paper	1%

Appendix A: Turnitin Report