Decentralising Content Retrieval on the Decentralised Web

Navin V. Keizer

A thesis submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London

Department of Electronic and Electrical Engineering

University College London

September 29, 2023

I, Navin V. Keizer, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

In recent times, the control, governance, and management of the Web have become increasingly centralised, which has led to several challenges such as a lack of security and privacy protection, as well as increased censorship. To overcome these issues, a number of initiatives have emerged that offer decentralised counterparts of various components of the Web, aiming to create a fully decentralised Web, also known as DWeb or Web3. Novel technologies like blockchains and decentralised storage networks (DSN) offer ways of establishing trust and storing content without centralised trust assumptions, while established technologies like peer-to-peer (P2P) overlay networks are used as their base.

This thesis explores the *feasibility of decentralising the Web from a content retrieval perspective*. In order to understand the emerging paradigm of the DWeb, a framework is proposed for studying novel works in the area, and an extensive analysis is provided on current initiatives. The area of content search has been most unexplored, and therefore this thesis presents a truly decentralised search engine based on similarity search, which can be extended to implement keyword search. This mechanism achieves up to 57% recall of results compared to baseline, and achieves sub millisecond delays in keyword search for a network size up to 5000.

Furthermore, in order to facilitate a DWeb, resource sharing over P2P networks requires fair exchange of work for resources. To realise this, a decentralised allocation mechanism is proposed on the blockchain based on the stable matching problem. This system achieves low smart contract costs, reducing cost by at least 52% compared to prior work. Furthermore, allocation throughput and delays improve over know auction mechanisms, staying under 0.5 seconds for 9000 clients. To further facilitate this system, trust and reputation systems are required, and therefore a personalised reputation system framework is proposed, using a model learning from previous blockchain transactions and user preferences.

Finally, novel DWeb technologies generally leverage P2P networking as their base, and a number of known challenges remain in this area such as security and privacy concerns. One main challenge is presented by networking infrastructures such as network address translators (NAT), which limit individuals ability to participate in P2P networks. To tackle this issue, a decentralised NAT traversal mechanism is proposed using blockchain smart contracts for resource sharing, and reputation-based peer discovery. Evaluation of the system shows that using a reputation system of combined metrics and a two round peer discovery is able to achieve well under 5% malicious nodes chosen as NAT providers, even with the number of malicious nodes in the network reaching 30%, and the system is able to stabilise in 5-10 service cycles.

Impact Statement

In the period during which the work for this thesis was completed, the demand and interest in DWeb technologies grew exponentially. More commonly known as Web3, and linked to the underlying blockchain and associated cryptocurrencies, this technology has become well known throughout society. While many individuals became interested due to the seemingly astronomical returns on investment in cryptocurrency markets, many businesses have explored transforming key processes using blockchain services [1, 2]. From the networking perspective, this movement has proven that traditional concepts such as peer-to-peer networks remain relevant today, and have scaled to serve millions of users [3]. Because the general area moves at a rapid pace, a lot of misinformation has been circulated regarding the underlying technology.

My work provides a comprehensive overview of the field with a focus on content retrieval, both in terms of academia and industry, from a research perspective, and analyses if the claims hold up in practise. To achieve this, I present a clear framework to study the area, and provide extensive background on key concepts and works from the extensive research literature. Furthermore, through my works on search engines, allocation mechanisms, reputation systems, and network infrastructure sharing, I have carefully assessed the feasibility and pitfalls of decentralisation on the Web. As discussed throughout this thesis, while there are challenges in realising a DWeb, there are many potential benefits to society, as it may lead to a more transparent, secure, open, and democratised Web. As the Web has become a central component in daily life, this may have an enormous impact on the way people interact with information and each other.

The work in this thesis is relevant both to academics and industry practitioners, and was published at four conferences, while several submissions and collaborations with researchers at various institutions are still ongoing. My work is relevant to new, experienced, and cross-disciplinary researchers beyond the field of networking, as well as industry communities in Web3, general networking, as well as outsiders who wish to learn more and adapt their business by integrating these technologies. I have interacted with these communities by presenting my work at academic meetings, secondary education schools, as well as industry events. Finally, I have had the opportunity to involve a number of students to work in this area as part of UCL undergraduate projects, which has produced interesting and informative reports on decentralised advertisement markets and non-fungible token (NFT) [4] creation.

Acknowledgements

I would like to express my gratitude to everyone who has supported me in my academic journey, and who directly and indirectly supported me in finishing this thesis. First of all, I would like to thank Prof. George Pavlou for his support and direction during my PhD, and for stimulating my personal learning and development. It has been inspiring to work with someone with his experience and expertise, and I have greatly appreciated the combination of freedom and guidance in developing my research ideas. Furthermore, I am grateful to Dr. Onur Ascigil and Dr. Michał Król for our collaborations and exchange of ideas, and for helping me push through some of the difficult moments in my journey. In particular, Onur has consistently helped me from the beginning of my PhD to develop mature ideas and papers, and his guidance, feedback, and discussions have proven invaluable.

I thank Prof. Dimitrios Pezaros and Dr. Alejandra Beghelli for their feedback and discussions on this work. I also thank Dr. Ioannis Psaras who helped in my initial stage, and Prof. Izzat Darwazeh and Prof. Cyril Renaud for their support in the critical stages of my progress.

I am immensely thankful for the many collaborators I have encountered throughout my journey, helping me shape and challenge my thinking. In particular, I thank Fan Yang, with whom I started my PhD journey at UCL, not only for our academic work, but also for our friendship and reciprocated motivation. I also greatly value Adrian-Cristian Nicolaescu, the other PhD student in our group, for his help, and I thank Dr. Puneet Bindlish for his mentorship.

After 8 years at UCL there are many academics and friends to mention, and I am grateful to the university and department where I spent the formative years of my life, and the many people who shaped me in to the individual I am today. I would like to highlight my colleagues and friends in the lab for helping me throughout my PhD. Additionally, I would never have been able to achieve what I have without the endless support of my friends outside academia. Specifically, I want to thank Dr. Prithika Prasad and her family for all the support, motivation, and for making me feel at home in London.

There have been many teachers in my life in various subjects and activities, who have all contributed to my achievements, and for that I am filled with gratitude. During this period I restarted learning music, and I thank the late Sanjay Guha for his guidance.

Finally, and most importantly, I am eternally indebted to my family for their love and support. My mother Prof. Sharda Nandram and father Wim Keizer always believed in me, motivated me at difficult times, and always enthusiastically proof-read my work. While the COVID pandemic caused much difficulty and sorrow around the world, I am especially thankful that it gave me the opportunity to spend time at home with them. I also value my sister Sharan, as well as my many aunts, uncles, and cousins, who have supported and accepted me. I will also always cherish my moments spent together with my late grandfather, and continue to be inspired by his honesty and emphasis on education. Thank you all.

Publications

- Keizer, N.V., Ascigil, O., Król, M., Pavlou, G., 2023. **Ditto: Towards Decentralised Similarity Search for Web3 Services.** In Proceedings of the IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS '23), IEEE.

- Keizer, N.V., Ascigil, O., Psaras, I., Pavlou, G., 2021. **FLOCK: Fast, Lightweight, and Scalable Allocation for Decentralized Services on Blockchain.** In Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency (ICBC '21), IEEE.

- Keizer, N.V., Yang, F., Psaras, I., Pavlou, G., 2021. **The Case for AI Based Web3 Reputation Systems.** In IFIP Networking Conference (IFIP Networking '21), IFIP.

- Keizer, N.V., Ascigil, O., Psaras, I., Pavlou, G., 2020. **Rewarding Relays for Decentralised NAT Traversal using Smart Contracts.** In Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc '20), ACM.

- Keizer, N.V., Ascigil, O., Król, M., D Kutscher, Pavlou, G., 2023. **A Survey on Content Retrieval on the Decentralised Web.** Under Journal submission.

Contents

List of Figures

List of Tables

Abbreviations

CDN Content Distribution Networks.

CID Content Identifier.

DHT Distributed Hash Table.

DSN Decentralised Storage Network.

DWeb Decentralised Web.

ICN Information Centric Networking.

LSH Locality-Sensitive Hashing.

NAT Network Address Translation.

NRS Network-Resource Sharing.

P2P Peer-to-Peer.

QoE Quality-of-Experience.

QoS Quality-of-Service.

TEE Trusted Execution Environment.

Chapter 1

Introduction

1.1 The Centralisation of Network Services

The inception of the Internet has produced the largest distributed system in the world, consisting of countless interconnected nodes. On top of this network, various services have been created such as the World Wide Web, Cloud, and Email, which have become essential and integral to everyday life [5].

Freedom and autonomy have always played an important role in the Internet. In fact, the original design of philosophy of the DARPA Internet, an early predecessor to the current network, clearly defines the distributed management of resources as a key principle [6]. While the Internet infrastructure has remained distributed on lower layers, the applications built on top of them have become increasingly centralised, taking away from this design philosophy and user freedom.

In the past decade society has grown to be highly dependent on these centralised services, increasing the importance of them staying operational at all times, as well as being secure and private. This becomes especially apparent when looking at content retrieval on the Web, which accounts for the majority of Internet traffic (video streaming alone was predicted to reach 82% in 2022 [7]).

Content retrieval is comprised of a number of procedures to deliver relevant content to a user, which all have increased in the level of centralisation. For example, the entry point for accessing content for most users are *search engines* such as Google. They are able to optimise performance, but suffer from centralised control and a lack of privacy, leading to a power imbalance with users. Furthermore, since their business model is profit-centric, they keep their service free by monetising user data through advertisements [8].

Another prime example of the centralisation of services is the *Cloud* [9, 10], which allows for easy and cheap storage and computation. This not only includes personal storage, but also hosting Web pages and applications, bringing a high degree of centralisation to the Web.

The Cloud allows for cheap storage at distant servers, which incentivises both business and individuals to use this service to store and fetch large amounts of data. However, this approach is not feasible for all use-cases, especially in low-latency applications like video streaming, where the delay and throughput constraints are much more stringent compared to regular files. For this purpose, *content delivery networks* (CDN) [11, 12] were established, allowing for data caching much closer to the user, providing a higher *Quality-of-Service* (QoS) and *Quality-of-Experience* (QoE) [13, 14].

CDN's allowed for much more decentralisation of content storage, and therefore solved many of the problems associated with centralised infrastructures. However, they achieved a degree of infrastructure decentralisation while keeping management centralised (*i.e.* companies still have complete control over the infrastructure, at the expense of users).

Looking to the future, it is important to look beyond traditional CDN's to keep providing satisfactory QoS/QoE with many new users, as well as keep their personal data secure and private. Decentralised services have the potential to solve some of the challenges of current infrastructure.

1.2 The Need for Decentralised Services

While the current host-centric content retrieval ecosystem on the Web has several advantages, mainly in terms of performance and cost, it is exposed to serious flaws and vulnerabilities, due to the centralisation in control and ownership of the entities involved (*e.g.* search engines, DNS, content storage). Users are expected by default to trust these centralised entities unconditionally, forming *a centralised trust model*.

When a single platform controls the majority of the market, a power imbalance is created between the platform and their users [15]. This may impact society as a whole as these tech giants are increasingly gaining civic power [16], which includes the power to influence votes, enable collective action, communicate news, command attention, give people a voice, and hold power to account.

This power imbalance enables tech giants to abuse their power, fueled by their desire to keep their leading market share and their profit-oriented approach. For example, it is made difficult for users to migrate to new platforms, and therefore this prevents new initiatives from entering the market, limiting innovation and competition [17, 18]. Incentivising users to continually use big tech services such as social media may also promote addiction and have a negative effect on mental and physical health, depending on users [19, 20]. Another example of this is online gaming, where the intention of platforms to make money could promote health disorders and leave users in financial distress [21].

Furthermore, these centralised giants are able to gather enormous amounts of user data without any control, in order to monetise their services through *e.g.* personal advertisements. Their lack of transparency may also lead to a number of other issues. In the case of search engines, the lack of ranking transparency may lead to filter bubbles [22], influencing popular and individual opinion, which in turn may lead to radicalisation [23]. The lack of control may also lead to censorship and bias (*e.g.* by governments [24, 25]), and ultimately control over what users think [26].

On a more technical level, by trusting one party (and sometimes even one machine or server) a single-point-of-failure is created. This not only introduces risks in terms of potential malicious attacks, such as an eclipse or *Denial-of-Service* (DoS), but also may cause a global network outage due to a single human error or natural disaster, rendering a service completely useless [27, 28].

Finally, another reason why centralised systems are not future proof has to do with the rapid increase in the amount of users who are connecting to the Internet, consisting of people who did not have access before, as well as many *Internet-*

of-Things (IoT) devices. Current infrastructure is not compatible with upcoming economies of scale [29, 30, 13].

Recently, numerous projects have begun to adapt and create solutions to the problems introduced by centralisation. These projects have higher degrees of decentralisation and leverage a *distributed trust model*. Rather than being reliant on a single root of trust, these projects use other tools like cryptographic proofs and consensus algorithms to establish trust in an environment where users have equal privileges. Collectively, this movement of decentralising important Web components is denoted by the terms decentralised Web, DWeb, or more common in industry: Web3.

Specifically, the DWeb can be defined as *a collective of projects and protocols which aim to make the Web more decentralised, open, and transparent, and thereby aim to decrease the power imbalance between users and service providers.*

1.3 First Steps Towards Decentralised Services

1.3.1 Distribution in P2P and the Edge

In the early 2000's, researchers increasingly saw the flaws of the centralised client-server model, leading to the creation of *peer-to-peer* (P2P) networks, such as BitTorrent [31], a decentralised file sharing system. These networks marked the first step towards decentralised services in the application layer. In these systems, participants acted as a full node in the network, acting simultaneously as a client and server (this is described in more detail in Section 2.2). Figure 1.1 gives a broad overview of the timeline of progress in decentralised services.

There were a number of issues with their approach however. To start with, Internet infrastructure was set up for the client-server model. This meant there was a presence of many firewalls, *virtual private networks* (VPN), and *network address translators* (NAT), which made it impossible to participate in P2P networks for regular nodes, due to the lack of connection to the public Internet. On top of this, incentives were often misaligned, causing free-riding [32, 33, 34]. This made fair exchange of work for reward impossible, and over time made many of the protocols

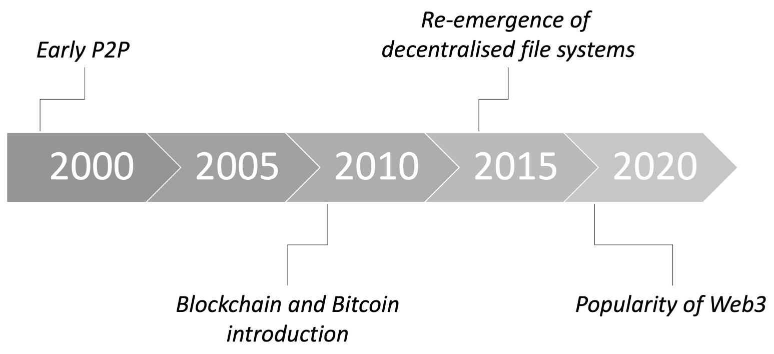
Early P2P
Re-emergence of decentralised file systems
2000
2005
2010
2015
2020
Blockchain and Bitcoin introduction
Popularity of Web3

Figure 1.1: Timeline of progress in decentralised services.

unusable. Other issues in these systems include instability due to network churn, low security due to openness, and difficulty in system maintenance.

Soon it became apparent that pure decentralisation in the form of a truly distributed P2P network was not attainable due to these performance and security issues. Rather than completely decentralising both the management and infrastructure, projects next focused mainly on the latter (*i.e.* infrastructure). From this CDN's emerged, where data was cached closer to end users compared to the distant Cloud, while at the same time being controlled by a single entity to make maintenance and updates easier. This then moved beyond storage with the emergence of edge computing [29, 30, 35, 36], where computation was moved closer to the network edge to accommodate for low latency environments such as real-time remote surgery or natural disaster use-cases, where resource-scarce end-user devices like phones lacked performance.

This type of infrastructure can be argued to only be partially decentralised, as equipment is generally managed by single entities. Even if they are open to participation from the public, it is not a truly open network due to the high equipment investments needed. Although these approaches greatly improved in terms of performance, they still lacked openness and might be prone to censorship and security issues. This can be seen as a less general version of the concepts of the P2P systems, as now there is a barrier to entry, either in the form of hardware or membership of the managing entity. Ideally, a system should be as open as P2P networks so anyone may fairly participate, while keeping the performance guarantees of the edge-based approaches, and having better security than both.

1.3.2 Blockchain Networks

The introduction of blockchains marked a renewal of interest in P2P technologies both from academia and industry. As described originally by Bitcoin [37], a blockchain is an open immutable shared ledger, which is replicated across all members of a P2P network who collectively update the ledger using a consensus mechanism (this is further described in Section 2.3). The original use-case for the blockchain was a digital currency due to their security benefits, as it prevents double spending attacks.

Blockchains have since adapted to more complex use-cases, especially with the introduction of smart contracts [38]. These contracts allow more complex logic compared to simple transactions to be added to the blockchain, forming 'programmable transactions', which could for example include conditions of payment.

With the new interest in blockchain, new initiatives have tackled the problems which P2P systems aimed to solve, but with new tools. The main difference that blockchain has made is that *crypto-economic incentives* can be added in P2P protocols in the form of rewards for useful work, which prevents free-riding. It also adds an extra layer of security to the networks, as participants are seen to have 'skin in the game', and therefore are less likely to attack the system. Although Internet money has been around before blockchains [39, 40, 41], the way blockchain integrates research concepts like cryptography and P2P networks makes it an excellent candidate to power truly decentralised services.

Besides user incentives, blockchain based network services have another advantage over traditional distributed systems: they allow for anyone to offer a service, without making it mandatory to participate for all nodes. An example of this is that users can be simple clients and connect to a full node in the network to participate in transactions, but do not need to keep up the entire ledger themselves, as only full nodes have this task. This structure allows for much more flexibility and openness to participation.

It has been argued however, that many of the same problems that P2P networks faced still remain a problem in blockchain networks [32], especially with permissionless and open blockchains. It is argued that a permissioned (private or consortium) version of blockchain [42], which reintroduces some centralisation in terms of network management is a better suited candidate for decentralising the Internet. Although this might be the case when looking at first generation blockchains, new developments tackle a lot of the challenges such as system maintenance and security. It can be argued that user incentives and the associated elimination of network churn have been key design principles from the start.

1.3.3 Decentralisation of Web Services

The usage of a technology beyond its original use-case is a characteristic of the fourth industrial revolution [43], and the blockchain applications beyond simple payment are no exception. Smart contracts and micro-payments using blockchain started being leveraged to create a shared network economy, where users could share spare network resources, in return for which they would receive rewards in the form of cryptocurrency tokens. Conversely, if they want to use a service they will pay for it in tokens. Examples of these *network-resource sharing* (NRS) services are storage [44], computation [45], and bandwidth [46, 47, 48].

Simultaneously, a renewed interest arose in P2P storage networks, producing a number of novel *decentralised storage networks* (DSN) [49, 50] which leverage a mix of concepts like *distributed hash tables* (DHT) and content-addressing from P2P and *information centric networking* (ICN) [51].

These two concepts (*i.e.* NRS services powered by blockchain and DSN's) together form a solid foundation to create alternatives for Web services. In the current Web, most users access and discover Web content using a keyword-search based workflow. A user generally opens a search engine and submits a number of keywords, specifying the content they are looking for. The search engine returns the relevant results of the domain name pointing to the location of the server which can provide the content.

In a decentralised Web, each step of content discovery and interaction is envisioned to be decentralised and performed collaboratively by peers in the network (*i.e.* decentralised search mechanisms, name-registry, storage and more), incentivised using a reward layer using smart contracts.

1.4 Open Issues and Challenges

While the objective of the DWeb is to achieve decentralisation, it is an open question whether this can be achieved in practice or not. Current Web centralisation is driven by economic concentration, and it is unclear if the same would not happen to the DWeb. Furthermore, interacting with untrusted, anonymous peers requires additional security mechanisms that are difficult to design and can lower the overall performance of the system.

There are a number of challenges to address in order to realise decentralised, blockchain-based services. Such a system needs to be carefully designed, with user incentives aligned, in order to leverage the benefits of decentralisation while not compromising on performance, scalability, and security. First there is the issue of infrastructure barriers to entry, mainly in the form of NAT's, VPN's and firewalls. These restrict participation in the network as a full node. Beyond connection, quick service discovery and fair exchange of work for reward are prominent issues.

Moreover, there is the challenge of finding a reputable service provider. As anyone in the network may provide a service, there is extra care which needs to be taken in order to avoid malicious peers. For this, metrics like trust and reputation may be used to distinguish potential malicious nodes. Trust and reputation systems have been implemented in centralised systems like e-commerce, as well as in distributed P2P networks, but they are yet to be optimised for blockchain networks. Finding content on the DWeb is also challengeing as current models leverage one or more trusted parties in the process. Throughout the rest of this thesis, open issues are discussed in each section.

1.5 Research Question and Hypotheses

In order to thoroughly study decentralised content retrieval on the decentralised Web, this work aims to answer the following *research question*:

> **Research Question:** How can novel P2P solutions and blockchain be leveraged to create decentralised counterparts to centralised Web services (especially in content search and retrieval), without compromising on key features such as security and performance?

In order to answer this research question I have formulated the following core *hypotheses*:

> **Hypothesis *H1*:** Decentralised network resource sharing services can be designed using blockchain-based incentives, in order to overcome limitations of both traditional P2P systems, as well as centralised services.

> **Hypothesis *H2*:** A decentralised Web can be realised with the aid of network resource sharing services, blockchains, and decentralised storage networks, decentralising the major components of content retrieval on the Web (search, name resolution, storage).

> **Hypothesis *H3*:** While a decentralised Web can be realised, it is not trivial to ensure satisfactory QoS/QoE for users, especially in terms of security, privacy, and performance.

The *research question* and *hypotheses* can be used to identify a number of sub-goals, which will be referred to throughout this thesis to highlight and frame the contributions. These sub-goals will need to be realised in order to answer the research question and test the hypotheses.

> **Goal *G1*:** Develop a framework for studying and integrating decentralised Web services and protocols, and use this to gain insights on the components of content retrieval in a DWeb environment.

Goal *G2*: Create a decentralised content search mechanism which allows users to interact with DWeb content, accommodating for both human-readable and self-certifying names, as well as a variety of content and underlying protocols.

Goal *G3*: Create methods for peers to find others who are able to provide a service, in a manner that is similar in performance to centralised solutions, while ensuring that there is a mitigation of malicious nodes.

Goal *G4*: Create protocols which allow for fair exchange of useful work for rewards in cryptocurrencies. The amount rewarded should be based on the performance of a node in terms of QoS. This may be done for example using micro-payments.

Goal *G5*: Develop network resource sharing services with a careful consideration of crypto-economic incentives. This needs to keep nodes honest, deter malicious attacks, and make the overall network useful.

Goal *G6*: Allow for peers in a network to identify malicious and honest nodes by keeping metrics of each other, thereby forming a trust and reputation system.

Goal *G7*: Study and develop solutions to connectivity problems in P2P systems, in order to ensure openness and security of decentralised NRS services.

1.6 Methodology

To answer the research question, test the hypotheses, and achieve the sub-goals mentioned above, a number of research methods are used. First of all, a relevant body of work in decentralised Web was surveyed. This includes works from academia starting from the distributed systems, P2P era, and content retrieval, up until recent works on blockchain, DWeb, Web3, and DSN's. Furthermore, a number of techniques from cryptography, reputation, mathematics, and game-theory were examined to be used in system design.

Besides academic literature, industry work was surveyed as the field is rapidly changing and a lot of developments happen outside academia. Particular attention was paid to works which were cited in academic sources in order to curate a high quality body of work, and they were manually inspected to verify correctness and objectivity.

Second of all, complete system solutions were designed to tackle one or more of the sub-goals highlighted above, and these systems were analytically studied and evaluated, partially through simulations. For example, smart contracts need to be tested to make sure they work properly and determine their scalability and cost. On the other hand, large P2P simulations can be performed to ensure a system reaches an equilibrium, or to assess whether effective cooperation between peers is possible.

Besides these practical simulations, more theoretical approaches, as well as comparative evaluations were used, directly comparing the proposed systems to results found in other works to draw conclusions. Finally, real world network measurements allowed for testing viability of an idea, or to more clearly understand the problem that needs to be addressed.

1.7 Contributions

This thesis includes work ranging from creating a framework for the DWeb and creating alternative systems for Web retrieval systems, to tackling networking and resource allocation problems. Specifically, I make the following contributions:

Contribution 1: Introduce a framework for studying DWeb initiatives, in order to define a global view and vision for interoperability. This is used to comparatively analyse current projects and systems, and define open issues and challenges [52]. This satisfies sub-goal *G1*.

Contribution 2: Analyse current decentralised search mechanisms, and design and implement a decentralised crawler-indexer style search engine for a DSN [53]. Furthermore, based on the lessons learned, design, implement, and evaluate a truly decentralised search system based an alternative search-workflow using similarity search [54]. This satisfies sub-goal *G2*.

Contribution 3: Design and evaluate an allocation mechanism for NRS services, which is highly scalable, fast, and lightweight. This uses stable matching algorithms and improves on auction-based approaches on smart contracts [55]. This satisfies sub-goals *G3*, *G4*, and *G5*.

Contribution 4: Analyse methods of establishing trust and reputation in decentralised environments and NRS services, resulting in the design of an AI-based personalised reputation system using blockchain data [56]. This satisfies sub-goal *G6*.

Contribution 5: Analyse networking problems in novel P2P systems such as DSN's, and expose security vulnerabilities such as content-eclipsing attacks. Develop a decentralised NAT traversal system, which uses smart contracts for rewards and reputation to select reliable relays [57]. This satisfies sub-goals *G4*, *G5*, *G6*, and *G7*.

1.8 Thesis Outline

The rest of this thesis is organised as follows:

Chapter 2 introduces preliminary concepts and classifies the literature used in this thesis. This consists of P2P networks, blockchains, network resource sharing services, and reputation systems.

Chapter 3 gives a thorough overview of content retrieval on the DWeb by analysing current works from industry and academia using a defined framework for DWeb initiatives. Using this, a number of open issues are defined in the key DWeb components of search engines, name-registries, and file systems.

Chapter 4 explores decentralised search mechanisms for DWeb content. A search engine based on similarity search is proposed which captures a range of use-cases including keyword-search, and is agnostic to underlying blockchain or storage network assumptions.

Chapter 5 presents an allocation system which is highly scalable, fast, and lightweight, targeted at NRS services. The system leverages stable matching algorithms, as well as smart contracts and trusted execution environments.

Chapter 6 focuses on connection issues, and particularly NAT traversal, in P2P networking which affect DWeb services. A decentralised NAT traversal mechanism is introduced which relies on network peers functioning as relays, who are incentivised to contribute using blockchain rewards.

Chapter 7 summarises this thesis and describes a number of directions for future work.

Chapter 2

Preliminary Concepts

This chapter introduces a description and classification of the literature used in this thesis, after which preliminary concepts are discussed. Particularly, an overview is given on peer-to-peer networks, blockchain, network resource sharing services, and trust and reputation systems. Figure 2.1 provides a broad overview of the key preliminary concepts discussed in this chapter.

2.1 Classification of Literature

The background literature can be classified in a number of ways. First, there are related technologies which power DWeb solutions, which include blockchain, reputation systems, and more. These are broadly discussed in this chapter, as they form the base for many of the DWeb services, and are referred to later throughout the text. On the other hand, concepts, theory, and implementations of the decentralised services are further discussed in Chapter 3.

Another way that literature can be subdivided in decentralised services is based on time-periods. First, there is the P2P era, where there are mainly fully distributed collaborative services based on an altruistic model, and without much attention to monetary rewards. While these systems have many shortcomings, they should be closely studied in designing current systems in order to overcome well known challenges. Second, there is the blockchain era, which is characterised by a distributed shared ledger which is used to store transactions and other data. Among these are also NRS services such as storage, as their design principles and assumptions of-

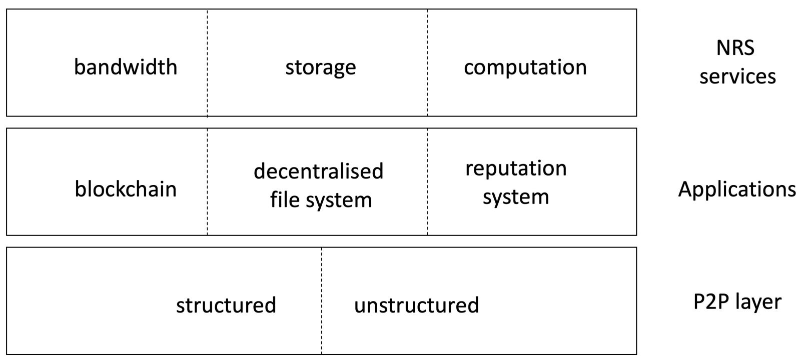

Figure 2.1: Overview of key preliminary concepts.

ten align, as well as their underlying technologies. Here, the area of economic incentives is much more prominent, as incentives can be modified to protect against malicious behaviour, leading to unusable systems.

2.2 Peer-to-Peer Networks

Peer-to-peer (P2P) networks emerged as alternatives to the client-server model, where a network of many nodes pools their resources together as equal peers without central control or orchestration, in order to provide a distributed application architecture (originally for content storage and delivery) [58, 5]. In P2P networks, network tasks and workloads are partitioned between peers, who are equally privileged participants in the application and can simultaneously listen for request and send queries. These peers share resources such as storage, bandwidth, or computation with the rest of the network, alleviating the need for centrally managed Web servers and services, based on altruism [59].

P2P networks are realised by forming a virtual overlay network on top of the physical network topology. Nodes in the overlay form a subset of the nodes in the physical network, and geographically close nodes are not necessarily close in the overlay routing. Peers form connections with other nodes in the network, who become their *neighbours* in the topology. When routing requests, a peer relies on their neighbours who forward the request to their neighbours (who are relevant in finding the result), which continues for multiple hops until the request is satisfied.

Exchange of data still relies on the underlying network (*i.e.* TCP/IP), but peers can communicate with each other directly using the overlay links in the application layer. These overlays are also used for tasks like indexing and peer discovery, making them distinct from other network topologies. Most DWeb initiatives leverage P2P networks, as they provide a solid base for sharing resources and providing application infrastructures in a decentralised environment without a trusted centralised entity.

There are two main types of P2P networks based on the overlay implementation: *(i)* unstructured and *(ii)* structured.

2.2.1 Unstructured P2P Networks

In unstructured P2P networks, the overlay is not designed to form a particular structure of connections between peers. Instead, peers form random connections with other nodes in the network in order to form their view of the network [60, 61, 62]. Generally, unstructured P2P networks are easy to implement and are robust to churn, as peers do not require a global view and constantly update their neighbours.

Querying for content is more challenging in unstructured networks. In early works, flooding was the most prominent way of sending queries [60], which involved sending the query to all neighbours, who would do the same. While this maximises the chance of finding a peer who has the content, it is also highly inefficient and unscalable, as its overhead on the network grows linearly with the number of search queries. Furthermore, in terms of performance, popular content may be found fairly quickly if many peers have it cached, but if the content is less popular high average latencies are to be expected to fetch these items. More recent unstructured P2P systems use slightly more scalable search mechanisms such as random walks [63] and scoped flooding [64] (some of these works are discussed in Chapter 3), but they still do not match the performance of structured networks in ensuring unpopular content is found quickly.

2.2.2 Structured P2P Networks

Structured P2P networks differ from unstructured network as their overlay is organised into a specified topology. Because of this, any node can efficiently search the network for content, even if the resource is extremely rare. Most commonly, structured P2P networks are implemented using a distributed hash table (DHT) [65, 66, 67, 68, 69].

A consistent hashing algorithm is used to assign all content to a particular peer among the network, based on a distance parameter. For example, in Kademlia [65], the most popular DHT implementation, nodes are assigned ID's on a 160-bit keyspace envisioned as a binary tree, by creating a random public/private key pair. To assign responsibility for a {*key:value*} pair to the closest node the XOR distance metric is used, which uses longest prefix matching between peer and key. The peer closest in the keyspace to the content becomes responsible for maintaining the record. Now, peers can easily search the network using a hash table, allowing for efficient retrieval of content (usually in $O(log(n))$, where n is the number of peers in the network).

Maintaining the overlay of organised peers makes structured networks less robust in networks with a high rate of churn. Maintaining a structure also exposes the network to vast range of attacks that can be more difficult to perform in an unstructured P2P network [70]. On the other hand, while this approach adds more overhead, it improves performance, especially the recall of rare content.

2.3 Blockchain

2.3.1 Overview

Blockchain and its use as a distributed ledger technology was originally proposed in the Bitcoin whitepaper [37]. Its original use was as a digital currency, which was resilient to double spending attacks, due to the immutability and security of the digital ledger. Since then however, blockchains have been extended and used in various use-cases.

A blockchain is built on top of a P2P network and consists of a large number of nodes which all have a full view of the shared history (the chain of blocks of transactions). This chain is append-only, which avoids double spending. Each full node in the network keeps a copy of the entire ledger, leading to redundancy and security, as now 51 % of the network would need to be malicious in order to propagate an incorrect view of the ledger.

Nodes in the network send transaction messages to all other peers using a flooding protocol, and these nodes aggregate all incoming messages in order to produce blocks. A consensus algorithm determines how these blocks are produced [71]. Proof-of-Work (PoW) [72, 73] for example does this by letting all nodes compete to mine the next block using a cryptographic puzzle. Nodes hash all transactions together in a Merkle root, and combine it with the hash of the previous block on the chain. They then add a nonce (random integer) to this and hash the values together in order to produce a new hash. If this hash contains a preset number of preceding zeros, it is considered the next block. Miners keep repeating the calculation with different nonces until they find this block, after which they broadcast it to the network, which then validates its correctness. Mining and consensus protocols vary with different implementations of blockchains.

Besides PoW there are number of other consensus algorithms [71] which may be used to maintain the chain. Proof-of-Stake (PoS) [74, 75, 76] is the most popular, and it assigns a higher probability of producing the next block to users who have a higher stake in the system (in terms of cryptocurrency collateral). Its main improvement over PoW is that it does not waste as many resources by performing useless work to calculate the cryptographic puzzle, and does not consume as much energy. It also lowers participation barriers as any user can participate in consensus and earn rewards, without expensive computational setups. PoW has become increasingly centralised due to a small number of nodes controlling a large amount of computational power, lowering network security. Other consensus algorithms include Proof-of-Storage [77, 78], Proof-of-Authority [79], and Nominated-Proof-of-Stake [80, 81].

Beyond simple payments, smart contracts were introduced by the Ethereum

[82] blockchain. These allow for complex logic (*i.e.* scripts of code) to be added on-chain [38]. In Ethereum, these contracts can be defined by users with various conditions in the Turing-complete Solidity programming language, after which it can be deployed by compiling it to the Ethereum Virtual Machine, which results to the code to be added to the chain under its own address which is callable by nodes. Executing functions defined in the smart contract has an associated gas cost, which is proportional to the added load on the network, as well as a time-constraint set by the user. This monetary gas cost needs to be paid by the user for the computation, regardless of success of the transaction. Gas is paid in Ether, the cryptocurrency used in Ethereum, and the amount depends on the time constraints of the transaction. In order to write scalable and user friendly smart contracts, the interaction with the smart contract needs to be kept to a minimum, as every function call results in gas costs. Furthermore, storing data on-chain in a smart contract becomes extremely expensive. Therefore, using a blockchain as a data storage source is not feasible. Smart contracts also have difficulty interacting with data off-chain, although this can be solved using data oracles [83, 84, 85].

Smart contracts can be used to implement conditional payments, thereby creating decentralised applications (DApps) [86, 87]. Blockchains are especially useful in DWeb architectures due to their ability to incentivise users to participate and contribute to the network or DApp by paying them rewards in cryptocurrencies or tokens. However, blockchains are only able to ensure a fair exchange of reward for work, if the resource contributors can produce verifiable proofs of resource consumption for *useful* work. For example, a node can prove that bandwidth [88], computation [89, 72], or storage [90] resources were actually provided or used for system up-keeping, and a subset of the participants in the system can collectively verify these proofs, which can then trigger automatic rewarding of contributors for their valid proofs. Section 2.4 will discuss how NRS services are implemented further.

Proving useful work is not always straightforward, for instance, for continuous services that take place for a period of time. When such proofs are unavailable, beneficiaries may issue periodic payments to contributors (at the end of fixed or increasing time intervals) as long as the provided service is satisfactory. However, if the counter-party is malicious, it could lead to a loss of revenue for at least one interval, and the absence of penalties for malicious behaviour may encourage more nodes to behave undesirably. In this case, trust and reputation systems are useful, as described in Section 2.5.

2.3.2 Permissionless and Permissioned Blockchains

Blockchains were proposed as open systems, where anyone can join without compromising on privacy. Although it has been shown that Bitcoin is only pseudonymous [91] and personal data can be inferred from on-chain data, it remains open and permissionless. This means that anyone can join at any time without any barrier to entry. It has been argued that permissionless blockchains have some inherent issues [32], similar to P2P based solutions, which could make them unsuitable for creating a decentralised Web. It remains to be seen whether this will be the case when blockchains evolve and adopt scaling solutions (Section 2.3.4).

As an alternative to these open systems, permissioned blockchains [92, 93] emerged, allowing entities to control the blockchain and exercise more control over who entered the network. While this takes away from true decentralisation, it is more attractive to corporations and governments as they retain control over access management and identities [94, 95]. Most notably Hyperledger Fabric [96] offers a modular architecture for implementing these types of blockchains. As permissioned blockchains take away from openness, privacy, and decentralisation, and add a component of control, the rest of this work is mainly focused on permissionless blockchains.

2.3.3 Blockchain Challenges

Current blockchains have a number of inherent challenges, which need to be overcome in order for them to reach their true potential. As characterised by Vitalik Buterin, blockchains suffer from the scalability trilemma [97, 98]. This means that out of the following three properties, they can achieve at most two at a time. These properties are: *scalability, security,* and *decentralisation.*

The scalability problem has proven to be a performance bottleneck for many projects. However, decoupling the blockchain layer (*layer 1*) and scalabillity solutions (*layer 2*), as described in Section 2.3.4, seems a promising development which may solve this issue. Another remaining challenge is that of security and privacy [99, 100]. As mentioned before, blockchain transactions can expose a significant amount of user data. To tackle this a number of projects such as Zcash [101] use zero-knowledge proofs [102], leading to better privacy guarantees. Furthermore, there are a number of attack vectors such as sybil, eclipsing, and phishing attacks which may severely impact the blockchain security. Centralisation of mining power is another known issue possibly undermining the security of the chain, and key management issues may lead to loss of funds or ownership.

Finally, there are a number of lower layer challenges related to blockchain. First of all, there is the connectivity issue, which causes participation difficulty for users due to current Internet infrastructure. Specifically, nodes who are behind a NAT or firewall are not able to participate as a full node in the P2P network, and therefore a large portion of the network is only able to function as a (light) client node. Second, transaction and block propagation in networks like Bitcoin are quite inefficient, as they generally use a flooding approach. Recent work aims to make the underlying communication layer more efficient by using alternative message propagation techniques [103, 104]. Other known issues in P2P networking also apply, which includes churn, free-riding, and system maintenance and upgradability.

2.3.4 Next Generation Blockchain

While it could be argued that first generation blockchains are not suitable for a decentralised Web as they suffer from instabilities and scalability limitations, next generation blockchains are expected to overcome these issues.

As the field of blockchain is relatively young, it is still under very active development. As such, many projects are working on solving the challenges of Bitcoin and Ethereum. For example, in Ethereum 2.0 [105] this is achieved in two steps. First, like many systems it's moving away from the expensive PoW mechanism towards PoS-based consensus. On top of this, sharded chains [106] are leveraged to achieve scalability at the blockchain layer. Furthermore, layer 2 solutions [107] are used, which are solutions allowing for scalability off the main chain. These include optimistic and zero-knowledge rollups [108] and payment channel networks (see Section 2.3.5).

Other big projects follow a similar trajectory of achieving scalability. Polkadot [1] uses a parachain approach, where a number of individual chains are linked together using a relay chain, which is governed using Nominated Proof of Stake (NPoS), allowing for a higher transaction throughput [80, 109, 110]. Cardano [2] uses Ourobouros [74], a modified version of Proof of Stake.

2.3.5 State Channels

Payment channels are one of the scalability solutions which has been implemented. As cost of transactions need to be low for any blockchain system to be usable, especially when lots of smaller micro-transactions are sent as is the case with continuous, time-bounded services, payment channels present a solution to keep fees low.

Instead of on-chain payments, which form a performance bottleneck, payment channels only require a fee for an initial deposit and withdraw, allowing any intermediate transactions to be sent free of charge. This is achieved by sending payment receipts, which allow the recipient to cash them in at any point on-chain. These receipts keep track of the balance between both parties, and any party can send as many as their initial deposit allows. Payment receipts can be hash locked, conditional, or optimistic, allowing more security on the client side. The interval (*i.e.* time window) upon which receipts are sent, for example for a service, can also be adjusted based on the trust or reputability of the counter party.

A more general extension to payment channels are state channels [111]. They allow for the execution of any arbitrary application on the channels, bringing the functionality of smart contracts off-chain and extending channels beyond simple

[1]`https://polkadot.network`
[2]`https://cardano.org`

payment transactions.

A further extension comes in the form of payment channel networks [112, 113, 114, 115, 116]. An example of an implemented payment channel network is the Raiden Network [3] for Ethereum. These rely on payment channels, but extend them with cross-channel payments, allowing for transactions between nodes who do not have a direct channel. These are also called virtual channels. A locked transaction is routed through a path of nodes who have payment channels with enough capacity to support the payment amount. The receiver requests the secret to unlock the payment, which it sends to the closest node (last hop) to receive a balance proof of the payment. This happens between all other nodes on route to finalise the payment. When nodes become unresponsive, the payment can be settled on-chain, using the blockchain as an arbitrator.

As payment channel networks are realised using a type of unstructured overlay network where all channels have a capacity, a number of works [117, 118, 119, 120, 121, 122, 123] have worked on optimising the routing process in terms of privacy, security, and routing efficiency. This can be achieved using a global view of the network to calculate optimal paths and using algorithms like Breadth-First Search (BFS). However, gathering a global view is difficult without the use of one or more privileged peers. The other option relies on routing through random paths. In Flash [124] larger payments are sent over optimal paths and smaller transactions use random static paths.

2.4 Network Resource Sharing Services

An important aspect of a decentralised Web is the ability to outsource tasks to spare resources, creating *network resource sharing* (NRS) services. Sharing network resources in a decentralised network isn't a new concept, but what makes DWeb initiatives unique is their integration with blockchains to create an incentive layer, as well as their focus on decentralisation of control and security (rather than scalability and performance of earlier P2P works). These services are essential as central

[3]`https://raiden.network`

servers (e.g. the Cloud) need not be blindly trusted.

As discussed above, classical peer-to-peer (P2P) systems suffered from a number of problems in the long term, including, free-riding, instability due to churn, and security vulnerabilities. Furthermore, these systems either lacked incentives to do so, or these incentives were not strong enough to keep users honest. By providing a fair exchange for performed work in the form of cryptocurrency rewards, blockchain-based NRS services add incentives, security, and robustness.

NRS services can broadly be classified as *storage*, *computation*, or *bandwidth* sharing services. A service may also target all of these, as is the case for decentralized content delivery networks.

In terms of storage, there are many P2P based systems that focused on producing DHT based storage networks. Most notable of these was BitTorrent [31]. Tit-for-tat was used to deter users from free-riding, and the protocol became widely adopted. However, BitTorrent faced a number of issues such as instability due to churn, security, and system complexity. Furthermore, its main use became the distribution of unlicensed products, leading to copyright and legal issues.

Recently, a number of projects aim to tackle the flaws of previous attempts, and create usable decentralised storage networks (DSN). For example, IPFS [50] is a storage network which is based on a DHT, but differs from previous attempts by using content addressing. This means that content is hashed when added to the network, and anyone in the network can fetch it from anyone who serves the file, regardless of their location. It also ensures persistence of content, which is not the case currently, for example when a file is moved to a different location. Novel DSN's are further covered in Section 3.6.

IPFS does not have built in incentives mechanisms to encourage nodes to store files. To achieve this, Filecoin [44, 125] was built as an incentive layer, forming a decentralised storage market. Filecoin is a prime example of a NRS service. It uses a blockchain, and allows for value transfer in FIL cryptocurrency on-chain, without the need for external trusted parties. On top of this, Filecoin requires storage nodes to send proofs of storage on predetermined intervals, which earn them rewards. Storage deals are also made on-chain using an orderbook approach. On top of this, there is a secondary retrieval market, which allows nodes looking for content to use an intermediary retrieval miner to find and serve the content, earning micropayments. These deals are completely off-chain. However, due to the large system requirements, Filecoin mining has become increasingly centralised.

When it comes to bandwidth sharing, a number of initiatives, both from research and industry, have aimed to incentivise users to sell spare bandwidth in return for cryptocurrency rewards. Ghosh et al. [88] use TorCoins to reward users who share their bandwidth as relays in the Tor network. In order to verify that claimed rewards are proportional to users' work, a proof of bandwidth is used. Contributors in a circuit are able to claim rewards based on the end-to-end QoS measured over the whole path.

A number of recent initiatives aim to create decentralised Virtual Private Networks (VPN), by allowing users who have spare bandwidth available to sell it in return for cryptocurrency rewards. These include Mystereum [46], Orchid [47], and Sentinel [48].

Finally, decentralised computation outsourcing can be enabled by blockchain. Similar to edge computing, a node with a heavy computation task may offload to the network in return for payment. In contrast to edge computing, any node with spare capacity is able to perform and assist on the task. Golem [45] uses Ethereum and an ERC20 token to store computation off-loading deals and pay for services. Examples of computation which can easily be outsourced include machine learning tasks, video transcoding, and simulations.

In decentralised systems, any participant can create and control an identity without the involvement of a trusted third party. This makes it possible for malicious nodes to simultaneously use multiple identities as part of a Sybil attack. By generating multiple Sybil identities (who pretend to be genuine users), malicious parties can trick a fair exchange mechanism into issuing undeserved rewards, for instance by inflating the amount of actual resources consumed by the node. To prevent such attacks, proofs of resource consumption must be Sybil resistant.

2.5 Trust and Reputation Systems

While the blockchain can be used to establish trust for transactions on-chain, the actual NRS service is provided off-chain and occurs directly between two parties. This means that one cannot rely solely on an honest majority of the network for security. A simple illustration is a provider node which promises a service, but is not able to complete the service. While it does not gain extra rewards, the client may experience additional negative consequences. As any node in the network is potentially malicious there is a risk with every deal.

To discern between honest and malicious parties a trust and reputation system can be used. Generally, a reputation system is a mechanism which produces a score for nodes in a network, indicating the trust in a likely positive experience with them. Trust and reputation metrics play an important role when making transactions on the Internet. Initially, these were mainly used for e-commerce purposes, where customer ratings played an important part in identifying a trustworthy party to buy from, as well as a trustworthy product [126]. These reputation systems were centralised, and often controlled by the same entity who controlled the sale platform, who sometimes were the seller themselves. This created security vulnerabilities, which is why decentralised reputation systems have risen in popularity.

Distributed reputation systems emerged in parallel with P2P systems, and many of them were built on top of P2P overlay networks. There was a wide range in their implementation and score calculation functions. Some used simple additions of positive scores, while others used more complex logic. Metrics used in the calculation could be either public (available to the whole network) or private (metrics only based on a node's own experiences), or a combination of the two. Metrics and scores could also be (partially) shared to others in the network [127, 128, 129, 130]. However, these ultimately did not reach mass adoption due to their complexity and security vulnerabilities.

While decentralised reputation systems can be susceptible to the Sybil attack described above, researchers have proposed reputation systems that can identify Sybil nodes through specific mechanisms, which include voting [131] and social network analysis [132]. These mechanisms are able to identify outlier nodes as Sybils in the presence of an honest majority. For example, when a node lacks connections to other honest nodes in a social network, it can be seen as a sign of potential Sybil activity.

An interesting use-case for reputation systems are blockchain services [133, 134, 135], as two parties transact and need a measure of trust. Dennis and Owenson [136] present an interesting opportunity, to use blockchain transactions on-chain as public metrics in a distributed reputation system. Decentralised trust and reputation systems and their integration with blockchains is further discussed in Section 5.4.

Chapter 3

The Decentralised Web: Overview and Framework

This chapter gives a thorough overview of content retrieval on the DWeb, and explores if the decentralisation objective of novel initiatives is realised by investigating properties such as *incentive structures, performance, security and privacy*. To analyse DWeb content retrieval, this chapter identifies a framework which is analogous to content retrieval on the current Web. This starts with *decentralised search engines, decentralised name-registries*, and finally *decentralised file systems*.

These areas are identified as focus areas, for which decentralised alternatives will need to be developed. This chapter first describes the status quo, *i.e.* how operations are performed in the current Web. This is then compared with state-of-the-art decentralised implementations and proposals from both academia and industry. The insights gained are used to formulate a number of open issues.

3.1 Introduction to the Decentralised Web

In recent decades, the World Wide Web has become a fundamental and integral part of everyday live. The Web not only supports the global economy and provides entertainment, it is also often the main source of information about the world [137]. Additionally, peoples views, opinions, and choices can largely be impacted by the Web [138].

While originally designed as a decentralised network of equal peers, the Internet has evolved into an increasingly centralised system over the past decade, with a handful of players (*e.g.* tech giants like Google or Apple) controlling the majority of the market [139]. While centralisation allowed these giants to provide outstanding services and QoE for users for free, their centralised model of service delivery has introduced several drawbacks such as lack of transparency [140], lack of privacy-protection [141], a single point of failure [27], and censorship [142]. The challenges introduced by centralisation have been discussed in Section 1.2.

Recent initiatives in research and industry aim to tackle these issues by creating an open and decentralised Web (DWeb), also known as *Web 3.0* or *Web3*. As mentioned in Section 1.3, this movement aims to fix the problems that come with centralisation, by focusing on openness, security by design, and decentralised governance and control. This is achieved by using transparent, open source software, as well as using P2P networks [143], which allow anyone to join and contribute to the system. Tools like blockchains [144], proofs of work [71] and self-certification through content addressing [50] form important building blocks to establish trust between anonymous users and reliably reward system contributors.

The objective of the DWeb is to achieve decentralisation, in terms of redistribution of ownership and control from centralised infrastructures to individual users. However, it is an open question whether this can be achieved in practice or not. Centralisation of the Web is caused by economic concentration, and it is unclear whether the same would not happen to the DWeb. Furthermore, interacting with untrusted, anonymous peers requires additional security mechanisms that are difficult to design and can lower the overall performance of the system. Finally, the current Web retrieval model is derived from services that monetise through advertisements, and usually are able to deliver high QoS and QoE to users without monetary compensation. Although end-users do not directly pay for these centralised services, service providers collect user-related data and display targeted advertisements, making the ecosystem economically viable [145]. To be successful, the DWeb would have to reward service providers and content creators, while combating users' reluctance to spend money.

To overcome the challenges of centralisation, a number of decentralised initiatives have emerged that offer decentralised counterparts of various components of the Web, starting initially with content storage systems. Over the years, decentralised file systems have gained increasing popularity, with a major file system recently reportedly serving tens of millions of content requests per day [146, 147]. While technically decentralised file systems are built on top of DSNs, throughout this thesis they are used interchangeably, as they have become synonymous. Besides **(1)** decentralised file systems, two other important components have recently emerged: **(2)** decentralised search engines and **(3)** decentralised name registry infrastructures. These three components together form the foundations for a *Decentralised Web* (DWeb), because they form the key pillars of popular content retrieval workflows. This chapter analyses research trends and emerging technologies used for decentralising content retrieval on the DWeb.

Several open issues are highlighted throughout the chapter which need to be overcome in order to realise a truly decentralised Web. For search engines, achieving good performance (comparable to current centralised search engines) without sacrificing decentralisation is a major challenge. Furthermore, decentralised file systems achieve secure content retrieval in a decentralised manner, but they face both usability (*e.g.* lack of human-readable names), performance, and privacy issues. Finally, decentralised name registries can solve some of the challenges of file systems, but require thorough security analyses and can possibly be extended with better governance and crypto-economic incentive mechanisms.

The rest of this chapter is structured as follows. Section 3.2, describes how the literature was collected to curate a high quality body of work, after which Section 3.3 gives an overview of Web content retrieval and present a systematisation framework used for structuring this work. Section 3.4 discusses search engines, after which Section 3.5 analyses name registries. Section 3.6 examines decentralised file systems, after which the chapter is summarised in Section 3.7.

3.2 Literature Collection

In order to provide a comprehensive overview of DWeb initiatives from a content retrieval aspect, a large body of work was surveyed and analysed. This section explains the scope of the literature, methodology of collection, and related work who surveyed DWeb related topics.

3.2.1 Scope

While the documentation of novel industrial DWeb projects is often scarce, their underlying concepts are usually derived from an extensive body of research, which is utilised for background and comparison to recent projects. The main body of work spans the time period 2009-2022, and focuses on projects that have produced working implementations, as well as research proposals. While an overview is provided of how components are handled in the current Web, there is no analysis of specific centralised solutions, except when this is appropriate for comparisons.

Furthermore, this chapter highlights architectures, their properties, and their aims. However, it is too early to definitively conclude that they are able to live up to their claimed potential, and this nuance has been added in the open issues section. This work mainly serves as a general analysis of the DWeb at large, a framework for analysing and implementing new initiatives, and the first comprehensive body of work looking at DWeb technologies and their role in content retrieval analogous to the current Web. This work is therefore relevant both to industry practitioners and researchers who aim to get a better understanding of the field at large.

3.2.2 Methodology

In order to survey a relevant body of work, research search engines (*e.g.* Google Scholar) were queried for works which contain {*decentralised* + *Web*} in their title, keywords, or abstract. Further queries for {*distributed* + *Web*} were added, which generally returned works of the prior P2P era. These early works were used in conjunction with general {*Web* + *content retrieval*} works, in order to identify key components and determine the framework. Key components of the framework were also inspected using keywords for {*search engine, name registry, file system*}, and works were surveyed which combined these components with keywords like {*Web3, blockchain*}.

Besides academic works, industry works were surveyed, which included white-papers, yellow-papers, and blog posts. Here, particular attention was paid to works which were additionally cited or extensively studied in academic works, in order to curate a high quality body of work without marketing focus, obscure or incorrect jargon, and over-optimistic claims. To verify quality, sources were manually inspected and selected. The reason industry works were included and highlighted is because the area is still rapidly developing and many concepts have not made it into the formal research stage yet. However the underlying technology was always inspected and third-party sources were checked in order to ensure objectivity.

Many of the discussed industry platforms lack clear documentation and a vision of integration to realise a DWeb content retrieval model. Furthermore, terms used in their documentation differ greatly across projects and the fast development pace in the field makes obtaining a clear view and deep understanding challenging. This work aims to clear up some of the contradictions and confusion. By defining a clear framework, this chapter helps to provide a big picture in order to understand and define future research opportunities.

3.2.3 Related Work

To the best of my knowledge, this work is the first to provide a holistic view of the technologies that are useful for decentralised content retrieval. Although the main focus of this chapter is on the recent works (blockchain-era technologies) related to Web3, notable P2P-era research is also covered which introduced the key concepts used by the next-generation decentralised content retrieval systems.

In one of the earliest works on Web information retrieval, Kobayashi et al. [148] survey the content retrieval technologies in the early Web (*i.e.* Web1.0) when it was only few years old. In this work, the authors discuss the search engines and the users' experience with the early search technologies of that time.

Other works have focused on surveying only a subset of the technologies involved in decentralised content retrieval in blockchain-era systems. For example, a recent survey by Daniel et al. [49] discuss decentralised storage systems in the DWeb, but without focus on other technologies that enable search and retrieval of content in those storage systems. Li et al. [149] take a different focus and survey how future data-driven networks can be realised using blockchains as the underlying technology to enable decentralisation, security, privacy, and resource sharing. However, their focus is mainly on the blockchain-based solutions and do not take into account the rest of the DWeb stack. Similarly, Benisi et al. [150] describe how blockchains are used to create decentralised storage networks, where nodes can rent out their untrusted storage hardware using smart contracts.

In terms of blockchain, Zheng et al. [42] present a comprehensive overview of blockchain technologies, which includes technical components such as consensus, as well as potential applications. While certain aspects such as security and privacy enhancements, and reputation systems have been mentioned, a global DWeb use-case has not been mentioned. Neudecker and Hartenstein [151] describe the network layer aspects in terms of attacks, as well as design implementations and considerations for permissionless blockchain networks. While they focus on blockchains, other DWeb components like file systems often share network layer design and concepts, and are therefore highly related. For example, Filecoin and Ethereum 2.0 have both used GossipSub based messaging protocols in their network layers [103].

Earlier work also surveys the P2P-era content distribution research prior to the DWeb, which is also briefly used in this work. For a general overview of P2P networks, Keong et al. [152] study and compare network overlay architectures. More related to this work, Androutsellis-Theotokis and Spinellis [153] present an early survey and framework for analysing P2P content distribution technologies. Similarly, Hasan et al. [154] focus on storage techniques within distributed file systems.

Xylomenos et al. [155] present a comprehensive survey of information-centric networking (ICN), which aims to implement a content-centric network layer replacing IP. Although the content-centric paradigm (*i.e.* fetch content by name, not location) is also central to many decentralised file systems, the latter are application-level systems designed to run as overlays on top of an IP network layer.

A number of works have also surveyed popular techniques which distribute Cloud solutions, but do not necessarily decentralise their ownership and governance (in contrast with the focus of this chapter). Zolfaghari et al. [156] discuss the state-of-the-art solutions and future directions for content distribution networks (CDN). They also describe how CDNs converge with emerging paradigms like Cloud and edge computing. Ghaznavi et al. [157] focus on CDN security challenges and possible solutions to these. Mach et el. [158] describe the emerging concept of mobile edge computing, and present use-cases, integration and standardisation efforts, as well as technical solutions. Mao et al. [159] also survey mobile edge computing, but focus on the communication perspective. As mentioned before, while these solutions tackle some issues associated with centralised Cloud and Web, they remain centralised in their control and governance.

Some works also focus on hybrid solutions which combine distributed storage and computation techniques with decentralised solutions and governance such as P2P networks and blockchains. Related to content retrieval, Anjum et al. [160] survey techniques that complement centralised content delivery with P2P content retrievals in CDNs. However, such techniques use a centralised architecture, with trusted CDN servers resolving requests to appropriate peers. Jia et al. [161] also present a survey on collaboration for content delivery, focusing on collaboration techniques in network infrastructures including P2P-CDN, collaborative caching, SDN, ICN and more. Finally, Yang et al. [162] survey attempts to integrate blockchains with edge computing solutions in the areas of network, computation, and storage. If these techniques can be integrated with security and privacy first, they could be used as a building block for the DWeb, for example using computation and storage platforms to crawl and create indexes, maintain blockchains, and enhance storage networks.

3.3 Web Content Retrieval

This section describes the process of retrieving Web content on the current Web and the DWeb, introduces background on content distribution networks and content addressing, and defines a systematisation framework for studying works on content retrieval on the DWeb.

3.3.1 Retrieving Content on the Current Web

On the current Web, content retrieval is comprised of multiple interacting steps. Most of the time, a search-based workflow is used, where users submit a query to their favourite search engine with a description of the content object they are interested in. This query often takes the form of a number of descriptive keywords, which may include a content creator or publisher name, or a real-world description of content. The search engine in turn returns results to the query, which consists of Web references in the form of *Uniform Resource Locators (URLs)* (*e.g.* https://www.ucl.ac.uk/example/)

URLs referencing a content object generally use a *hostname/pathname* structure, and thus embed both the hostname of the content's provider, and the (server-specific) location of the object within the directory structure of the hosting provider's server(s). As a result, it becomes difficult to move content objects between providers as it invalidates the existing reference names to the content. Additionally, replicating objects across different servers is non-trivial as it requires duplicating server-specific directory structures across different servers. This makes both replication and movement of content a difficult task in the current Web [163], which has led to an increase in centralisation.

After querying the search engine and receiving a valid URL of a content object, a user needs to find the correct storage location of the content provider for the content. This step requires a name resolution step to be performed, which links URLs to locations. In the current Internet infrastructure, this task is performed by the Domain Name System (DNS), which can be seen as a distributed database, storing mappings from domain names (host/domain names) to IP address (location) of hosts. Once the user has resolved a name through the DNS, the desired content

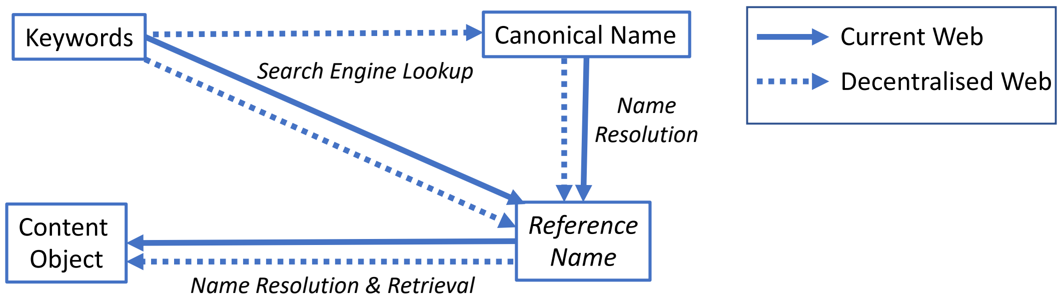

Figure 3.1: Decentralised content retrieval process on the DWeb.

object can be retrieved.

Currently, the host-centric content retrieval ecosystem on the Web relies on a *centralised trust model*, which is exposed to serious flaws and vulnerabilities. This means that each step in the content retrieval process relies on one or more forms of centralisation. For example, search engines are mostly controlled by a single owner (*e.g.* Google). The same can be applied to the DNS (which is centrally managed by ICANN), as well as content storage (*e.g.* Cloud) providers. While it can be argued that this adds trust and authority to the network, allowing absolute control to one party may also introduce several drawbacks. At present, there is a large power imbalance between these centralised entities and users, and this allows these centralised parties to influence users by adding bias and censorship, track and sell personal data, influence public opinion, and more. Users are expected by default to trust these centralised entities unconditionally, while they operate without much transparency.

3.3.2 Retrieving Content on the Decentralised Web

The DWeb operates using *a distributed trust model*, which means that content retrieval can no longer depend on trusted third parties (*e.g.* single root of trust like the DNS). Instead, users should be able to verify all steps of content retrieval, which are depicted in Figure 3.1. To start, users should be able to verify bindings between content of a retrieved data object and its reference name, in order to verify correctness. This allows users to ensure the object retrieved is the correct one to match the queried reference name, without a centralised, third party vouching for its provenance (*i.e.* the verification of the origin source of content). This verification process can be implemented by technical solutions such as self-certifying names and zero-knowledge proofs [164].

The DWeb aims to evolve the current Web beyond its current host-centric paradigm and instead use *a content-centric paradigm*. This allows reference names (also referred to as content identifiers or *CID*) to directly identify and verify content objects. This is also known as content addressing, and allows retrieval of content objects from anywhere in the network, rather than being restricted to retrieve them only from one of the content providers' locations.

The location-independence of this novel paradigm is essential because frequent replication and migration of content is the expected norm in the DWeb. This is also important because decentralised services are possibly realised by any node in the network, regardless of their location or elevated privilege levels. Decentralised services can be categorised broadly by outsourcing tasks like storage, computation, or bandwidth. Incentives and rewards are added and play an important role to ensure a fair compensation for work, and to mitigate against malicious entities.

In the DWeb, this work envisions a similar search-based workflow as is the case in the current Web. This starts with decentralised *search engines*, which serves as a translation between user queries and CIDs of content that are relevant. CIDs are typically self-certifying names to secure the binding between name and content object it refers to, and are therefore not human-readable (this is further explained in Section 3.3.4). Because of this, *canonical names* for content are important as they allow humans to refer to content. For this reason, a decentralised *name-registry* service is required to replace the DNS and perform the name resolution process from canonical names to CIDs. Finally, to retrieve the actual content, an extra resolution is needed to obtain location(s) for CIDs, which is typically performed by decentralised content storage networks (*i.e.* decentralised *file systems*).

Although the search-based workflow is popular, other workflows exist to access content on the current Web such as following hyperlinks from one page to another, as well as shared direct links to objects on Cloud-based shared drives. This work focuses on the search-based workflow, as it can be argued that this encompasses the other workflows as well. Other workflows differ as they start from later points in the same sequence of events and resolution, and therefore analysing only the search-based workflow is sufficient.

3.3.3 Content Distribution Networks

In the current Web, a large part of content retrieval has been physically distributed in order to improve performance over remote storage locations like the Cloud. In fact, content producers increasingly rely on Content Distribution Networks (CDNs) for large scale content distribution such as video streaming to a large number of geographically distributed users. However, these networks use proprietary technologies to serve content requests using a distributed infrastructure of content caches, and remain centralised in terms of control and management. It can be argued that the need for CDNs in the current Web stems from the lack of a viable decentralised content delivery technology.

A number of initiatives have aimed to add further degrees of decentralisation to CDN's. The first major step towards this was using Peer Assisted (PA)-CDN architectures [160]. Here users in a P2P network relieve some of the burden from the traditional infrastructure by allowing the users to serve content locally where possible. This hybrid approach adds scalability, as well as traffic savings. There are however a number of issues with this approach, as it inherits some of the vulnerabilities of both P2P systems, as well as centralised systems. There is still a heavy reliance on central parties, although this ensures high speeds in the case when the P2P network is not able to serve users. More importantly, PA-CDN's suffer from the inherent issues of P2P networks such as instability and a lack of incentives. However, with the advent of blockchain, peers can be incentivised to be honest and perform useful work.

Several DWeb projects [50, 165, 166] aim for replacing the CDNs with decentralised file systems, which also form large distributed caches for content. These are discussed further in Section 3.6. However, in the current infrastructure, CDNs are mainly used when performance and scalability are essential and require more stringent guarantees compared to Cloud based solutions, for example for video streaming or conferencing. The workflow used slightly differs from the one described above, as users generally access content through applications provided by the content owner, which directly integrate with CDN solutions.

In the DWeb, general file systems may not reach the required performance guarantees for these niche use-cases. For this reason, a number of initiatives have emerged which aim to specifically create fully decentralised counterparts to CDNs. While most DWeb services focus on either one of the service sharing types (storage, bandwidth, computation), these decentralised CDNs require a combination of all three in order to produce a usable system.

A few blockchain initiatives claim to create decentralised CDNs, by combining an incentive layer with an underlying storage layer. Skynet [167] for example leverages the Sia [168] storage network, and allows for pinning of content by users to create permanence. However, these initiatives are not well researched in terms of scalability or real-world performance and their design specifics remain unclear.

More targeted to the CDN use-case, Theta [169] is a blockchain which was built for the Theta Edge Network, a P2P video streaming service. The Theta network consists of validator and guardian nodes, as well as edge nodes who are responsible for the relaying of video. In order to earn rewards, edge nodes can upgrade to elite nodes by staking the associated cryptocurrency TFuel [170]. Video platforms pay for their traffic to be served on-chain and part of these fees are used as incentives. Incentivisation is twofold: nodes are rewarded for being online by uptime mining, and are offered additional rewards based on the traffic they relay, using a proof of relay. Besides video streaming, Theta has added live video streaming to the platform. However, it remains to be seen how well it is able to handle traffic, as well as whether the platform is usable. There are also many open questions regarding implementation.

Livepeer [171] is another initiative which aims to deliver decentralised live video streaming. A network of transcoders perform useful work (*i.e.* transcode video to be streamed) and in return receive rewards. Delegators in the network are tasked with curating trustworthy transcoders by staking their tokens. Although this project tackles the stage before the delivery of content, it may be a relevant building block for decentralised delivery of live video. The same uncertainties like in the case of the other projects in terms of real-world performance and implementation remain.

The rest of this chapter focuses specifically on the workflow mentioned in 3.3.2, as decentralised file systems lie somewhere between CDNs and distant storage solutions, and they are expected to handle the majority of use-cases. For niche use-cases, decentralised CDNs may play an important role, but currently the work and practical implementations of this are scarce, and it may be reasonable to assume that current CDNs will continue to play an important role for these low-latency services in a DWeb future.

3.3.4 Addressing Decentralised Web Content

As mentioned above, the distributed trust model of the DWeb requires a secure and verifiable content retrieval process. This means that the authenticity of the binding between reference names and the retrieved content object must be verifiable by users. This can be achieved by using content addressing, as more importance is given to the integrity of the file, rather than its origin. Decentralised file systems typically use verifiable (also known as self-certifying [172]) CIDs as reference names in order to achieve verifiability in the absence of trusted third parties.

Self-certifying names for content objects are typically generated using one of two mechanisms. First, the hash of the content itself can be used. CIDs are generated by applying a well-known hash function to the content. Users can simply apply the same hash function on the retrieved content object to verify the binding between the name and the object. Second, the hash of a public key controlled by the content owner can be used. Now, CIDs are generated by hashing a public key whose private counter part is used to sign the content object. Content object includes a signature, which can be used to verify the name-to-content binding of an object. The signature is typically generated by the content publisher who owns the private key, and this can be updated to achieve dynamic naming.

In practise, it has been shown that the properties of distributed trust (decentralisation), binding between names to object for self-certification (security), and human-readable names (usability) are non-trivial to achieve simultaneously, as also conjectured by *Zooko's trilemma* [173]. This states that naming systems can only have two of the following three properties: *human-readability, security*, and *decentralisation*.

Among these three properties, there are a number of contradictions. For instance, security is at odds with human-readability, because secure, self-certifying names are not human-readable due to the hash function applied. Similarly, the intrinsic binding between a human-readable name and its content or content producer is weak, and verification of this binding through a centralised trusted party such as the DNS contradicts decentralisation.

Another desirable property is persistence, which ensures that names should not change when location or ownership changes. Ideally, a minor update to a Web file should not produce a completely different name. This, however, can be at odds with the security property, because self-certifying names lead to modifications in the names of mutable (*i.e.* dynamic) content upon updates to content (*i.e.* hash of the content) or ownership (hash of the public key).

As DWeb content use hash based addressing, they satisfy only the decentralisation and security properties, which are discussed further in Section 3.6. Decentralised name-registries have the potential to *square Zooko's trilemma*, in order to achieve usability while maintaining security and decentralisation. This is done by mapping human-readable, canonical names to CIDs in a decentralised manner using a blockchain, as discussed further in Section 3.5.

3.3.5 Systematisation Framework

The process of traditional Web retrieval, as described in Section 3.3.1, can be used to define a framework which can be applied to study DWeb initiatives. This framework can be used as an exhaustive method to divide the work in three main categories, and for each category it defines key components. On top of this, a number of overlapping components are integrated, which each step of the framework leverages.

As shown in Fig. 3.2, Web retrieval can be divided into three main compo-

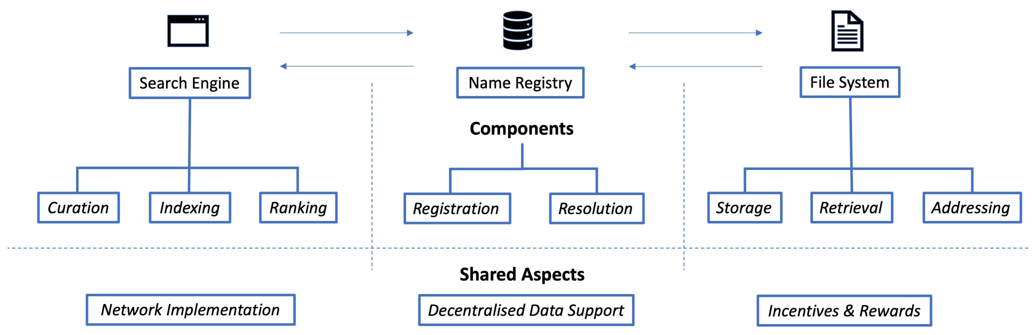

Figure 3.2: Overview of key content retrieval components on the Web.

nents: **search engine**, **name-registry**, and **file system**. For each of these areas, decentralised initiatives should be developed. This framework should allow them to position themselves amidst others in the space, and define how interoperablility can be achieved.

In order to search for content on the DWeb, users will need to use a search engine, which is able to index DWeb content. The search engine also needs to decide which content to index through curation, as well as the ranking in which order results are returned to users, defined by a transparent ranking algorithm. Indexing and convenient retrieval of content are both dependent on human-readable names (*i.e.* canonical names), which are linked to CIDs using decentralised name-registries. Users need to be able to register name-value mappings to this service, and resolve any registered name to the corresponding CID. Finally, content is stored on a decentralised file system, (or blockchain or Web servers) and needs to be retrieved from these networks using its address or CID.

This framework identifies these orthogonal components in order to clearly describe the key pillars of a DWeb infrastructure. However, in practise a lot of components may be overlapping, and they may share underlying technologies. For example, each of the mentioned components uses blockchains to promote honest participation in NRS services through incentives. Each component could even use the same blockchain network and underlying P2P network (*e.g.* Ethereum [82]), and support similar decentralised DWeb data.

In order to keep clarity and structure in this chapter, most of these components have been introduced in the preliminaries (Section 2), and are only referred to in analyses when relevant or distinct in implementation.

In the upcoming sections (3.4, 3.5, 3.6) each key component of this framework will be further defined by going over the status quo of centralised systems, after which decentralised initiatives are examined and compared. In the end of each section, a number of open issues and challenges are discussed.

	Curating	**Indexing**	**Ranking**		**Incentive**	**Advertisement**	**Decentralised**		**Network**
			Function	**Location**			**Search**	**Content**	
Presearch [174]	Crawling	-	-	Gateway Server	Y	Y	Y	N	Ethereum
Yacy [175]	Voluntary Crawling	Distributed By Document	Combined	Local	N	N	Y	N	Hybrid P2P
Brave [176]	Crawling	Centralised	-	Centralised	-	Y	N	Y	-
Nebulas [177]	Crawling	Centralised	NebulasRank	Centralised	N	N	N	Y	-
The Graph [178]	Token Signaling	Subgraph At Indexer	-	-	Y	N	Y	Y	Ethereum

Table 3.1: Overview of decentralised search engine industry projects.

3.4 Search Engine

This section first investigates how search engines currently work and identifies a number of their characteristic components. After describing these currently centralised components, a number of decentralised search engines are introduced. These are then analysed based on how they incorporate their key components. Specifically, this section discusses how DWeb search engines differ in terms of *curating*, *indexing*, *ranking*, and *incentives*.

3.4.1 Overview of Centralised Search Engines

Currently, when a user looks for content on the Web, they often start by submitting a query to a centralised search engine, consisting of one or more keywords. Proactively, the search engine has **curated** content to add to an index by crawling the Web. Keywords are then extracted from the content, and added to an inverted **index**, which maps keywords to the Web pages where they can be found.

Upon receiving queries, the inverted index is used to compile a list of pages which might be relevant to the users. These results are then **ranked** using a ranking algorithm and returned to the user. The centralised search engines control what ranking mechanism (e.g. PageRank [179]) is used and are not always transparent about the specifics. Furthermore, ranking is generally personalised, which may lead to filter bubbles [22].

To incorporate a healthy business model, most centralised search engines monetise their services by adding advertisements through keyword-auctions in search results, which allows the service to be free for users [180]. While the network infrastructure used might be distributed, the control, management, security, and policy are centralised, thus introducing a single-point-of-failure which may also lead to cascade failures [28, 181]. As these network tasks are managed centrally they do not need to add **incentives** for participation. However, in a decentralised model, services likely need to leverage alternative business models and incentives for economic feasibility.

3.4.2 Implementations

Decentralised search engines can generally be classified by their degree of decentralisation. The content which is being searched can also be classified similarly. Centralised data refers to 'traditional' Web content which is hosted at Web servers. On the other hand, decentralised data encompasses content stored using decentralised file storage, as well as blockchains. Table 3.2 provides an overview of these content types. Using this, one can distinguish between three different decentralised search types: *centralised search on decentralised data, decentralised search on centralised data*, and *decentralised search on decentralised data*.

Type	Addressing	Location	Name Registry
Blockchain Data	Block Hash	Blockchain	Blockchain Name-Registry
Decentralised Storage Data	Content Hash	Decentralised File System	Blockchain Name-Registry
Traditional Web Data	IP	Web Servers	DNS

Table 3.2: Classification of decentralised web content.

These classifications are now used to analyse early stage implementations, as well as a number of proposals in the research literature which generally have a narrow but detailed focus. Table 3.1 gives an overview of notable industry projects and summarises how they approach the various search components. Table 3.3, on the other hand, presents an overview of research proposals, focusing specifically on decentralised search mechanisms on decentralised storage networks. The reason this work divides between research and industry works is because the former generally focus on one or a few aspects of search, rather than presenting complete systems, and therefore they have been analysed using different properties. As these projects are generally narrow in focus, their main properties are discussed first, after which they will only be referred to occasionally in the rest of the analysis, as they do not present full and operational systems.

	Index Storage	**Ranking**	**Performance Optimisation**	**Security Features**	**Privacy Features**	**Governance**
SIVA [182]	IPFS DHT	-	Bloom Filter & Caching	-	-	-
Li et al. [183]	Kanban Cloud	-	Decoupled State and Computation	Verifiable Search, TEE, Decoupled Verification	Message Equalising, TEE	
Zichichi et al. [184]	Hypercube DHT	-	Routing using Hypercube	-	-	DAO
Zhu et al. [185]	B+ Tree / Hashmap	-	Index Storage Methods	Version Control	-	-
Wang and Wu [186]	IPFS DHT	Network Metrics	-	-	-	-

Table 3.3: Overview of research proposals for decentralised search mechanisms on decentralised storage networks.

3.4.2.1 P2P Search Engines

The idea of decentralised search engines was first explored by P2P search engines in order to improve the privacy, security, and performance of search on the Web and P2P storage networks. A number of initial distributed search engines relied on unstructured P2P networks [187], which offered high resilience to peer churn and good performance in retrieving popular items [188]. Some projects focused on improving the performance of unstructured search using techniques such as replication [189, 190, 191] and random walks [190, 191].

Another method of realising distributed search engines leveraged structured overlays, specifically DHTs [192, 193, 194, 195]. This allows for more reliable performance guarantees and better efficiency, especially when retrieving less popular items. A number of these focused on performance optimisations such as incorporating bloom filters [196, 197] and caching [198, 197], as well as efficient routing using ant-like behaviour [199]. Some of these used popularity scores to determine the number of indexers per file [198] or ranking of results [196].

In order to optimise performance, a hybrid of structured and unstructured networks were used. For example, Yacy [175] structures all peers in a DHT, without implementing DHT routing. Another approach [200] locates rare items using a structured overlay, while popular items are located using flooding, leading to better performance and lower overhead.

These early search engines, however, often lacked additional security measures and incentives for useful work, which are needed due to the absence of a trusted third party [201]. This ultimately led to their loss in popularity. The rest of this section focuses on recent initiatives which are able to query novel decentralised file systems (see Section 3.6) or blockchains.

3.4.2.2 Centralised Search on Decentralised Data

There are a number of centralised search engines, which are able to query decentralised data. Recent works often focus on allowing users to fetch content using CIDs [176]. However, keyword search is also possible [202], where the central entity sniffs the structured [203] or unstructured network [204] to discover new content to add to the index.

Rather than creating search engines for decentralised file systems, some works have aimed to make centralised [177] and decentralised [205] search infrastructures for blockchain and smart contract data. While the projects above rely on centralisation, they are likely to play an important role towards adoption of the DWeb.

3.4.2.3 Decentralised Search on Centralised Data

Another class of search engines are those that are decentralised but search the traditional Web. These offer much better privacy guarantees than centralised engines, but are not suitable for the DWeb, as they currently do not support indexing content on blockchains or decentralised file systems.

As mentioned above, P2P search engines lacked incentives to add robustness and security to the system. Recent decentralised search engines often leverage a blockchain to add financial rewards, thereby making the network more secure and robust. For example, Presearch [174] rewards users for participating in upkeep functions such as crawling and indexing. Instead of centralised methods of issuing and distributing rewards, smart contracts may be used for decentralised incentive governance [206]. Smart contracts can be also used for reaching consensus on indexing and ranking, as is done by Raza et al. [207] to create a framework for privacy preserving, decentralised search.

3.4.2.4 Decentralised Search on Decentralised Data

Finally, decentralised search engines are discussed which operate on decentralised data, as these are the only suitable ones for a fully decentralised Web. However, at the time of writing and to the best of my knowledge there are no implemented projects which entirely achieve this.

A number of projects [53, 208, 203, 209] focus on decentralised crawling and indexing of decentralised storage and blockchain data. Most notably, The Graph [178] is a decentralised indexing protocol for blockchain data, which itself is built on top of a blockchain.

Besides these industry projects, a number of research works have proposed decentralised keyword-search mechanism for decentralised storage networks like IPFS [50]. As these projects are generally narrow in focus, their main properties will now be discussed, only referring to them occasionally in the rest of the analysis as they do not present full and operational systems.

Li et al. [183] proposed DeSearch, which is a search engine for decentralised services which decouples state from computation by using a centralised Cloud solution to store the index with high data availability, while maintenance of the index uses decentralised workers executing verifiable tasks (*e.g.* indexing, query processing). The verifiability property ensures that any third-party (such as receivers of search results) can confirm that any search-related task involved in the search process (carried out by an untrusted worker) is performed correctly. This property is crucial in a decentralised setting where any worker can misbehave.

A number of works present systems which are fully decentralised (*i.e.* they also store the index over a P2P network). SIVA [182] builds a decentralised index for IPFS and stores it on the IPFS network using the native DHT. To increase performance, caching based on the Least Recently Used (LRU) [210] strategy and bloom filters are used. Wang and Wu [186] also propose to use the IPFS DHT to store the index and rank retrieved results from the index based on network metrics such as freshness, proximity, resource quantity, and bandwidth.

To increase performance, existing work has proposed storing the index in optimised structures rather than a general purpose DHT. For example, Zhu et al. [185] propose decentralised keyword search on decentralised data networks using B+ Tree and hashmap data structures to store the index. Zichichi et al. [184] propose a hypercube DHT to store index items, structuring network topology using keywords. Furthermore, existing work proposes delegating governance of the index to a Decentralised Autonomous Organisation (DAO) [211], which allows peers to make governance decisions in a decentralised manner, *e.g.* propose and vote for changes, as well as implement tokens.

Another interesting idea is proposed by Fujita [212], who argues for implementing similarity search on IPFS based on locality-sensitive hashing, as an alternative to the prevalent keyword-search mechanisms. In their system, content hashes are stored on a DHT, although further implementation details and feasibility analysis are an interesting avenue for future work (See Chapter 4). Furthermore, it remains unclear if this scheme is sufficient for users who expect to submit queries consisting of keywords and retrieve a range of relevant information, rather than submitting content and retrieving similar content.

As discussed in Section 3.4.7, while these research systems seem promising, they are mostly early stage works and therefore suffer from a number of limitations and require further work. A particularly interesting question is whether they actually truly achieve decentralisation. The remainder of this section examines implemented projects and highlights how some of these projects uniquely implement the components of a search engine.

3.4.3 Curating

The curation process defines which content is added to the index. A number of projects take a similar approach to centralised search engines, which rely on crawling. Yacy is an example of a decentralised crawler, which allows users to crawl locally, either manually or proactively. Optimisations for decentralised crawlers have also been proposed such as leveraging the geographic proximity of resources [213]. Most other projects [177, 176, 193] remain reliant on centralised crawlers.

In order to crawl decentralised storage networks however, different approaches are needed. To gain insights on peers and content in structured networks one may sniff the DHT traffic to discover new peers and CIDs, which can be fetched to gain insights [202]. A similar approach may be used for unstructured networks, for example, in the case of the IPFS Bitswap [214] protocol traffic (Section 3.6.4), which is used to query peers for CIDs, may be monitored [204].

Another approach besides crawling is curation based on network consensus, as is used in The Graph [178]. Nodes in the network act as curators and use tokens to signal to indexers what content is valuable. While this might be a viable approach for on-chain data, it remains to be seen if this approach would work for other content types. This can be compared to research works which use popularity scores or managers [198] to signal which items should be indexed, although the latter lack monetary incentives and are therefore more prone to performance problems.

3.4.4 Indexing

The indexing process in decentralised search engines consists of two main steps. First, metadata is collected from content in order to create index entries which map extracted keywords to content identifiers. The second step decides where the index is stored, which is generally based on partitioning *by document* or *by keyword*.

Partitioning by document means that the content objects to be indexed are divided among peers who each maintain a reverse word index for a subset of the content objects, as is often the case in unstructured networks. This approach is inefficient when locating rare items, as nodes are required to flood the network in order to locate and retrieve the query results. Storing replicas of popular items can increase the performance in these networks [198], and in general many distributed search engines offer a degree of replication, which also adds resilience against Denial-of-Service (DoS) attacks.

Most structured and hybrid engines are based on partitioning by keyword, where each node maintains an index for the words that appear across different content, generally by mapping to the closest peer in a DHT [65, 215]. For example, peer might be responsible for the entry for keyword *Apple*, and consequently stores all content items where this keyword can be found, which it returns to any node it is queried by for the keyword.

Another distinct approach is used in The Graph, where indexers simultaneously perform the tasks of producing and storing an index in the form of subgraphs of blockchain data. Users can then directly contact these indexer nodes to access the indexed data, and in return issue off-chain conditional micro payments. Other recent engines manage the index centrally [177, 174, 176].

In DeSearch [183], decentralised workers perform indexing of content in a verifiable manner through a "witness" process which runs in a Trusted Execution Environment (TEE) within each worker. The witness process provides logs of inputs and outputs of tasks carried out by workers for third parties to verify the causality between the inputs and outputs. The witness logs are also stored in a verifiable data structure, even thought this happens in a centralised public cloud. Other research works [182, 186] have proposed to store the index directly on the storage network on which they operate, as well as optimised structured overlay networks [185, 184].

3.4.5 Ranking

When a user submits a search query, the relevant entries are fetched from the index, after which the results need to be ranked based on various metrics to be ordered and returned to the user. There are various ranking algorithms, which may be applied to decentralised search engines. Most well known is the PageRank [179], which scores importance of Web pages based on the references pointing to and from the pages.

PageRank can be modified to determine the value of an entity on the blockchain, as done in NebulasRank [216]. In this work, transaction graphs are used to infer an entity's liquidity, propagation, and interoperability in order to determine its value. Nodes, smart contracts, as well as an entity's contribution to the network over a time period can be ranked, in a similar fashion to LeaderRank [217].

In centralised search engines, the ranking process generally runs globally. In a decentralised search, clients may locally select and implement their own ranking policies [218], or combine pre- and post-rankings, where results are initially ranked based on a number of standard metrics, after which they can be ranked again by the user based on local configurations [175]. While most research proposals overlook ranking of results, it has been proposed [186] to use network metrics such as freshness, proximity, resource quantity, and bandwidth.

Distributed ledgers can also be utilised to reach consensus on ranking, for example using random groups of TOR (The Onion Router) [219] block nodes and the Practical Byzantine Fault Tolerance (PBFT) algorithm [207].

3.4.6 Incentives

Centralised search engines are able to offer free services by monetising advertisements and user data. Most early distributed engines rely on an altruistic model where users are assumed to participate in the system honestly without the need for rewards. Recent systems have incorporated incentives using the blockchain. For instance, the revenue collected from advertisements could be used as rewards for up-keeping of the system [220].

The monetary inflow and outflow of the system will now be discussed separately to illustrate the decentralised network economics.

3.4.6.1 Inflow

There are generally three sources of inflow of money into the decentralised search mechanisms. The first are users paying for a service. For example, this is the case for users querying the indexed data in both The Graph [178] and DeSearch [183]. This assumes that users are willing to pay for decentralised services instead of using free centralised options, which may not hold true in practice.

The second source of inflow comes from advertisements. Generally, advertisers submit bids to show their advertisements with higher priority for particular keywords on search engines. Centralised engines generally use auctions to determine which advertisements are shown with higher priority [221, 222], although decentralised advertisement markets have been proposed as alternatives. An interesting example is keyword staking in Presearch, where the advertiser who stakes the most tokens on-chain for a particular keyword will be shown. In this case, inflow is expected to come from per-click fees. However, currently this approach retains centralisation as it relies on dedicated ad servers.

The advertisements shown to users are generally personalised, which is based on data collected from previous search behaviour. In this scenario, the user loses control over their privacy and is required to trust the central entity. To alleviate these concerns, Google introduced FLOC [223, 224], which uses federated learning [225, 226] to group users in clusters, without data leaving the user's device. Although this is argued to be decentralised and privacy first, it may lead to an advertisement monopoly, as other third party cookies will be removed. A number of research works have investigated decentralised and privacy preserving methods of personalised advertisements [227, 228], for example using blockchains [229, 230, 231].

Finally, in search protocols built on top of blockchains there is a third source of inflow. These are newly minted tokens, which are periodically released for the purpose of rewarding for network upkeep [232]. There are also transaction fees that clients pay to use the underlying blockchain network, which are proportional to the added load placed on the miners. These fees are often collected directly by miners.

3.4.6.2 Outflow

The monetary inflow into the search protocols needs to be redistributed and flow out towards involved parties. In centralised search engines, the revenue generated by advertisements is collected by the centralised operator. In contrast, decentralised systems may delegate the ad revenue back to the users who watch the ads [176], or to nodes who assist in network upkeep [174, 220].

For example, in the Graph, *indexers* earn tokens by serving client queries to their indexed subgraphs. *Delegators* can decide to stake tokens for a specific indexer, for which they receive a percentage of their profits. *Curators* are incentivised to signal subgraphs honestly, as they can earn a percentage of the query fees.

Similar to other platforms, slashing of tokens [233] may occur when malicious behaviour is detected. This leads to a penalty deduction of a node's staked deposit on-chain.

On the other hand, DeSearch [183] rewards both workers for carrying out search-related tasks (*e.g.* indexing) and publishers of content using tokens. The reward tokens flow from the *consumers* of search results all the way to the publishers of content (that appear in the search results) as in the following chain: consumers→rankers→indexer→crawlers→publishers. This chain follows the functional dependency between the tasks involved in the search process and rewards publishers of content based on their popularity, as similarly done in decentralised social media platforms [234].

3.4.7 Open Issues

3.4.7.1 Reliance on Centralised Infrastructures

As discussed, there are only a few projects which aim to provide fully decentralised search on decentralised data, and many still rely on centralised back-end or gateway servers. For example, DeSearch [149] uses a hybrid infrastructure consisting of both centralised and decentralised components, but with built-in accountability (verifiability), achieving some of the desirable properties of decentralisation with good overall performance.

On the other hand, storage of the index directly on the storage network like IPFS, while being more decentralised, introduces new challenges. Because the index should be a mutable object that is frequently updated, storing it on an immutable storage solution is difficult. The naming layer can be used to alleviate the problem of mutable data, for example using name-registries. However, there still remain a number of issues such as management of private keys. In Section 3.5 this is discussed further.

To conclude, building a truly decentralised search engine is non-trivial, and therefore a feasibility analysis is required. Specifically the question: "*are industry or research projects actually able to provide true decentralisation?*" needs to be answered. Particularly, the process of curating content to be indexed, maintaining and partitioning the distributed index, and ranking in a decentralised fashion need to be explored further. The difficulty here also applies to designing a system which encompasses all of these simultaneously. Alternative search workflows such as those based on similarity search [212, 54] seem promising in achieving higher degrees of decentralisation, but these and other workflows should be investigated further. On top of this, while privacy improvements are desirable, they should not come at a significant performance degradation, and thus this trade-off should be analysed.

3.4.7.2 Complete Systems

The area of decentralised search engines has been investigated less compared to other DWeb infrastructures, and this reflects in the fact that most systems are not complete in coverage of all search steps users expect. For example, the industry projects covered generally have a specific niche in terms of DWeb network, data type, or application. They also are not as sophisticated in implementation and design as some research works, which have a much more narrow focus.

While most research works have proposed some performance optimisations, few have looked beyond structuring and storing the index, as well as routing of queries. For example, how results are ranked after fetching them from the index has been barely explored in these works. Furthermore, how governance using incentives can be used to make the system more secure, robust, efficient, and usable has been largely overlooked.

3.4.7.3 Analysis of Claims

It is argued in most works, both in industry and research, that a decentralised search will lead to better privacy and security, but this has not been shown in practise, as novel attacks may arise in this new infrastructure. Therefore this work believes security analyses to be vital. Security is partially dependant on the cryto-economic incentives and mechanism design, which has not been considered in detail in most works, specifically in industry. Similarly, there is the issue of trust, as not all operations can be mediated through the blockchain. Here, reputation systems could play an important role.

3.5 Name Registry

This section first gives an overview of the name-registry currently used on the Web: the DNS. While the DNS is physically distributed, it is controlled and managed by a centralised entity. Next, two important aspects of name-registry systems are described, namely *registration* and *resolution*. Finally, a number of decentralised name-registries and DNS alternatives are presented and analysed in terms of how they differ in their key aspects.

3.5.1 Overview of the DNS

The DNS is the default name-registry system used in the current Web, and one of its main uses is to perform the mapping from domain names to locations. To achieve this, the DNS maintains a large number of name records, consisting of domain names such as hostnames in URLs, mapping to IP addresses of the location of the server from where a user can retrieve the content. The DNS maintains these records on a large number of distributed servers, which are used to respond to user queries.

Domain names are hierarchical in structure. At the highest level of the hierarchy are the top-level domains (TLDs) such as *.org* and *.com*. The TLDs can be extended to numerous subdomains such as *company.com*, which in turn can extend arbitrarily to sub-domains such as *mail.company.com*. The DNS namespace consists of portions called *zones*, which are each managed by a dedicated organisation or administration. DNS records for each zone are permanently stored on an *authoritative* DNS server (controlled by the zone's administration) that has the authority to respond to DNS queries for its zone(s) [235].

An authoritative DNS server for a zone is able to delegate its authority over the subdomains to other servers, resulting in the hierarchy of distributed DNS servers across the globe, each responsible for a portion of the hierarchical domain namepace. The hierarchy of servers starts from the *root name servers* that hold "pointer" records, also known as nameserver records (NS), which map each TLD zone to its corresponding authoritative DNS servers. Each authoritative server for a zone also maintains a list of authoritative servers of its delegated subdomains. Using these pointer records, users can determine the authoritative domain servers responsible for a domain name through a sequence of queries starting with at the root servers.

For the **resolution** of a hostname from a user query, a user first contacts its local DNS server. If the local server has not previously cached the result, it returns either a root name server or an authoritative name server for one of the zones that are part of the queried domain name. In the case where the server is not able to resolve the name, it returns the authoritative name server for the next subdomain using its pointer NS record.

The root zones and TLD namespace are centrally controlled by the Internet Corporation for Assigned Names and Numbers (ICANN) [236], who delegate the administrative responsibility of each zone to a single manager such as an organisation or government. In turn, these managers run authoritative servers for the zone and can allocate subdomains through sale, and delegate the control over that zone to other parties. Domain names under TLDs are **registered** with a registrar or reseller, who is accredited by ICANN and certified by the registries.

Centralisation in DNS refers to ICANN's control and management of TLD zones and the root name servers. In addition to the top-level zones, governments have full power over the DNS servers residing within their territory. This may lead to censorship, for example the blocking of *wikileaks.org* by several countries. Furthermore, there are other known, security issues with the current infrastructure such as DoS attacks [237], DNS hijacking [238], DNS spoofing [239], and DNS cache poisoning attacks [240]. Existing security extensions, such as DNSSEC [241], have slow adoption [242] due to large overheads impacting performance and also due to intrinsic reluctance to change already deployed protocols.

3.5.2 Implementations

A number of decentralised name-registry systems from industry and research are now discussed. Within the context of the DWeb, these provide registration and resolution from human-readable names to CIDs. In doing so, they have the potential to overcome Zooko's trilemma, as the content names remain secure (due to hashing), human-readable (due to the name-registry), and also decentralised (as the registry happens on a decentralised network or blockchain).

3.5.2.1 P2P DNS Alternatives

Decentralisation of the DNS was initially proposed by research in P2P systems. A number of these initiatives proposed alternative name services on top of P2P overlay networks, and target improved fault-tolerance and security. Overlook [243] targeted scalability in terms of clients, look-ups, and latency, as well as the ability to handle flash crowds. This was achieved by using dynamic replication, and by structuring only server nodes in a DHT, rather than incorporating all clients into the overlay. Similarly, the solution from Cox et al. [244] was built on top of DHash, a DHT on top of Chord and advantages of their approach include load balancing, improved security, and alleviation of system administrators. These improvements present a trade-off, as the latencies are much higher than the conventional DNS.

Abu-Amara et al. [245] used Chord to build a countermeasure to intentional DoS attacks of malicious root and TLD servers. DNS resolvers are structured in a P2P network, and in addition to conventional lookup, a round-robin approach is used to forward the query to another node, resolving the query in parallel paths. Handley and Greenhalgh [246] also proposed structuring multiple DNS servers in a P2P network in order to distribute the top level hierarchy of DNS namespaces. They argue that, while this is a brute force approach to defend against attacks, it adds lots of robustness while keeping costs manageable.

P2P initiatives eventually suffered from the limitations of P2P networks and DHTs, such as the lack of incentives. The rest of this section focuses on solutions which solve this by leveraging blockchain technology.

3.5.2.2 Hybrid Name-Registry

A number of hybrid approaches have aimed to provide name-registry improvements over the current DNS by leveraging a combination of centralised and decentralised infrastructures. DNSLink[1] allows IPFS CIDs to be mapped using DNS txt records to DNS names. This does not overcome Zooko's trilemma, as it remains reliant on the centralised DNS.

[1]`https://dnslink.io`

Some works use consortium blockchains to create a decentralised DNS. This work considers these to be hybrids as well, as these networks are not entirely open and decentralised. They generally publish domain name operations on-chain, but store actual domain name data off-chain.

ConsortiumDNS [247] uses a single consortium chain with miner and query nodes. Additionally, a gossip protocol is used for node synchronisation and an index is generated of blocks and transactions. In contrast, a hierarchical structure of multiple chains may also be used [248, 249]. The rest of this section focuses on solutions implementing open blockchain and smart contract based name-registry systems.

3.5.2.3 Blockchain-based Name-Registry

A number of industry and research projects have proposed using blockchains for name-registry, mapping human-readable names to CIDs in a decentralised manner and claim that they overcome the Zooko's trilemma. First, projects are described which use first-order registration, *i.e.* those that modify the blockchain state directly using transactions, rather than smart contracts.

A generic name-value registration system is implemented by Namecoin [250], offering a naming system with decentralised governance. Similarly, NXT [251] and Emercoin [252] implement generic name-value storage service on their native blockchains. Another blockchain-based naming protocol is Handshake [253], which aims to replace the root zone file and root servers. Rather than targeting to replace the entire DNS infrastructure, the control of the TLDs is decentralised, allowing an infinite number of names to be created. Therefore, compared to other solutions which allow naming operation within the scope of one or a few TLDs (e.g. *.bit* for Namecoin), Handshake is more flexible and customisable. On top of these naming protocols, other systems can be built to create secondary marketplaces for reselling names and easy participation in name auctions [254], as well as to add security and accessibility[2]. In sections 3.5.3 and 3.5.4 key aspects of these mechanisms are discussed in more detail.

[2]`https://github.com/okTurtles/dnschain`

Besides these industry initiatives, a number of research works [255, 256, 257, 258] have focused on the security vulnerabilities of the DNS and propose using blockchain solutions to enhance the security of the current infrastructure. Security issues are partially due to the absence of a method to certify the integrity of information of queried name records. A number of works have improved this by storing verifiable record hashes on the blockchain [259, 260].

Blockchain-based registry systems may also be extended to Public-Key Infrastructure (PKI) encryption schemes, which generally suffer from similar issues due to reliance on centralised certificate authorities [261, 262].

3.5.2.4 Smart Contract Name-Registry

Decentralised name-registry systems can also be implemented using smart contracts on top of existing blockchains. The advantage of using smart contracts is that many services can be offered on the same blockchain. Blockchains that solely implement naming operations can be less secure as the network is often smaller, and may have limited functionality. On the other hand, as there is less overall traffic, better performance can be expected. The advantages of both are expected to converge with sharding [106] and layer-2 solutions [107].

A number of projects use the Ethereum blockchain as underlying infrastructure [263, 264], and generally use a set of smart contracts for registration and resolution. Most developed among these is the Ethereum Name Service (ENS) [265], which is a general name-registry for the DWeb content including cryptocurrency addresses. However, around 98% of currently registered names on ENS seem to identify Ethereum addresses [266]. Stacks [267, 268] also created the Blockchain Naming System (BNS) on top of their native blockchain using a smart contract, after initially using the Namecoin blockchain [269].

The industry projects discussed above still have many security vulnerabilities [270], particularly in the areas of malware, name-registration mechanisms and markets, phishing, and immutability. Specifically, looking at name-registration, *domain squatting* [271] attacks present a big threat. In this attack, malicious users register as many names as possible at low costs, with the sole purpose of selling them in the future for profit rather than using them, or using them for fraudulent activity based on misdirection or impersonating another source. To illustrate some of these issues, studies have identified that in Namecoin, squatting is a significant problem [272], a single entity controlled over 51% of the network [269], and that there are possible domain extortion and phishing schemes [270].

Another aspect often overlooked in the design of decentralised name-registries are **incentives**. Similar to the other components in a DWeb infrastructure, nodes will need to collaboratively perform work to keep the system working, for which they expect rewards. In the case of blockchain and smart-contract based solutions, some of the incentivisation for networking tasks are taken care of by the underlying blockchain and consensus protocol. However, to mitigate some of the attacks mentioned, malicious behaviour should be protected against by aligning incentives with honest behaviour, specifically tailored for the name-registry use-case. This has been partially achieved by the registration mechanism, as described in Section 3.5.3.

In the research literature, Liu et al. [249] target availability and consistency problems for the current DNS, and propose a name resolution and management system called FI-DNS, which consists of a two tier architecture where the first tier uses a permissioned network of root peers, governed by smart contracts. The second tier uses a permissionless distributed file network, where authoritative peers store domain name data. Evaluation of the system shows a resolution delays up to double those of DNS, which is argued to be due to slow execution efficiency of the blockchain. This performance bottleneck however makes this system unfit for a DWeb which has a similar QoS/QoE as the current infrastructure.

	Scope	**Ownership**	**Off-Chain Storage**	**Registry Fee**	**Resolution**	**Allow Subdomains**	**Network**
Namecoin [272]	TLD	Permanent	N	Flat Fee	Local	N	Bitcoin
BNS [268]	Root zone	TLD Dependent	Y	TLD Dependent	Local	Y	Bitcoin
Handshake [253]	Root zone	Permanent	N	Auction	Local	Y	Handshake
ENS [265]	TLDs	Lease	N	Length Based	Local	Y	Ethereum
NXT [251]	TLD	Permanent	N	Flat Fee	Local / Server	N	NXT
Emercoin [252]	TLDs	Lease	N	Length Based	Local / Server	N	Emercoin
CNS [263]	TLD	Permanent	N	Premium / Regular	Local	Y	Ethereum

Table 3.4: Overview of decentralised name-registry projects.

The remainder of this section highlights unique aspects of blockchain and smart contract name-registries, specifically in the areas of registration and resolution. Table 3.4 gives an overview of key aspects for select projects described in the previous sections.

3.5.3 Registration

Ownership, pricing, and control of names are handled differently among projects. Ownership of a namespace can be permanent [253, 250, 251, 263], in which case the owner has control over the subdomains indefinitely, although there may be periodic renewals required to ensure liveness at no cost. For example, Namecoin allows peers to store and update name-value mappings on the blockchain, by paying a fee in Namecoin's own crypto-currency (*i.e.* NMC) to miners. All names are bought using a flat fee transferred to a 'black hole' address. Name registrations must be renewed after expiry. Similarly, aliases in NXT can be created, edited, and transferred by users for a flat fee. However, studies on the initial design of Namecoin indicate that the flat valuation of names and very low fees coupled with the lack of a proper secondary market, where interested buyers can search for names, makes the system susceptible to domain squatting.

Conversely, ownership may also be temporary and may require periodic renewal fees to extend the lease period [265, 252], which may deter squatting attacks. Ownership permanence may also be set differently among namespaces within the same system. In BNS [268], different namespaces adopt policies for name registration. In most popular domain *(.id)* registration fees are transferred to a 'black hole', which aims to deter squatting behaviour. Registration is more expensive for shorter names, and needs to be renewed every two years. Once domain names are owned, they can be used to create and sell subdomains off-chain. These are cheaper as registration happens off-chain, but has similar guarantees as it is anchored to the blockchain by sending batch transactions from the domain name owner.

Pricing of domains and namespaces also varies across systems (and even within the same system [268]). Initially, low flat fees were the norm for acquiring domains [250, 251]. However, as mentioned it was shown this pricing model made the system susceptible to squatting. To counter this, a number of projects started charging differently based on the perceived value of a name [263], for example based on their length [265, 252]. Another method is leveraged in Handshake [253], which uses Vickrey sealed-bid auctions [273] on-chain using covenants [274] to sell namespaces. A registration covenant start the auction process for a domain, after which bids are expressed using HNS coins. A renewal covenant may be used to regularly renew the registration period. There are no yearly renewal fees, and ownership is private as a domain is linked to a user's private key.

All systems allow for reselling of domain names on a secondary market, as this is seen to be a security feature against squatting. Some extend this further by allowing sale of subdomains of a name [253, 265, 268, 263].

3.5.4 Resolution

The hybrid projects mentioned either rely on servers[3], the current infrastructure, or a permissioned chain to resolve names. On the other hand, for blockchain and smart contract based solutions, the main difference in resolution with the DNS is that they directly use the blockchain to resolve names. This can be done locally by running a full node on the network and querying the blockchain records pointing to the content for each resolution, using a simplified payment verification (SPV) node [253], relying on browser extensions, or using servers [250, 251, 252].

When querying the blockchain, the entire naming records could be traversed to find a relevant entry. A faster method uses separate resolver (which maintains an "authoritative" record set by the owner) and registry (where the search starts) smart contracts [265, 263]. ENS uses resolver contracts to provide the resolution service. A name owner can program the resolver of a name as they desire, which is used when it is queried by users during lookups, similar to the role of a zone's authoritative server in DNS. The registry contract is used to map names to corresponding resolver contracts, and is used to start a search by a user.

[3]`https://www.opennic.org`

3.5.5 Open Issues

3.5.5.1 Security

Decentralised name-registries and DNS alternatives are recent developments, especially those built on top of blockchains. While the initial implementations and results seem promising, more research is needed into how they hold up in practise, especially in terms of security. Recent works [270, 272, 266] have exposed some serious security threats and design flaws in early systems. They focus on specific vulnerabilities such as domain squatting and phishing, but a wider attack vector needs to be analysed and evaluated, before one can claim that they offer better or similar security guarantees as the DNS, and that they are actually able to "square" Zooko's trilemma. Furthermore, these systems rely on trust and performance assumptions of the underlying blockchain network, which has been shown to be too slow [249] in certain instances. Some projects rely on centralised servers for name resolution to increase performance, but this adds a layer of centralisation [250, 251, 252].

3.5.5.2 Namespace Management

Another aspect which has often been overlooked is how these systems handle instances where public keys to alter names are lost, compromised, or even just upgraded. It may also be desirable to use a threshold of public keys, instead of just one, to verify the identity of owners or publishers for security reasons. Although P2P literature has attempted to tackle these issues, for example using social and personal naming systems [275], the blockchain based systems have not identified or addressed these issues.

The prevalence of various financially-motivated attacks (such as domain squatting) is a sign that there is room for improvement in the decentralised management and governance of namespaces. For example, popular names, especially those with commercial values (registered trademarks), require careful management, as they are obvious targets for such attacks [266]. While decentralised name registries that are governed by smart contracts have developed mechanisms such as auctions to manage namespace ownership, more research is needed for building algorithmic mechanisms for robust namespace governance (possibly together with crypto-economic incentive mechanisms) to deter financially-motivated attacks on the namespace.

3.5.5.3 Deployment and Support

In terms of ease and practicality of deployment for decentralised name registries, recently several browsers have introduced extensions (plug-ins) for ENS support. However, despite the browser support, a recent study [266] has reported only few thousands of URLs being stored in ENS, while the vast majority (98%) of the names identify blockchain addresses. This suggests that the deployment, incorporation, and support for these naming systems are still very young and need time to reach adoption, which can take a long time in Internet infrastructures [276]. However, the number of names registered on the ENS system (including the number of URLs) has been reported to be steadily rising.

3.6 Decentralised File System

This section first describes how content is currently stored on the Web, and discusses how *storage*, *retrieval*, *addressing*, and *incentivisation* are handled. Next, a number of decentralised file system implementations are described and analysed based on how they approach these key aspects.

3.6.1 Overview of Web Storage

In terms of content **storage**, the current Web ecosystem is dominated by centrally-controlled public Cloud infrastructures. While these infrastructures provide users with on-demand access to a large pool of shared resources, they operate with little or no transparency. As a result, concerns over the security of confidential or sensitive data can favour the deployment of private Cloud infrastructures which require large upfront costs.

More importantly, the centralisation in the infrastructures means that they reside in a few locations in the Internet. Consequently, simple network failures can lead to unavailability of these infrastructures, as experienced by users during recent outages at Amazon Web Services (which resulted in loss of access to a significant portion of the Web) and Facebook [181, 28]. While replication of content across provider boundaries would lead to better performance and availability for users, the lack of incentives prevent such cooperative action between providers.

Content **retrieval** from centralised Cloud infrastructures deployed at remote datacenters can experience large communication latency. To reduce this latency, the emerging *edge computing* [29] paradigm promises to deploy small-scale datacenters at locations close to users. However, such small-scale edge infrastructures are mostly appropriate for small-scale, low-latency (or high-bandwidth) applications and can not cope with the workload of the entire Web. Instead, a truly decentralised Web can be realised by pooling the vast amount of global user resources and **incentivising** their proper collaborative usage in order to achieve scalability and sufficient performance.

As discussed in Section 3.3.3, other important actors in content retrieval in the current Web are Content Distribution Networks (CDNs), which provide large-scale retrieval of content requiring a high QoS, through on-demand replication of content at distributed caches around the world. While on-demand replication and caching of content with simple reactive caching policies (such as LRU) have been shown to be effective in providing sufficient content retrieval performance, it remains difficult to replicate or move content across as such actions invalidate existing references to the content, due to the location-based **addressing** of content. CDNs have tackled this issue by using proprietary name resolution mechanisms that immediately update the invalid Web references to content upon movement or replication.

Despite being a distributed infrastructure, CDNs are centrally-governed systems and charge content producers for distributing their content. This makes content delivery expensive, especially for small content producers. Finally, in order to serve content using HTTPS, CDNs need to hold content publisher's private keys further increasing centralisation and lowering security of the entire Web [277].

3.6.2 Implementations

The ideas behind distributed storage networks were first developed for P2P networks, and produced unstructured networks like Gnutella [60]. While these were able to perform well in fetching popular items, they were not as successful in

Architecture	Hash	Decentralised	Self-Certifying	Human Readable	Hierarchical
IPFS [50]	Multihash	Y	Y	N	N
Swarm [165]	bzzhash	Y	Y	N	N
BitTorrent [278]	-	N	N	Y	N
Skynet [168]	Skylink Hash*	Y	Y	N	N
Storj [166]	-	N	N	Y	Y

Table 3.5: Comparison of addressing of decentralised Web content. * *unclear which hashing algorithm is used.*

quickly retrieving less popular content. A number of projects started leveraging structured networks and particularly DHTs to achieve more reliable performance guarantees. Most prominently among these was BitTorrent [278]. Over time it became clear that many of these networks lacked robustness in terms of availability, security, and stability, partially due to the lack of incentives. Furthermore, BitTorrent's main use became the distribution of unlicensed products [279], leading to copyright and legal issues.

Recently, novel decentralised storage networks have emerged and gained popularity [49], most notably IPFS [50], Sia [168], and Swarm [165]. These can be built on structured, unstructured, or hybrid networks and use content addressing. While the principles of these projects are closely related to Information Centric Networking (ICN) [51, 155], the implementation of a content-centric paradigm directly in the network layer that replaces IP, these novel projects differ as they work in the application layer.

Content addressing is a natural fit for decentralised file systems targeting a public DWeb, as content is distributed over the network with a level of replication (as every node which downloads content automatically caches it), and therefore any node (or a set of locations) may be able to serve a requested file. It would be counter-intuitive to restrict file retrieval to only a single location as is done in the current Web (or a few locations if the file is cached at multiple nodes). For storage of private data however, similar to personal Cloud storage, content addressing is not always necessary. Such is the case with Storj [166], which also introduces optimisations targeted towards decentralised Cloud storage and uses satellite nodes which manage parts of the network.

DStore [280] takes another approach to create a distributed outsourced data storage and retrieval scheme. It uses smart contracts to audit the integrity of the outsourced data, achieving security and efficiency. Liang et al. [281] designed a storage and repair scheme for fault-tolerant data coding, realising a regeneration code with high precision and repairability, focusing on blockchain-based networks.

Another distinct project that proposes decentralising storage is Social Linked Data (SoLiD) [282], which is designed to decouple user's personal data from the applications that use them and allows users to set access control policies to maintain privacy of their data stored in decentralised storage units. However, users must trust the decentralised storage units with properly authenticating applications and following their access control policies. More importantly, the current SoLiD protocols rely on centralised infrastructures such as the PKIs and DNS.

Finally, blockchains should be mentioned as an alternative method of storing data in a decentralised manner. While storing on the blockchain is secure, it is extremely expensive, as the data is replicated over all peers and thus distributed with extreme redundancy. In the rest of this section, the focus is on recent decentralised file systems on the application layer with live implementations, and their key aspects are analysed.

3.6.3 Storage

Recent decentralised file systems are generally implemented over P2P storage networks, where a DHT structure may be used to find peers that are the providers of a specific content. Content is typically stored initially only by the publisher who serves the file, given that the publisher can (and is willing to) actively function as a provider of their content. Additionally, any peer downloading a content can cache that content and become a provider [50]. Furthermore, some protocols allow for nodes to formally publish deals governed by a blockchain, where one node pledges to store a particular content item [44, 168]. Secondary off-chain markets have also emerged where providers offer to *pin* specific files (*i.e.* permanently make the file available). Some systems also introduce coding techniques such as erasure coding to improve the retrievability of content, meaning that only a certain percentage of coded segments of a content item is sufficient to restore the content. Combined with incentivised pinning of files at multiple locations, coding can further improve the permanence of content stored in these systems.

The content stored on the decentralised file systems is generally public data, and anyone in the network with the CID can fetch the content. This approach causes privacy concerns for users, which has been overlooked in some systems. For example, the content searched by a peer can be easily monitored, especially by others that are directly connected to the peer in the P2P network [204]. Some systems such as OneSwarm [283] distinguish between trusted (*e.g.* friends and family) and untrusted peers and introduce address obscuring techniques to increase privacy protection of participants.

Protecting the privacy of storage nodes is important to avoid censorship of content [284], and there may be several legal implications. For example, serving nodes may need to disclose any prohibited content hosted by them, and therefore the ability of plausible deniability becomes important in these shared storage solutions. Servers can deny knowledge of content they store, if the clients store content in encrypted form and separately from the keys used to encrypt the content [285]. MaidSafe [286] and Storj [166] both store data in an encrypted form, and content is divided into a sequence of chunks which are stored individually on the DHT. In MaidSafe [286], each chunk is encrypted with the hash of the previous chunk in the sequence, and each encrypted chunk is then XORed with the concatenated hashes of the original chunks for further obfuscation. Furthermore, publishers have to publish a manifest file containing meta-data that maps the hash of obfuscated chunks to the hash of the real chunks.

Some DHT implementations allow nodes to search for peers who store a given CID. In these implementations, a client searching for a content object by its CID can retrieve *provider records* by querying the DHT, consisting of the IP addresses and peer identifier of peers that store that content. As this *storage privacy* is a challenging problem, a possible privacy extension is to store encrypted provider records, and allow content publishers control over access to the key, as was done in SoLiD [282]. Another recent idea is to encrypt provider records of a CID using a key derived from the CID itself [287], which allows providers to be revealed to only those who know the CID of that content, where the CID is now constructed as a combination of the hash of the content and secret information known only to the publisher.

3.6.4 Retrieval

Retrieving content from decentralised file systems can happen through the network they are implemented on, being either unstructured, structured, or hybrid. In the unstructured case of Sia [168], nodes gather hints of the possible location through for example the blockchain deals, after which a select number of nodes are queried, rather than using a flooding-based approach. The other projects use modified versions of the Kademlia [65] DHT either just for locating peers [278, 44], or both peer and content discovery [50, 165, 166].

The hybrid approach in IPFS aims to optimise the performance of content retrieval through both unstructured connections with a set of peers and the structured DHT network based on Kademlia. As part of the unstructured network, each node maintains connections with a small set of peers that are discovered either through DHT communications or incoming content requests. These connections are used as part of the Bitswap [214] protocol to request for content. In the Bitswap protocol, nodes exchange lists of wanted content (using their CIDs) with their directly connected peers. Upon receiving a Bitswap *want* request, one or more peers may respond with an acknowledgement of having the content cached locally. Upon receiving one or more acknowledgements, the node then attempts to retrieve the content from all of the acknowledging peers in parallel, similar to downloading content using BitTorrent [278].

A node who wants to download a content object first asks its Bitswap peers for that content's CID. If none of the direct peers have the requested content locally cached, then the node queries the DHT for a list of peers who can provide the content. In general, clients may be able to retrieve content using their direct connections (especially, the popular content that are previously retrieved and cached by many peers) without using the global DHT. Because retrieval of content through a global DHT can be slow (requiring contacting O(log n)), Bitswap can reduce the content retrieval latency. The Bitswap protocol also helps with reducing the burden on the DHT network. However, attempting to retrieve unpopular content from BitSwap peers may end up delaying the retrieval, as it delays switching over to the DHT to query for content. Therefore, a hybrid system may require optimisations to improve the content retrieval latency by perhaps using both networks at the same time at the cost of additional overhead in the system.

In addition to the performance of content retrieval, privacy is another important consideration. Ideally, a system should not reveal which particular content is searched by a given client, providing *reader privacy*. A recent lightweight extensions to the IPFS DHT enables clients to search for content without revealing the exact content that they are looking for [287]. This way, the discovery of providers of a CID and querying the content providers for content become decoupled. Now, a client can search for providers of a CID by querying the DHT using a prefix of the CID as the search key. This still allows the DHT to find a region with potential providers, while hiding the original CID from the DHT nodes that route the query. When potential providers of the CID are found, the client queries these peers using the hash of the CID, instead of the CID itself. This way, the providers are unable to find out which CID a reader is looking for, unless they have the content cached.

3.6.5 Addressing

As discussed in Section 3.3.4, addressing content on the DWeb is not straightforward, because many of the desirable properties cannot be achieved simultaneously, as described by Zooko. Most recent projects targeting public data, such as those for the Web, use content-addressed, self-certifying hashes to refer to content [165, 168]. This can be extended to support multiple hash functions by using prefixes, as is done by multihash [50]. Human-readability can be achieved using trackers [278], at a loss of security or decentralisation.

A desirable property of naming is that even mutable content objects have persistent names that users can always use to refer to them. This means that CIDs of content objects should not change when their attributes (*e.g.* location, file contents, or ownership) change. Hash-based names do not provide persistence, because the contents of a file determines its name. This could however be achieved with public-key based names (refer to the types of self-certifying names in Section 3.3.4) such as IPNS[4], which allow identifiers to be linked to public keys. This way, a user can update a file by signing the updated file with their private key, while keeping the name of the file the same.

3.6.6 Incentives

Currently, centralised storage options are cheap or even free, because the centralised parties are able to monetise their services. Decentralised services mitigate against security vulnerabilities and add transparency, but still require workers to be able to cover their cost of work, which is why incentives play an important role.

Early P2P storage networks generally leveraged non-financial incentives, such as BitTorrent's tit-for-tat [288], which rewards for resources put towards the network by faster downloads in return. Another example is Samsara [289] which focuses on tit-for-tat behaviour for contributing storage resources, in a symmetric storage relationships between peers. In Samsara, a peer stores a chunk of data for another peer, in exchange for the receiving an equally-sized storage promise. The existence can be verified using a challenge-response protocol which prevents nodes from removing or compressing. However, malicious peers can refuse to store data later when requested as the promise mechanism can not enforce peers replacing the promise with data. Also, the verification adds significant overheads on the peers.

A number of projects have also started incorporating blockchain based rewards in their networks. Filecoin [44] creates an incentive layer on IPFS where nodes create on-chain storage deals. Storage nodes regularly submit proofs that they have been storing unique copies of the data, for which they receive off-chain micropayments. Similarly, BitTorrent issued a token to add robustness in their platform, while Skynet, a decentralised CDN, leverages the Sia blockchain. Swarm and Storj issued blockchain tokens as well. Arweave [290] takes another approach towards re-

[4]`https://docs.ipfs.io/concepts/ipns/`

alising decentralised storage and uses a blockchain-like linked structure with mining rewards based on pseudo-random previous blocks linked to the latest state. Therefore, users pay a one-time mining fee for storage, assuming that miners are honest in keeping and providing their data, which may not hold in practise and lead to poor scalability and performance.

3.6.7 Open Issues

3.6.7.1 Achieving Desired Properties

One of the main issues in decentralised file systems remains the contradiction of desired properties of names (Zooko's trilemma). For example, secure, self-certified naming is at odds with human-readability. While solutions, such as name-registries have been proposed to handle these contradictions, they need further analysis and evaluation. Also, storage of mutable content is another challenge in decentralised file systems. Even when the hash of a public key is used for persistent naming of dynamic content, the file system must guarantee that a retrieval operation on a name would return up-to-date content and not an outdated file that is cached by the nodes in the network. One possible workaround is to add version control to names, but this also comes with problems such as retrievers not necessarily knowing the current version of content.

3.6.7.2 Privacy and Performance

Privacy of both the content retrievers (reader) and content providers (storage) in decentralised file systems is an active area of research, as discussed above. Recursive routing can improve the reader and writer privacy, and also reduce the latency of content retrieval, as fewer round-trip times are required to locate storage nodes [286], and content requests and the corresponding data are routed through intermediate nodes.

In terms of reader privacy, there are few other promising approaches that aim to hide the content that clients are searching for from other participants in the network. These approaches are generally lightweight, but still prone to content query leakage. When nodes use a prefix of the CID as a search key to find providers for example, while the original CID is hidden, nodes can still infer content queried by matching the key to popular or cached CIDs [287]. Another approach is to use threshold cryptography along with quorums of peers to enable routing queries with privacy, but this adds significant overhead [291].

In terms of performance, decentralised file systems can suffer from slow response times as reported recently by measurement studies [146], and it is an active area of research to improve the performance of content retrieval in these systems. The use of hybrid P2P networks is an effective approach, especially for retrieving popular content with low latency. However, striking a balance between performance and privacy is a challenging problem that requires more attention.

3.6.7.3 Legality and Moderation

Finally the unclear legal implications need to be highlighted across all components described in this chapter, which will require cross disciplinary work. For example, the legal implications of adding illegal content to the index and being returned by a search engine, registering a domain name for a company by someone else, or storing illegal files on the decentralised file system are not clear.

3.7 Summary

This chapter has presented a framework and thorough overview and analysis of the content retrieval process on the decentralised Web, also known as DWeb. After describing how content retrieval is handled on the current Web, essential components of the retrieval process are identified, consisting of search engines, name-registries, and file systems. In each of these areas, an overview of the state-of-the-art projects and proposals is provided, as well as a comparative analysis with the current centralised model.

The analysis has highlighted a number of open issues, which need to be addressed for a decentralised Web to be realised. In the area of search engines, most existing projects are not able to truly meet the demands of DWeb, and often violate the decentralisation property at one or more levels. Furthermore, there is more work needed to verify the claims made in terms of security, privacy, and performance guarantees. While name-registries have more mature implementations on blockchains, they also require feasibility analyses and recent works have pointed out a number of security vulnerabilities which need to be addressed. Out of the three key components, the file systems are the most mature, and have working implementations and applications. Similar challenges and open questions remain in this field regarding decentralisation and key ownership, as well as legal and privacy concerns. A number of contradictions also arise from the self-certifying property of names, which ensures security but comes at a loss of usability.

Chapter 4

Decentralised Search Mechanisms for the DWeb

4.1 Overview

This chapter explores decentralised search mechanisms for DWeb content. First, an experimental indexer and search engine is introduced for IPFS. From this implementation, a number of conclusions are drawn to determine that a truly decentralised search is non-trivial to implement, and the proposed infrastructure is not suitable.

In order to overcome the issues of current search systems, Ditto is introduced, which is a DWeb search mechanism based on similarity search. Ditto uses locality sensitive hashing (LSH) to extract similarity signatures and records from content, which are stored on a decentralised index on top of a distributed hash table (DHT). Furthermore, numerous underlying content networks and types are supported, as well as a wide range of use-cases including keyword-search. The evaluation confirms the feasibility of Ditto, and shows comparable search quality, delay, and overhead which are currently accepted by users of search and DWeb systems.

4.1.1 Introduction

In the current Web model, search engines such as Google or Yahoo are the main entry point for users accessing the Internet. Web search can broadly be broken down into two main categories: *keyword-search* and *similarity search*. In keyword-search, users submit their keywords to a search engine, which returns content it has previously crawled and indexed from centrally-managed Web servers. On the other hand, similarity search (*e.g.* Google reverse search) requires users to submit a base item (*e.g.* an image), for which the engine returns previously crawled items that are close to the base in terms of a particular distance metric. Similarity search is also used for multiple kinds of recommendation systems, where users are shown items (*e.g.* media content such as songs) that they might like based on their interaction history.

The decentralised Web has made lots of progress in recent times, as discussed in Chapter 3. However, despite the significant progress of decentralisation in recent years, one of the main problems of decentralised search engines remains unsolved. Previous attempts do not support both keyword and similarity search [182, 186, 185, 178, 184] (*search flexibility*), do not provide result integrity [182, 186, 185, 184] (*security*), are not completely decentralised [183, 202] (*decentralisation*), are bound to a specific underlying technology [182, 186, 185, 178, 184], or require a global view of the network and Cloud infrastructures to ensure performance guarantees (*high-performance*), making them difficult to deploy in practice.

This chapter explores decentralised search mechanisms. First, the design of a keyword-search mechanism for IPFS is provided, which is used to conclude that a truly decentralised search mechanism is non-trivial to implement following the traditional model of maintaining a crawled index. This then inspired the design of Ditto, a decentralised search mechanism for DWeb services based on similarity search, which allows for various search use-cases on a variety of DWeb content, independent of the underlying DSN or blockchain. The name Ditto refers to the property of returning items which are similar to each other.

Ditto achieves its functionality by using a similarity search mechanism based on locality-sensitive hashing (LSH), which extracts short content signatures that maintain similarity features. Consequently, Ditto is able to group together content with high similarity and on top of this build various search functionality beyond recommendation (which is discussed further in Section 4.4.3). Furthermore, by extracting and hashing keywords directly from content in the algorithm, a truly *decentralised keyword-search* can be implemented.

Ditto does not require centralised components or a global view of the stored content, and security is guaranteed by the verifiability of LSH computations. When adding DWeb content, providers are incentivised to compute similarity signatures. Mappings from signature to content identifiers (CID) are stored using a modified distributed hash table (DHT), allowing for quick and reliable lookup. A user or application can leverage search functionality by supplying a signature, content type, and similarity range. Section 4.5, describes the system in more detail.

Through an initial evaluation, the feasibility of Ditto is verified. Delays in terms of signature generation and lookup of LSH are found to be acceptable for a decentralised setting (sub-second as expected by current Web users). Furthermore, the overhead of participation does not greatly increase beyond current systems, and the quality of similarity search and keyword-search is acceptable and comparable to the baseline (recall up to 57%).

To summarise, this chapter proposes Ditto, which uniquely achieves the following:

- A general decentralised search mechanism with interoperability for various DWeb content sources and types.
- *Similarity search* for a broad number of use-cases such as recommendation, malware detection, and moderation.
- A decentralised way of achieving *keyword-search*, allowing for semantic search on the DWeb without any single root-of-trust.

4.1.2 Related Work

While a more complete discussion of decentralised search mechanisms is provided in Section 3.4, this section briefly extends this discussion of decentralised search for DWeb content, and additionally includes a discussion on similarity search works. These focus on works relevant for the specfic discussion in this chapter.

4.1.2.1 Decentralised Search on the DWeb

A number of works have proposed decentralised keyword-search mechanisms, similar to current search engine workflows based on crawling and indexing. Li et al. [183] proposed DeSearch, which decouples state from computation by using a centralised Cloud solution to store the index, while maintenance of the index uses decentralised workers executing verifiable tasks.

A number of works present improved decentralisation by storing the index on a P2P network. For example, SIVA [182] and Wang and Wu [186] propose to store a decentralised index for the IPFS DSN directly on the IPFS DHT, which is similar to the proposed experimental IPFS search mechanism in this chapter. Other initiatives [185, 184] have attempted to translate the centralised search engine workflow to a decentralised setting.

However, these search engines fail to capture the needs of a truly open and decentralised network, as they are not entirely decentralised in their index storage, or require a single root-of-trust for naming consistency and provenance. Furthermore, these approaches require lots of network participation for tasks like indexing and crawling, requiring additional incentives, which have not been implemented and it is unclear who will pay for it (monetary inflow source).

In contrast, Ditto hardly requires additional work for network peers who already actively store content. This is due to the fact that the crawling/indexing model isn't copied directly, but instead a similarity search framework is used. Further, for content producers, it is an extra incentive to participate in the system for their content to be found.

4.1.2.2 Decentralised Similarity Search

There has been lots of work on similarity search architectures in terms of implementations and optimisation techniques, and specifically in locality-sensitive hashing. A number of works [292, 293] have focused on implementing generic distributed versions of LSH, proposing a number of performance improvements. However, they either rely on centralised components, or they do not specify the networking implementation and solely focus on the algorithm details.

Some works have proposed to use a DHT to store the mapping of similarity signatures to content [294, 295]. Haghani et al. [296] leverage a cyclic DHT space based on Chord [66] to provide nearest neighbour search and queries within a range. While peers are organised in local DHT's, their system does leverage gateway peers.

Other works include Bahmani et al. [297], who improve network cost of LSH by proposing a layered LSH method on two distributed frameworks. Hamming DHT [298] implements a DHT where identifiers are generated based on LSH and the hamming distance metric is used in maintaining a Chord based ring structure. The works mentioned generally focus on increasing performance of LSH frameworks in a decentralised setting, but are not tailored towards the DWeb like Ditto, meaning that they lack details on specific security and privacy considerations.

Within the DWeb setting, Fujita [212] argues for implementing similarity search on IPFS using a DHT, but lacks details in areas like signature generation and network implementation. Yuan et al. [299] implement a LSH-based image retrieval scheme using blockchain smart contracts and distributed storage on IPFS.

Ditto differentiates itself in a number of ways from prior works. Interoperability and support for different content networks and types is guaranteed, which is required in a DWeb environment. Furthermore, the wide range of DWeb use-cases described, and specifically keyword-search using LSH has not been explored prior.

4.2 Experimental Indexer and Search for IPFS

4.2.1 Overview

In order to explore the feasibility and design of decentralised search engines for the DWeb, I developed and implemented an experimental search mechanism for IPFS, as a part of a call for projects from the DI2F workshop at IFIP Networking [300]. At the time of development, no decentralised search mechanisms implementations for IPFS existed (besides hybrid approaches *e.g.* [202]). The project was implemented and released as command line interface, gateway service, and source code [53].

Deece Search provides an open and collaborative keyword search mechanism for IPFS, where any node running the protocol can crawl content on the network and

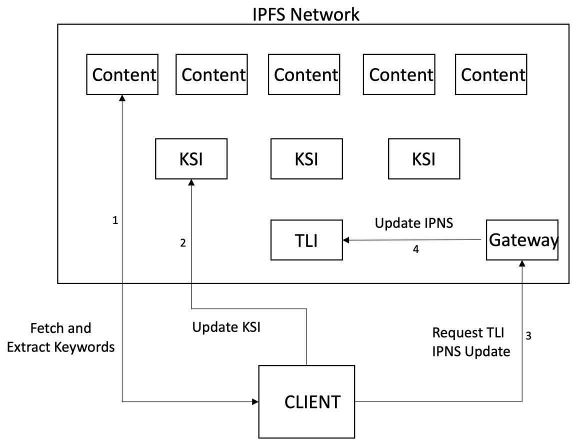

Figure 4.1: Adding crawled content to the top-level index in Deece Search.

add references of this to a decentralised index, stored on the IPFS network itself. Deece Search allows for decentralised search on decentralised content. The index is split up into a two-layer hierarchy, the first being the *Top Level Index* (TLI) and the second being the *Keyword Specific Index* (KSI), which are both stored on IPFS. The TLI contains the identifiers (CID) for the KSI for each keyword, and is constantly updated when a node submits a crawl result. This record is pointed to by an IPNS name record to ensure persistence. When crawling, the nodes add to the current KSI a list of the identifiers of files that contain that keyword.

Besides crawling, nodes can perform search queries, which leverages the latest TLI to find the KSI for each keyword in the user query. After fetching results, they are filtered based on decentralised ranking, *e.g.* based on the intersection of keywords.

4.2.2 Design

The proposed architecture relies on a collaborative network of nodes, as a subset of the total IPFS network, who collectively maintain a shared index and are able to perform searches. The system is built on top of the existing IPFS implementation, and therefore nodes can easily start participating, without having to run a separate P2P overlay.

The experimental implementation uses a trusted node and lacks security or incentivisation, following an altruistic model in the early stage. However, nodes

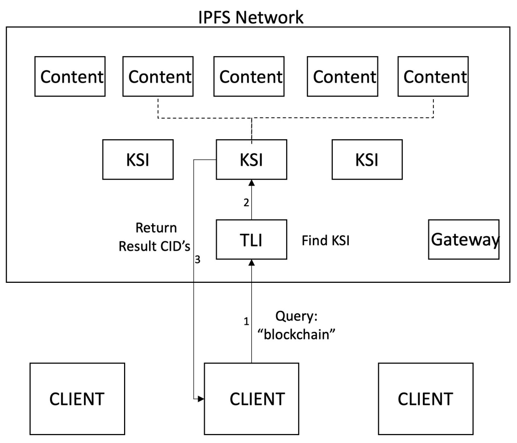

Figure 4.2: User submitting a search query in Deece Search.

need to be incentivised to be honest in updating the index, for which rewards may be used, possibly funded by a decentralised advertising market. The two main actions a node can perform in the network are search and crawl.

Search: A user starts a search by compiling a query which contains a number of descriptive terms (keywords), similar to a centralised approach. The node then fetches the latest TLI by resolving the IPNS[1] name set by the gateway to the corresponding CID, after which the TLI is traversed and checked against submitted keywords. If a match is found, the relevant KSI is queried, and results are returned to the user who can locally rank them. The user can then retrieve these files from the network.

The ranking mechanism is an essential component of search, which generally happens in a centralised manner. While sophisticated ranking mechanisms have not been implemented, clients are envisioned to locally rank, leading to greater control and transparency, as well as personalisation.

Crawl: In order to decide what content is added to the index, nodes can decide what they deem important information (in centralised search this is all Web content). In a crawl, a node fetches content and analyses it to extract important keywords. Besides extracting keywords, other metadata may be added. After extracting the keywords, a reverse word index is created, consisting of records of keywords and

[1]`https://docs.ipfs.io/concepts/ipns/`

all the content where they appear. Next, the index is stored on IPFS in a two-level hierarchy. Each keyword has an index file (KSI), and a separate index points to these records (TLI), which achieves persistence by using IPNS for naming. However, IPNS requires a private key to be updated, and hence uses a trusted node which holds the key (*i.e.* gateway). When a node updates the KSI's after crawling a file, they update the pointer in the TLI to these files, and requests the gateway to update the pointer which the IPNS record resolves to.

4.2.3 Key Findings

One key drawback of this project is that in its current form a trusted node is needed to update the naming records (IPNS) pointing to the latest version of the TLI. There is a trade-off between trust and security, as making the key available to more nodes increases decentralisation, but at the cost of possible security vulnerabilities. Mechanisms like zero-knowledge proofs or threshold cryptography could be useful, but this has been largely unexplored.

Furthermore, IPNS proved to be a serious performance bottleneck, which is due to the fact that nodes need to traverse all provider records in the DHT for a name record in order to ensure that the latest version of the index has been fetched, which becomes especially crucial when the TLI is often updated.

To summarise, this project found that a single root-of-trust is required for naming consistency, which can be described as the *top-level index problem*: Keyword-search systems storing their index on a DSN using self-certifying names face difficulty in tracking the latest index file. This can be solved using naming mechanisms like IPNS or name-registries, but this often leads to performance degradation, centralisation, or security vulnerabilities, analogous to the blockchain trilemma [97].

To overcome these issues a fully decentralised search mechanism is needed. The rest of this chapter focuses on how this can be implemented using an alternative model based on similarity search, which implements various search workflows including keyword-search.

4.3 Preliminaries: Similarity Search

Similarity search aims to provide a mechanism for identifying content with high similarity. When a query is submitted, the goal of the algorithm is to return a number of items with high similarity of content. This has been applied in a variety of use cases [301], and implemented either using space partitioning methods such as tree-based mechanisms [302], or using hash-based approaches like Locality Sensitive Hashing (LSH) [303]. As the latter has higher performance guarantees at lower overhead, this is the method used in Ditto.

The goal of LSH is to extract a short *signature* from content based on hashing, where the hashing algorithm maximises hash collisions for similar items. As signatures retain information about similarity, they can be used to perform comparisons and find similar items, rather than having to use the raw file, greatly improving performance.

Popular LSH mechanisms include random hyper-planes [304], multi-probe [305], and LSH forest [306]. In this chapter a minhashing [307] approach is used, which reduces a high dimensional content vector into a short signature using N random hash functions. First, a binary feature vector is extracted from the content, after which a large number of randomly permutated hash functions are generated, where the signature length determines the number of hash functions used. The values in the signature are then given by the position of the first row with a 1 entry following each hash function.

The minhashing approach maximises the probability that contents which are similar are mapped to the same bucket in the signature. In the general minhashing LSH algorithm, signatures are further split into buckets, creating a large number of hash-tables that capture part of the signature. When looking for similar content, a requestor checks all hash-tables to generate a candidate set, on which a similarity metric is applied to get the closest items. While a number of similarity metrics can be used [308] such as cosine similarity and Hamming distance, the minhashing approach estimates the Jaccard similarity [309], which is defined as the intersection over the union between two sets A and B:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

4.4 System Properties

This section discusses the assumptions made in the design of Ditto, a decentralised similarity search system. Then, the desired properties and possible use-cases of Ditto beyond similarity search are discussed.

4.4.1 Assumptions

This chapter has a focus on the networking aspects and feasibility of implementing decentralised similarity search for the DWeb, and while data pre-processing and LSH tools are used and discussed, they are not the main contributions. Rather, this work presents a system of known parts, applied to a novel environment and use-case. To achieve Ditto, a number of assumptions have been used in the design of the similarity search mechanism.

- A *search* is defined in the system as nodes submitting a large number of queries to the network, consisting of a number of parameters specifying the search, and in return they receive all relevant content in the network.

- The *network* in Ditto is defined as a collective of nodes in a P2P network who collaborate to provide search functionality. A node can simultaneously be a search node as well as a network node, but is not required to be both.

- When discussing *content*, this refers to data of various mime types, and stored on a number of different underlying P2P storage networks (*i.e.* multiple decentralised data sources). These collectively store a set of files and data, where each item is identified by a content identifier (CID). An agnostic approach is taken towards the storage network implementations and CID conventions. However, in the rest of the chapter, for simplicity, the focus remains on a network where the CID convention uses self-certifying names (which mirrors IPFS).

- A random distribution of files to nodes that store them is assumed (*i.e.* each participant decides independently which files to store without following any specific rules), although a DHT is used to store pointers to content providers (nodes who can provide the content).

- It is assumed that nodes in storage networks are incentivised to make their content available on a search system, and would therefore participate in a collaborative search system. The presence of arbitrarily malicious nodes in the P2P network is assumed, who may not follow the protocol for individual gain. No single node is trusted entirely by its peers. However, at least one neighbouring node is assumed to be honest (*i.e.* nodes are not entirely eclipsed).

4.4.2 Desired Properties and System Goals

In designing a decentralised search mechanism for the DWeb (specifically one based on similarity search), the following desired goals have been identified:

4.4.2.1 Search Flexibility

Since the DWeb is comprised of a large number of complementary protocols and content sources (*e.g.* many different DSN's and blockchains), a search mechanism should be flexible in supporting a wide range of content types (*e.g.* text and videos), as well as various storage networks. The naming convention of identifiers (*i.e.* signatures) should also be compatible with current DSN's.

Furthermore, keyword-search should be supported as this is currently the most popular workflow, but a number of other search use-cases should be implemented, which is described in Section 4.4.3. It is further desired that users have control over system parameters to allow for personalised search and recommendation services. For example, the query similarity threshold and ranking policies should be user-defined.

4.4.2.2 Decentralisation

There should be no central or trusted entity involved in the search mechanism, and global knowledge about all the files stored in the network should not be required to query or participate in system upkeep. Each participant should be able to calculate signatures of files and securely verify each step of retrieval.

In order to achieve decentralisation and realise an open system, participation or resource-constraint nodes should be enabled, and therefore the system should have low-resource usage, and should not require costly global synchronization.

4.4.2.3 Security

The search mechanism should be secure against malicious peers and attacks. Specifically, each step of the search process should be *verifiable*. Without verifiability, any malicious nodes could return fraudulent results forcing the requesting node to accept irrelevant files.

The calculation of signatures and the distance between them is deterministic and verifiable by requesting nodes. This property is required to ensure the correctness of the returned results, and each peer should be able to independently produce signatures. Finally, the integrity of the content itself should ideally also be verifiable, for example using self-certifying naming conventions, but this is dependent on the underlying content network.

4.4.2.4 High Performance Guarantees

For a decentralised search mechanism to be useful, it needs to have comparable performance compared to centralised alternatives. Here, performance refers to a number of properties. First of all, user queries need to be returned quickly without a high *delay* (users expect sub-second delays in current systems). Second, the results returned need to be of high quality (*i.e.* a high *recall* of the theoretical most relevant results globally), comparable to centralised search engines.

Finally, the *overhead* of calculating the signatures and storing them should be low, in order to limit the bandwidth, storage, and computational power that peers have to dedicate to the network. If the overhead becomes too large, peers may require extra incentivisation for participation in the network, adding a layer of complexity. Specifically, any network tasks should have sub-second delays on average user machines, and storage overhead should at most be comparable to other P2P network storage (*e.g.* storage networks).

4.4.3 Use-Cases

Ditto, the proposed similarity search system, can be applied in a number of DWeb use-cases. These are described below in order to provide additional context on the contribution of such a search system.

4.4.3.1 Decentralised Search Engine

One of the main use-cases for Ditto is to facilitate decentralised search and replace current search engine based workflows. Users can submit content queries and retrieve all network content which is similar to it. This is extended to traditional keyword-search by hashing keywords directly and producing signatures from extracted keywords alone. This is further discussed in Section 4.5.

4.4.3.2 Recommendation Engine

Ditto can be used to create a decentralised recommendation engine for services and applications (DApps), which gives content providers and users recommendations for similar content on the DWeb. For example, music or video services can use similarity search to propose relevant items for users based on content they enjoy, without running centralised machine learning or data collection. This may also be applied to achieve decentralised and privacy-preserving personalised advertisements.

4.4.3.3 Malware and Illegal Content Detection

DSN's can be used to store and distribute copyrighted and illegal content without control from centralised parties, flying under the radar of law enforcement. Distributors of this content may slightly change files from the original so they cannot be found using known image hash databases of illegal content. To combat this, similarity search can be used to do an extensive search of potentially illegal files comparing to these databases. Conversely, copyright owners can apply the same techniques to find copyright-infringing content, which they can report to law enforcement. It must be noted that while this content can be detected, it cannot be removed from the network, and hence local moderation techniques may be implemented.

A similar technique can be used to check files for malware, which involves

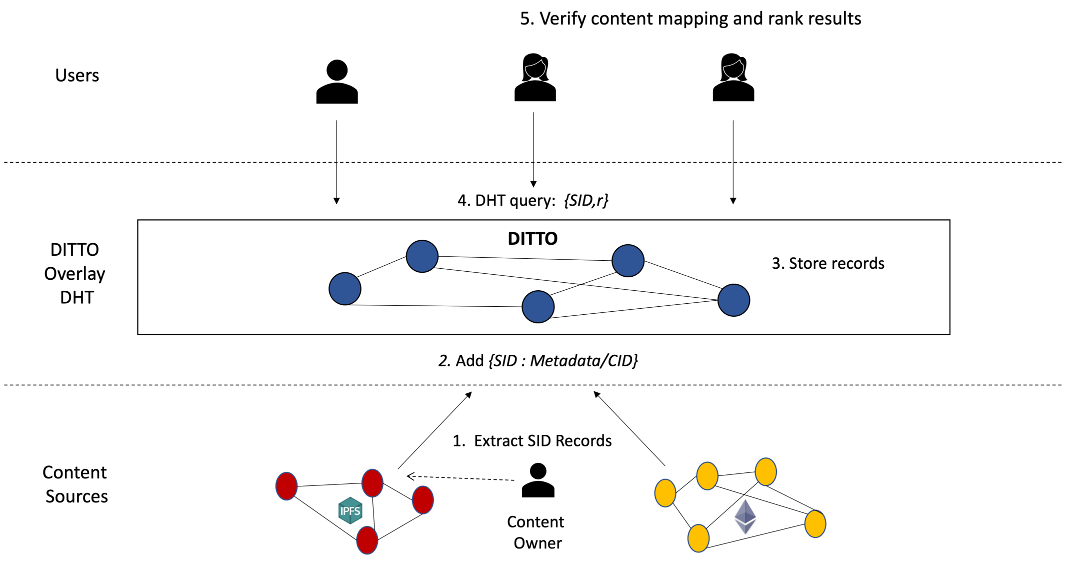

Figure 4.3: Overview of Ditto, where users submit queries and are able to query from various underlying content sources.

submitting signatures of content before opening them and checking for high similarity of known databases of malware and illegal content, allowing to asses whether the file is potentially hazardous.

4.4.3.4 Decentralised Moderation

As mentioned above, malware, illegal, and copyrighted content can be detected using similarity search. However, there are more types of content which may not be desired by a user based on their individual preference. For example, hate speech or extremist views may want to be avoided. As complete censorship in the network is not possible (similar to illegal/copyrighted content), users may implement local decentralised moderation strategies, which may avoid or assign low ranking to results which are similar to content that the user has flagged to be unwanted. This method allows for transparency and control of censorship and result ranking.

4.5 Architecture Design

4.5.1 Overview

This section describes the architecture design of Ditto, which captures the desired goals outlined above. The presented system is agnostic to content data type and underlying DWeb network, but to illustrate its functionality terminology from the IPFS DSN is used for simplicity. First, an overview is given of system functionality, before describing individual components.

As described in Section 4.4.1, Ditto takes inputs from different decentralised data sources. When a content owner uploads content to the network they simultaneously extract a *similarity identifier* (SID) based on an LSH signature. These content owners are incentivised to produce these identifiers for public content, as otherwise, their content will not be searchable in the network. Signature generation is verifiable by the network, as nodes can easily recompute SIDs. The SID is structured such that it supports different data types and content networks.

To store the decentralised index, an overlay network based on the Kademlia ID space is used, but incorporating the Jaccard similarity as the default distance metric. Nodes in the network store content records close to them in the ID space in the form of {*SID:Metadata/CID*}, where metadata specifies the content source and type, and CID is the identifier to query the content itself. When searching for content, a user sends a query to the overlay network consisting of a SID and a parameter r, specifying a *similarity range*. The SID can be in the form of the signature of content for which users want similar items or recommendations for example, or keywords to implement semantic search, depending on the use-case. The network then returns all results which are within the *similarity range*, and the user can filter these based on local ranking policies (*e.g.* based on content type or size) and then fetch them on the underlying data source. Figure 4.3 gives an overview of Ditto.

4.5.2 SID Generation

To generate a decentralised search index on the DHT, first SIDs need to be generated. To achieve this, an LSH signature is extracted and then arranged into a SID record format.

4.5.2.1 Signature Generation

In order to generate an LSH signature, separate pipelines are needed for different content types, as the pre-processing and hash extraction should be optimised per

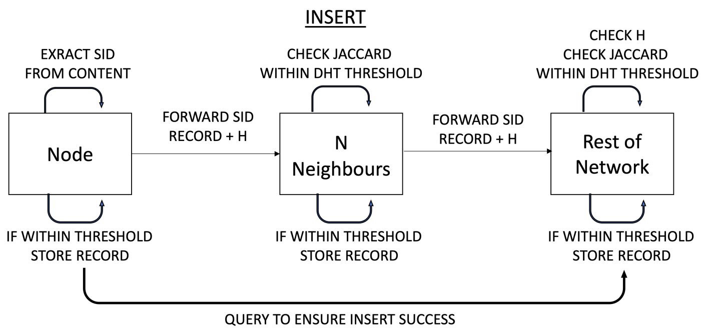
INSERT
EXRACT SID FROM CONTENT
CHECK JACCARD WITHIN DHT THRESHOLD
CHECK H CHECK JACCARD WITHIN DHT THRESHOLD
Node
FORWARD SID RECORD + H
N Neighbours
FORWARD SID RECORD + H
Rest of Network
IF WITHIN THRESHOLD STORE RECORD
IF WITHIN THRESHOLD STORE RECORD
IF WITHIN THRESHOLD STORE RECORD
QUERY TO ENSURE INSERT SUCCESS

Figure 4.4: Overview of inserting a similarity record.

type. For each content type, the dimensionality of data is first reduced and standardised, after which the reduced data is used to generate a feature vector, capturing the uniqueness of the content. Feature vectors are then parsed into the LSH algorithm, where parameters are tuned per mime type, and which (in the text case) is based on Minhash. The evaluation in this chapter focuses on text files, as this is the base for most Web searches (HTML is text). Describing the pre-processing for each specific type is out of scope.

In order to provide deterministic outcomes, the same LSH parameters should be available to the whole network. This could be done for example using blockchain solutions or a decentralised autonomous organisation (DAO) [211], allowing flexibility in changing the parameters. Alternatively this can be hard-coded in the protocol, although this is not desirable.

4.5.2.2 Addressing

One of the main benefits of the proposed approach is that it allows for a general DWeb search, rather than focusing on a single content source. This is achieved by formatting SID records to include metadata. In the hash-table entry the SID is the key, and the value is given by the path: *content_type/content_source/CID*. For example an IPFS image would have the entry {*SID:Image/IPFS/CID*}. Using this addressing, the security properties of the CID naming convention are maintained, while this could add additional useful metadata such as content title, adding human-

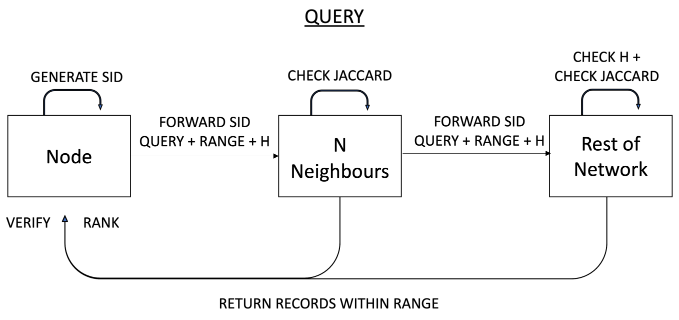

Figure 4.5: Overview of a user query for a specified SID.

readability.

4.5.3 Network Implementation

After generating the SID for a content item, the content owner stores the corresponding record on the DHT overlay network. Nodes close in the hash space to the content store the record, and respond to queries if their records show any content within range *r*.

4.5.3.1 ID Space

In order to structure the network peers, the Kademlia ID space is used which can be visualised as a binary tree. A node has more knowledge of close nodes in the ID space which speeds up look-ups. When nodes join the network, they randomly generate an ID in the space and starts building its routing table buckets.

4.5.3.2 Insert

Inserting and querying the DHT follow slightly different algorithms, because insert is not as time sensitive compared to serving user queries. In this protocol, another key parameter is the *DHT threshold*, which specifies the closeness of a node to a record for it to store it. This is because the Jaccard metric has the symmetry and triangular properties like the XOR metric, but it is not unidirectional (*i.e.* the distance between a node and any two others nodes may be the same), which means there may not be a closest node, but a number of closest nodes. The *DHT threshold* can be adjusted to tweak the amount of caching in the network.

To insert a record into the distributed index, a node checks the Jaccard similarity to itself and their direct peers. They then place the record with any node within the threshold, and send it to the *N* closest nodes (who also forward it to their closest nodes) in terms of Jaccard similarity. Along with the record, the node sends a parameter *H* to indicate after how many hops a receiving node can discard a request, so it does not stay live in the network. Nodes receiving records verify the correctness of the {*SID:CID*} mapping before adding it.

After a time period, the node makes sure the record has been added to the network, and if it cannot find it, the above steps are repeated. Parameters *H* and *N* are again set on a protocol level, similarly to the *DHT threshold* and LSH parameters.

4.5.3.3 Query

In order to query the network a node first needs a SID in which it is interested, for example based on similar content it has (as a recommendation engine). It sends the SID and a *similarity range* to its peers.

Each node which receives the query checks their local index for any entries which fall in the range, which they return to the sender. They also send it to their top N closest nodes until the termination condition H is met. In order to ensure security the node may recompute the distance of results to ensure it is correct. Alternatively, this system could easily be extended to return the top K closest results to an SID.

Ditto also implements *decentralised ranking*, as the querying node is responsible for compiling the results and ranking them based on a local policy. This can be based on similarity to the query, size, file type, or content network for example. However, the system can be extended to include metadata in the index files such as keywords, and content producer name, which allow for more fine-grained ranking mechanisms.

4.5.3.4 Keyword-search

Ditto, as described above, can be modified in order to allow for keyword-search. Rather than supplying SIDs of known content, a node can compute the SID directly based on the keywords it is interested in only. A system is desired where a user provides a number of keywords, which are then hashed into a signature, and which can then be queried in the network to receive relevant SIDs. However, if the system described above is implemented directly (with signatures in the SID based on the content shingles), it is likely that there will be extremely small similarity scores to compare. This is because the keywords are not weighted in the shingles and therefore their importance in the signature is not captured.

Instead, the most important keywords from content can be extracted and used directly to produce a signature and SID, allowing for a better comparison of Jaccard similarity. This could be achieved using natural language processing tools for example, similar to crawling. A drawback of this approach is that it requires a stronger mechanism to ensure the SID record mapping to content is correctly calculated (*e.g.* ensure that the keywords used in the signature are correct). This could be done using zero-knowledge proofs, which is discussed further in Section 4.6.5. The feasibility of keyword-search in Ditto is explored further in the evaluation.

4.6 Evaluation

In this section the feasibility of Ditto is examined and the design goals outlined in Section 4.4.2 are verified. As this work is the first to explore LSH as a general search mechanism including keyword-search for DWeb settings, the main focus is on its application and feasibility, leaving further evaluation of network and DHT aspects for future work. Specifically, the desired properties defined in Section 4.4.2 are assessed: the performance of the decentralised mechanism is measured, the search flexibility is tested using keyword-search, and the security aspects are analysed.

4.6.1 Setup

For the evaluation[2], a Wikipedia mirror is used which represents a large dataset of content which can be found on the IPFS network. From this, a subset of the Wikipedia articles is taken in Dutch from the archive[3] to use as the dataset in the rest of this evaluation. This dataset is chosen to model popular Web content, and the

[2] `https://github.com/navinkeizer/ditto`

[3] `https://wiki.kiwix.org/wiki/Content_in_all_languages`

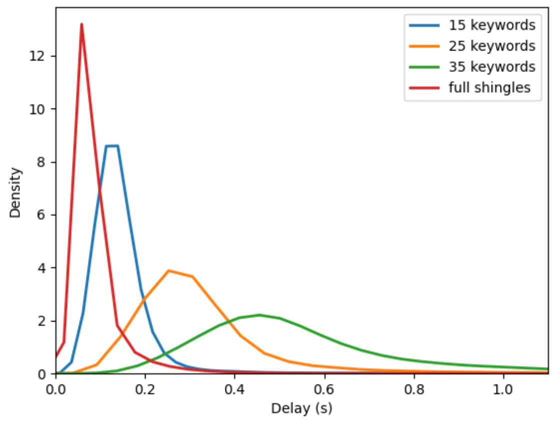

Figure 4.6: Delay (s) of signature generation for full signatures and different top keyword sizes.

	Mean	Standard Deviation
XOR	0.518	0.42
Jaccard	0.942	5.94
Euclidian	2.011	1.21
Hamming	2.029	2.38
Cosine	5.450	7.11

Table 4.1: Delay of similarity calculation with different distance metrics (in 10^{-5} s).

subset is taken to assess the scalability of the mechanism. The LSH implementation used can be seen as a lower bound, as no optimisations are proposed, and therefore the main goal of this evaluation is to establish that there is value in using LSH as a DWeb search mechanisms.

To calculate LSH signatures, a Minhash implementation in Python[4] was used. First, the text from a page was extracted in order to produce shingles with length k, where k is set based on the length of the text (generally k=6 for the data used). The algorithm then calculates the signature based on the specified number of permutation functions. The simulations were performed on a local laptop with average specifications.

[4]`https://github.com/ekzhu/datasketch`

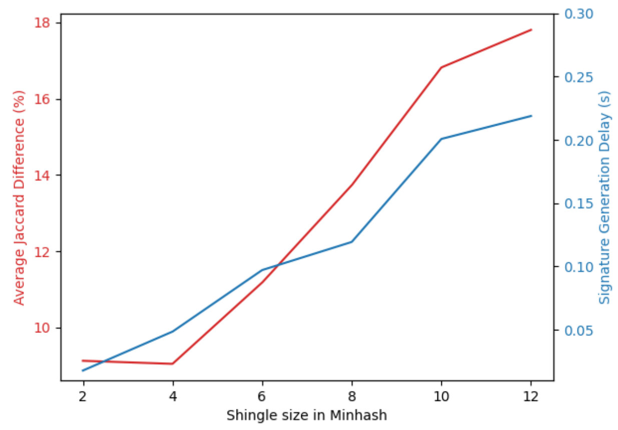

Figure 4.7: Jaccard similarity difference and signature delay for varying shingle size.

4.6.2 Similarity Metric

The proposed system relies heavily on the calculations of Jaccard similarity for querying and inserting content into the decentralised index. To verify that this does not deteriorate performance significantly, the performance of this metric is compared against other known similarity metrics.

A random subset from the dataset is selected and signatures are computed from them, on which the delay in computing a similarity score is measured. As shown in Table 4.1, Jaccard similarity is relatively fast and comes close to XOR performance. While XOR is taken as the baseline because of its speed and use throughout DHT-based solutions, it must be noted that XOR is not applicable in this system as it uses the longest prefix match, instead of comparing sets within the signature. XOR is not suitable as this may result in a low distance in signatures, even though there is a lot of overlap in the content (but in different positions). This is a problem particularly for keyword-search, and for this reason the rest of the evaluation focuses on the LSH specifics, rather than performing a comparative analysis with an XOR-based DHT.

4.6.3 Minhashing Performance

Next, the performance of the minhashing algorithm is assessed in terms of the delay and accuracy in translating the information in the original document to a short

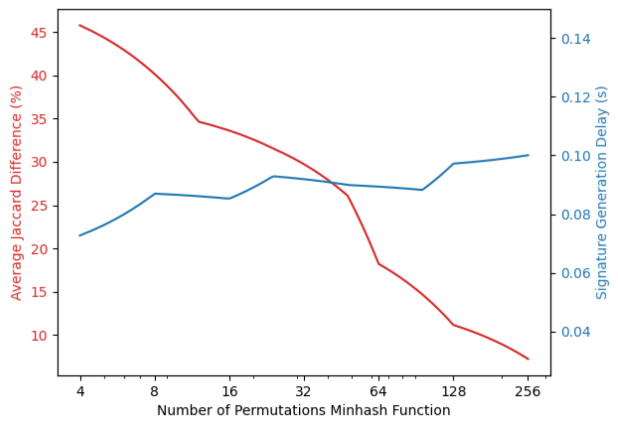

Figure 4.8: Jaccard similarity difference and signature delay for varying number of permutation functions.

signature. Two subsets are extracted from the dataset on which signatures are generated, which are subsequently stored along with the raw shingles. Content pairs are then randomly picked, on which the Jaccard similarity of the raw shingles is compared against the signature. The percentage difference between the two similarity scores is measured, along with the delay of generating signatures.

Figure 4.7 shows how both the delay and the difference (accuracy) go up with higher values of k, suggesting that for this particular use-case a smaller k-value is appropriate. Figure 4.8 shows that the delay and accuracy are inversely correlated with higher numbers of permutation functions. However, in either case the delays are reasonable as it adds sub-second delays to the overall latency, which are only incurred when adding or verifying content. Therefore, this delay is also not incurred often and meets the high performance goal.

4.6.4 Search Performance

Finally, the performance of search using LSH is evaluated. From the dataset, subsets of different sizes are used in order to analyse scalability. The main metrics measured are overhead, query delay, and the recall —which is defined as the percentage overlap of items returned compared to baseline (*i.e.* on raw shingles). The reason these metrics are chosen is because they directly impact the performance and user experience, and have therefore been used in related work to assess search

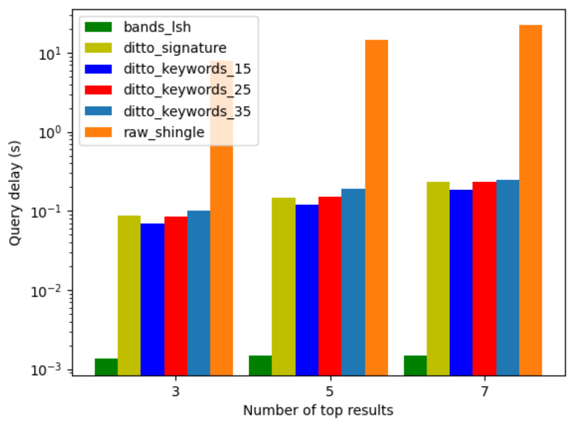

Figure 4.9: Query delay (s) of search mechanisms for varying number of top results.

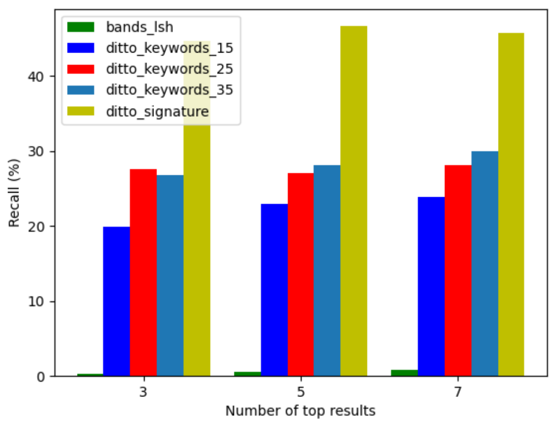

Figure 4.10: Query recall (%) of search mechanisms for varying number of top results.

mechanisms. As the baseline a search system based on comparing similarity of raw shingles is used, which is compared against an implementation of traditional Minhash LSH using bands, as well as Ditto's signature comparisons (including keyword search).

After taking a randomised subset, shingles and signatures are parsed into the various search mechanisms. Next, a smaller subset is taken to be used as query data on which the performance of each mechanism is assessed (as the test data for the simulation). The sorting/searching algorithm has not been optimised for the stored LSH data, so performance can be seen as a lower bound.

Besides these implementations, **keyword-search** is implemented by extracting the top keywords from the text file using Automatic Keyword Extraction[5] and hashing these directly. Queries are then performed on keywords only (extracting the top keywords from the query subset), which is then compared to the other implementations.

In terms of overhead, storing signatures instead of raw shingles reduces the size in memory by about 10^3 times. Figures 4.9 and 4.10 show respectively the delay and recall for varying the number of results returned by the search. While the delay is very low for lsh_bands, the recall is extremely poor. This implies that this mechanism is fast in returning results, but that the quality of results is not suffient. Comparatively, signature_lsh approach performs much better, and keyword-search also works well, especially when setting the number of top keywords extracted for the signature at 35. Using raw_shingles is not a feasible option as there are large delays which grow linearly.

When sweeping across the number of content in the system in Figures 4.11 and 4.12, the same can be observed, where delays grow linearly, but remain manageable for signature_lsh, and are very low for keyword-search. In terms of recall, signature_lsh performs best, while keyword-search also achieves reasonable quality. In terms of generation delays, Figure 4.6 shows that using full signatures generally achieves delays of under 0.1s, while the keyword signature delays grow with larger numbers of keywords extracted due to the extraction algorithm overhead.

To summarise, it has been shown that keyword-search is a feasible option because it has comparable delay, recall, and overhead compared to native similarity search, but extends the use-case to allow users to submit search descriptions, capturing the first design goal of search flexibility.

Furthermore, the proposed system achieves the goal of high performance, as delays are sub-second, and recall is between 30-50% of baseline. It is also lightweight enough with low-resource usage to be deployed in a decentralised setting. Section 4.6.5 discusses how the security property can be met in more detail.

[5]`https://github.com/LIAAD/yake`

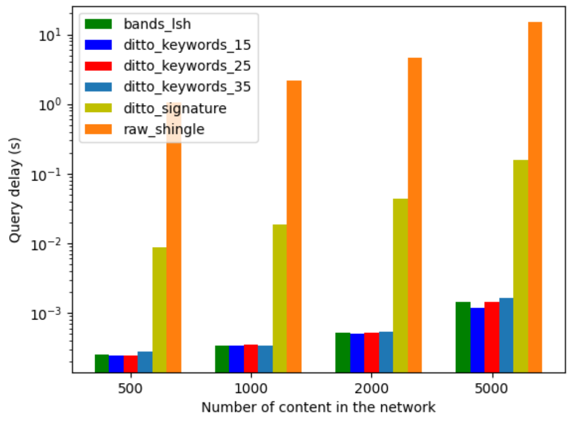

Figure 4.11: Query delay (s) of search mechanisms for varying number of content in the network.

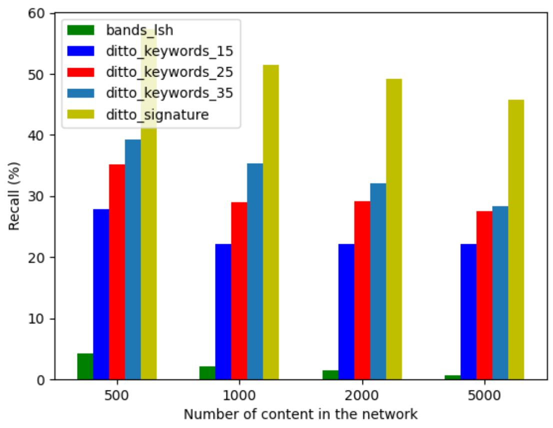

Figure 4.12: Query recall (%) of search mechanisms for varying number of content in the network.

4.6.5 Security Analysis

As mentioned in Section 4.4.2, one of the desired properties in this work is security, both in terms of SID record mapping, as well as protection against malicious peers, who may not respond or return incorrect content records, or may not participate in network tasks like computing SIDs. Ditto mitigates against these threats in a number of ways.

First of all, peers who upload content as public files are incentivised to calculate the SID mapping, as their content otherwise will not be discoverable. Second, using Jaccard similarity for inserting along with the *similarity threshold* parameter means that there will be redundancy in the network in the form of caching, as multiple close nodes store SID records. This means that even if one or more nodes are malicious, there is still a high probability of finding the file from an honest peer.

In the querying process, nodes can verify that returned results are actually within the similarity range wanted with a lightweight Jaccard check. Furthermore, the mapping from CID to signature is verifiable as the algorithm and its parameters are set on a protocol level and can be computed by any peer. When using keywords directly, this mapping can also be verified if the keyword extraction is standardised. However, additional mechanisms like zero-knowledge proofs may be used, avoiding the need to re-verify integrity at all nodes who come across the file.

4.7 Summary and Discussion

This chapter presented Ditto, a decentralised search mechanism for the DWeb based on similarity search. Ditto supports numerous decentralised content networks and types, and uniquely implements decentralised keyword-search. The evaluation verifies the feasibility and shows that the system goals of flexibility, decentralisation, security, and performance are met. A search recall of up to 57% recall of baseline is reached, and sub millisecond delays are achieved in keyword search for a network size up to 5000. Ditto opens up a new research direction of search for the DWeb using LSH, and specifically based on keywords. While these initial results are promising, improvements in terms of search delay, overhead, and quality of results are required in order to compete with centralised approaches.

While this chapter focuses on the feasibility of decentralised search, the networking specifics of this approach should be explored in future work. Particularly, design specifics and evaluation of the proposed DHT should be investigated, as well as unstructured and hybrid approaches for implementing decentralised search, in order to improve performance. Another area which needs to be worked on further is the security implications and mitigation, *e.g.* those based on verification like zero-knowledge proofs, or network consensus.

Another area which has not been covered in depth in this work is incentivisation of network participation. The argument was made that users do not perform much additional work and therefore incentives are not as crucial as for example in blockchains. It may still be desirable to add monetary rewards to add robustness and security to the system. A natural extension of this work could leverage a monetary inflow source backed by a decentralised advertisement market, where advertisements are mixed in with search results. Using similarity search personalised advertising could also be explored, analogous to the current Web model.

Chapter 5

Service Allocation for Decentralised Network Resource Sharing

5.1 Overview

Many decentralised services have recently emerged on top of blockchain allowing any node in the network to share its resources. In order to be a competitive alternative to their central counterparts, their performance needs to match up. Specifically, service allocation remains a performance bottleneck for many decentralised services.

This chapter presents FLOCK (Fast, Lightweight, and Scalable Allocation for Decentralised Services on Blockchain). On top of being highly scalable, fast, and lightweight, it allows nodes to indicate their preference for clients/sellers without needing to submit bids by using stable matching algorithms. Price discovery is decoupled and this function is outsourced to a smart contract on the blockchain. Additionally, another smart contract is used to orchestrate the allocation and take care of service discovery, while trusted execution environments securely compute allocation solutions, and off-chain payment networks are used to send rewards.

Evaluation of FLOCK shows that gas costs are manageable and improve upon other solutions which leverage auctions. Furthermore, the instance of the stable matching algorithm used in this work greatly improves run-time and throughput over auction counterparts. The end of the chapter discusses practical improvements to further increase performance.

5.1.1 Introduction

Ongoing efforts in the DWeb aim to decentralise traditionally centralised Web services, and offer alternatives built on concepts like blockchain and DSN. Main examples of these services are network resource sharing services (NRS), which focus on data storage, computation outsourcing, and bandwidth sharing. An important challenge for the functioning of these services is allocating nodes requiring services to ones that offer the corresponding services, while maintaining decentralisation. Although a brute force allocation approach, whereby nodes requiring and providing services individually discover each other, can be viable with limited participants, a *scalable* allocation mechanism is needed in order for the decentralised services to compete against their centralised counterparts.

In conjunction with scalability, two other desirable features for an allocation mechanism are *price discovery* and support for *preference-based* assignment. The price derivation determines appropriate rewards for nodes providing services based on market dynamics. Enabling nodes requesting services to indicate their preference over others in the allocation process is highly desirable in a decentralised system where any (potentially malicious) node can offer or request services.

These three properties, namely scalability, price discovery, and node preference, are in practise difficult to achieve simultaneously. For example, decentralised auctions, which have been proposed as a solution to this problem, inherently lack sufficient scalability.

This chapter presents FLOCK, an allocation system for decentralised services on blockchain, which captures all three desired properties. Ideal use-cases for this system are services with large volumes and tight time constraints such as content retrieval in decentralised storage markets. Unlike heavyweight auctions that combine price derivation with node assignment, FLOCK achieves lightweight and scalable allocation by decoupling its price discovery from the preference-based assignment, outsourcing this market function to an oracle smart contract. This oracle sets a global price per service, and can be triggered by any node in the network.

FLOCK uses stable matching algorithms to allow for node preferences, where nodes submit a partial preference ordering of nodes offering or requesting a service to a billboard contract. The computation of matching is outsourced to an off-chain trusted execution environment (TEE). This keeps smart contract costs low, and ensures privacy and speed. The TEE is compensated by the allocated nodes using an off-chain payment network, ensuring speed, low cost, and scalability.

Implementation and evaluation of FLOCK's contracts shows that gas costs are sufficiently low, especially compared to auction-based solutions (reducing cost by at least 52%). Comparing the proposed algorithm to auction algorithms demonstrates that FLOCK scales much better in terms of computation run-time and throughput, staying under 0.5 seconds for 9000 clients.

5.1.2 Related Work

Work directly related to FLOCK is now discussed. Stable matching algorithms have been widely studied and applied in real world settings [310]. Examples of this are in cloud resource allocation, student roommate allocation, and hospital resident allocation. The rest of the related work is divided in two sections: outsourcing computation using TEEs, and auctions on blockchain.

5.1.2.1 Computation Outsourcing Using TEEs

TEEs are used as secure computation enclaves (see Section 5.2.1), and are used in a variety of security use-cases, including off-chain computation outsourcing. One of main downsides of smart contract computation is the lack of confidentiality and privacy, which are essential for use-cases like sealed-bid auctions, because their code is stored on-chain which anyone can access. Therefore most works optimise privacy rather than scalability. The overheads in these solutions result in slow execution times, and are therefore not suited for fast and scalable allocation.

ShadowEth [311] presents a framework for leveraging TEEs to confidentially execute smart contracts off-chain using a bounty contract on-chain and distributed storage. Airtnt [312] and SPOC [313] outsource computation using TEE's, off-chain payment channels, and a smart contract, targeting fair exchange of resources, security, and correctness. One-to-one allocation of clients to workers is assumed (rather than multi-input, single output). Ekiden [314] is another system that uses TEEs to address the lack of confidentiality and poor performance of blockchains, by separating execution and consensus.

Although it does not leverage TEEs, Hawk [315] presents a decentralised smart contract system that tackles the privacy issue. Hawk private smart contracts can be programmed without cryptographic primitives, which are added by the compiler. Finally, Kosto [316] is a framework for secure computation outsourcing using TEEs, and uses a wrapper function for accounting, producing a proof of computation. A broker node allows for minimal open payment channels.

5.1.2.2 Auctions on Blockchain

There has been much work in auction mechanisms. These have traditionally been applied in a centralised manner, as decentralised auctions are difficult to orchestrate. Blockchain has made it possible to implement decentralised auctions on top of smart contracts, either on-chain or outsourced to a dedicated node to keep gas costs low. Because sealed bid auctions require strong privacy guarantees, they often require expensive cryptographic overheads, lowering their scalability. The following works focus mainly on ensuring privacy and accountability.

AStERISK [317] presents a single item Vickrey [273] auction on smart contracts. Distributed authority is used for issuing bidding credentials, as well as a number of cryptographic operations. Enkhtaivan et al. [318] implement an anonymous English auction using TEE and blockchain, and use group signatures to provide bidder anonymity.

Desai et al. [319] propose a hybrid auction mechanism on blockchain, combining public and private chains, and using simple cryptographic proofs. An auctioneer starts, orchestrates, and deploys the auction. Private chains lower the cost and latency, but the cost of the public portion of the smart contracts and role of auctioneer lower security and scalability.

Galal and Youssef propose a number of sealed bid auctions [320, 321]. Most recently, Trustee [322] presents a Vickrey auction on Ethereum using TEEs. The smart contract is used as a billboard, after which computation is transferred to an enclave using a relay controlled by an auctioneer. The auctioneer can be untrusted and gains no bid information.

The works mentioned above are not applicable for fast and scalable allocation, as they remain expensive in terms of costs and overhead. Besides, they are all single-item auctions, while large numbers of nodes need to be allocated at once. PASTRAMI [323] is a decentralised multi-item auction relying on the Vickrey-Dutch Multi-Item Auction [324] to derive the Vickrey-Clarke-Groves (VCG) equilibrium, allowing multiple buyers and sellers to be matched in one round. After assembling bids on the smart contract, a dedicated node performs the computation to minimise gas fees. All nodes can check the solution and submit proofs of misbehaviour. PASTRAMI is still not scalable enough for the targeted use-case as there are high gas fees and latency, and it suffers from general inefficiencies associated with auctions such as the delay of valuing items and gathering bids.

Finally, a number of papers propose using Secure Multi-Party Computation to create decentralised auctions (e.g.[325, 326]), but these are highly unscalable due to the expensive cryptographic functions used and its high interactivity.

5.2 Preliminaries

Key concepts related to FLOCK are now discussed: TEEs, and stable matching algorithms.

5.2.1 Trusted Execution Environments

TEEs allow secure computation to be performed on remote untrusted nodes, using hardware enclaves [327]. The code to be executed in the enclave can be verified for correctness, and external access to the enclave is protected against. Among others (e.g. [328, 329]), Intel software guard extension (SGX) [330] is a TEE implementation which allows users to upload and execute code into a tamper proof secure container, called the enclave. After being uploaded, the code cannot access the OS functions. SGX allows code to be attested to prove it is running properly. This can be done via remote attestation or using intra-attestation. Furthermore, enclave data can be sealed outside of the secure memory.

Although there are many benefits of using SGX, there are known drawbacks as well [331]. First, there is limited secure memory of 128MB. There can also be availability failures as the platform owner can (maliciously) terminate an enclave. SGX can also be susceptible to side-channel attacks [332] and single point attacks. Section 5.5.4 discusses why these drawbacks are not a problem in FLOCK. SGX is used as the TEE in the rest of this chapter.

5.2.2 Stable Matching Problem

Stable matching algorithms have been used in a wide range of applications [310], as a mechanism to pair entities from two sets based on their preferences for each other, without monetary bids. In its most basic form the problem is also known as the stable marriage problem, with a set of men and a set of women with preference ordering of each man or woman in the other set. Gale and Shapely [333] provided a simple algorithm to produce a stable matching, and the problem has since had many extensions, including partial preference lists, student-roommate allocation, and student-project allocation.

More notably, stable matching has played an important role in hospital resident allocation [334], where the simple problem is extended. Residents are allowed to include in their preference list any subset of the hospitals, and hospitals rank all residents that ranked them. Furthermore, all hospitals have a budget of residents which they can accept.

One feature of stable matching algorithms is that the set of nodes which takes the initiative in the algorithm produces a matching which is best for that set, and thus has an advantage. This yields that the second set is not incentivised to be truthful.

5.3 System Goals

This section describes desirable properties an allocation system should have, and discusses how other initiatives have failed to capture these. Next, the allocation landscape for decentralised services is sketched from which two types of allocation

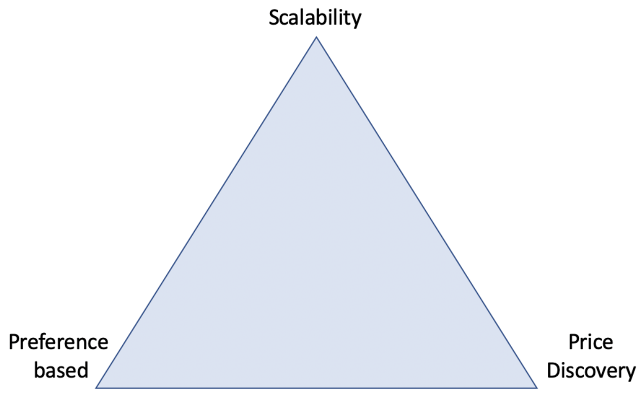

Figure 5.1: The allocation triangle, showing the desirable properties for decentralised blockchain-based service allocation.

can be derived. Finally, decentralised storage network use-cases for decentralised allocation are explained.

5.3.1 Desirable Properties

An allocation mechanism for DWeb NRS services should have several properties on top of decentralisation. First, the solution needs to be scalable, especially when user numbers grow into the millions (e.g. BitTorrent[1]). This scalability is in terms of latency, throughput, and cost. Second, any node offering a service is potentially malicious. To mitigate against the risk of attacks and insufficient service, nodes need to be able to indicate a preference over their allocated node. Last, as there is no centralised sale of the service, there needs to be some way to set prices. This price discovery should be based on market dynamics to reflect supply and demand, but can be extended to include other factors. Surveying previous works shows that it is been particularly difficult to achieve all of these properties simultaneously. To illustrate, the allocation triangle in Figure 5.1 can be used.

Normally, decentralised storage networks like Filecoin [44] use an on-chain orderbook to match clients and sellers based on their bid price. This approach would only capture the price discovery aspect of the triangle as it is not scalable and there is no method for specifying a preferred node in a bid. Derived prices are different for individual storage instances (i.e. storage is seen as heterogeneous and can be

[1] https://www.bittorrent.com/company/about-us/

priced different for different nodes).

Auctions generally improve on this as they capture both price discovery and user preferences through bids. The downside however is their scalability, especially in low latency settings. To start with, the preparation phase of auctions (valuating items and gathering bids) in itself incurs a time delay too significant for instant allocation. The execution phase increases this delay further.

In Vickrey auctions, large numbers of allocations require many subsequent auctions, while the performance of multi-item auctions such as the VCG quickly degrades with increasing numbers of participants and items. These auctions generally require extra privacy measures to keep bids sealed, as they need to mitigate against a number of attacks. These attacks become less lucrative when no money is involved in the allocation, as is the case with stable matching algorithms. These capture the user preferences, and are faster and more scalable than auctions. In Section 5.5.1 this is combined with price discovery to achieve all three desirable properties.

5.3.2 Types of Allocation

In an allocation, it is assumed that there is a large number of clients and sellers. Generally, the number of sellers will be less than the number of clients. These sellers, can offer many different types of goods, depending on the decentralised application. It is envisioned that for each large decentralised application there will be a separate allocation to which clients and sellers flock.

Two general service types are considered which need fast, scalable, and lightweight service allocation:

1. Services where a seller sells a single instance of a service. This service can be denoted as binary: either it is provided or it is not.

2. Services where the seller sells a part of a good within a certain capacity. Sellers have a budget and try to maximise their revenue by selling in smaller chunks.

These two service types require different modifications to the allocation algorithm used. For the first type, it can be modelled as a regular stable marriage problem, but extended using incomplete preference lists, which ensures that nodes do not need to submit preference orderings over large sets of users. This will produce a matching of clients to sellers for the single service instance.

The second type can be ideally modelled as the hospital/residents problem. This again allows for incomplete preference orderings from the client side, and allows the sellers to define a budget they have available. Clients can be assigned to this seller as long as it has remaining capacity, which is especially relevant when sharing network resources.

5.3.3 Decentralised Storage Use-Cases

The service types defined above can be easily illustrated using decentralised storage and retrieval markets (similar to Filecoin). Typically, such networks use the storage market to sell storage to client nodes in storage chunks (in GB) within their capacity. This is a clear example of the second service type, and can be modelled using hospital/residents allocation.

The retrieval market on the other hand is used by nodes in the network to fetch specific content from one of the storage nodes. In this case the client contracts a retrieval miner, which either delivers the file or not (for which it receives some off-chain payment). This is an example of the first service type.

The rest of the chapter focuses on these two service types, and for the evaluation specifically on the second type. This has been done for simplicity and clarity, but the solution can easily be adapted to support more complex cases. One such extension is as follows.

The retrieval market has stringent time constraints, as a client does not want to wait long before fetching a website for example, and hence performing the allocation as requests come in might not be fast enough. In Section 5.7.3 a method is discussed for performing the allocation before requests come in, by allocating retrieval nodes to content providers rather than clients to sellers.

5.4 Trust and Reputation for Decentralised Web Services

While the blockchain can be used to establish trust for transactions on-chain and implement allocation and auction mechanisms, the NRS service itself is provided off-chain and occurs directly between two parties. Therefore, an honest majority of the network does not automatically guarantee security in transactions. For example, a provider node may commit to a service, but not actually provide it. It does not result in additional rewards, but does result in negative consequences for the client in the deal, leading to a risk of malicious conduct at zero cost.

To discern between honest and malicious parties and incentivise honest transactions a *reputation system* is needed. As discussed in Section 2.5, a reputation system generates scores for network participants which indicate the trust in a likely positive experience of a transaction. A score is produced by aggregating various metrics and using a particular scoring mechanism. Current reputations for centralised and early P2P services are not suitable for a DWeb application, as they rely on a single root-of-trust, have excessive complexity, and suffer from security vulnerabilities. They also do not incorporate DWeb and blockchain transaction data in their scoring mechanisms.

5.4.1 DWeb Reputation System Requirements

Currently, NRS services either do not use a reputation system, or rely on a centralised service, forming a single point-of-failure and lacking in transparency and control (in terms of metrics and scoring functions used). For example, Storage.Codefi[2], a 3rd party Filecoin Dashboard, puts more emphasis on storage faults than ask price when calculating reputation scores, which may not represent all user preferences and their risk aversion.

A decentralised reputation system is needed for DWeb services. Due to the close integration of decentralised services and blockchain, a DWeb reputation system should *use on-chain data* to derive metrics for computation of reputation

[2]https://storage.codefi.network

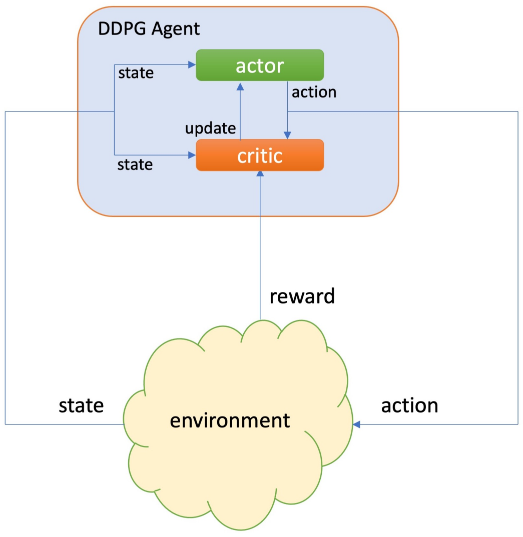

Figure 5.2: Overview of the DDPG algorithm, which could be used for reputation scoring.

scores. Furthermore, the system should be *accurate, quickly converging, secure*, and *lightweight*.

Finally, reputation scores are highly subjective metrics. One node's view of a sellers reputation might be different from others, based on personal preference and experience. In fact, a single seller may be seen reputable as a storage node, but not so much in other services. Therefore, a highly *personalised* reputation service is needed, which self-adapts based on a personal preference profile.

5.4.2 Artificial Intelligence Based Reputation

This section proposes to use artificial intelligence (AI) - specifically deep reinforcement learning (DRL) - to assist in calculating personalised reputation scores. This meets the requirements as described above, creating a decentralised reputation solution for DWeb services. Using DRL allows us to take into account a wide range of metrics to produce an optimised scoring function, as the algorithm continually finds the ideal weights, and may take into account otherwise overlooked metrics. The proposed system is an early stage design, with evaluation needed in future work.

Reputation calculation is comprised of two parts: gathering information and computing scores. The proposed solution uses blockchain data directly as input, as most NRS service deals are stored on-chain, and much can be inferred from past transactions. The user of the reputation system also submits a personal preference profile, which may include preference for *e.g.* cost savings or security with different weights.

The second part (*i.e.* the computation of scores) is taken care of by the DRL algorithm, which is learning from past experiences to create a scoring function. The specific DRL algorithm proposed is Deep Deterministic Policy Gradients (DDPG) [335]. The main advantage of DDPG compared to others is its ability to output approximate continuous actions. Since the application should have a wide range of possible outputs, other DRL algorithms will not be eligible. As shown in Figure 5.2, the algorithm takes environment information as input state, after which the actor inside the agent outputs an action. This action will be random in the beginning, but becomes smarter over time with training, as feedback is received by the critic based on the reward of an output action. This way, the critic continually updates the actor.

Using the personal preference profile of a user, the proposed model generates a personalised reputation scoring function, which the user then uses with blockchain data to calculate scores. While training the model can be computationally intensive, the actual calculation of scores when the model weights are obtained is lightweight. Training the model is initially done using historical blockchain data, after which it is continually updated with usage data.

5.4.3 Filecoin Example

To illustrate how the model could work in decentralised storage markets, Filecoin can be used as an example. Here, the input state to the agent consists of a node's information (who is to be scored), inferred from blockchain transactions, as well as a virtual balance of the user. Examples of blockchain inferred data are average deal time, storage size available, failed deals, and time since joining the network. Meanwhile, a preference profile of the user is required in order to evaluate the action. Initially the model will need to be trained based on historic data pulled from the chain in order to become accurate.

Every input cycle, the DDGP algorithm will output an action (a reputation score from 0 to 100) based on the input states. In this example, the consequence of a bad output could be a loss of profit due to a wrong recommendation, as well as failing to make a deal with an honest counter-party. Evaluating actions is essential to give feedback to the model for it to learn.

After an output score is generated, it is mapped to a *step* function which indicates the probability P of making a deal with the node. P is 1 if the score is over 95, and P is 0 if it is below 30 (following the step function). The accuracy of an action (and therefore the reward) is decided by the distance between P and the user decision D. In the initial training, D is obtained by inferring the preference profile from the node's information, whereas this preference can be obtained from user feedback. When P and D are both below or above 0.5, the reward will be positive, and this reward is increased the closer they are together. The reward is added to the virtual balance which is going to be forwarded to the next state. After enough training the model will learn user preference and optimise this balance.

5.4.4 Discussion

This section has described how the DRL model can be used to create a personalised reputation scoring system. While calculating scores is fairly lightweight when the function has been trained by the model, the actual training phase can be computationally intensive. Therefore, an open question remains how this training would happen in practise. A completely distributed scenario would have every node run its own DRL algorithm, but this might not be feasible for all nodes, especially mobile users, as their devices may be relatively resource and energy constrained. Furthermore, this would create lots of redundant work, as many users will have similar preference profiles and therefore similar functions.

A more feasible architecture would allow nodes in the network to outsource the training of the algorithm, while calculating the scores locally. They would submit their preference profile, fetch the matching function, and calculate the score locally after querying the blockchain for input data. This and performance considerations should be explored further in future work.

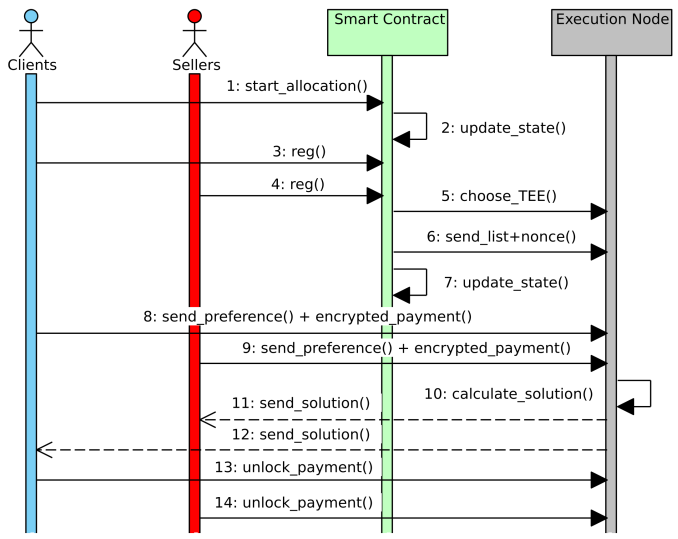

Figure 5.3: Simplified sequence of events in FLOCK. All sellers and clients have been represented in single actors.

5.5 Architecture Design

5.5.1 Overview

In this section FLOCK is presented, an allocation mechanism which satisfies all desirable properties of the allocation triangle of Section 5.3.1 for decentralised allocation of blockchain services. First, an overview of the solution is presented, after which each system component is described in detail.

The proposed solution is comprised of two parts, which together achieve scalability, price discovery, and preference based allocation. In the first part, a smart contract is used to function as a billboard for the allocation, which registers participants and orchestrates the initial phase. Service discovery is taken care of by this contract, as it is public and reachable by all nodes. The complexity of this contract is kept to a minimum to save gas costs, and execution of the allocation algorithm is outsourced to an SGX enclave (execution node), which assures privacy and correctness of execution without expensive techniques such as SMPC or zero-knowledge proofs.

To incentivise proper behaviour of the execution node, a small reward is transferred by nodes in the allocation using off-chain payment networks. This keeps the cost and overhead low, ensuring scalability.

Algorithm 1 Setup (*threshold*)

```
Require: ! $in\_progress$
1: Delete: $clients, sellers, execution\_node$
2: $THRESHOLD \leftarrow threshold$
3: $c\_count, s\_count \leftarrow 0$
4: $in\_progress \leftarrow true$
5: $waiting\_for\_node \leftarrow false$
```

The algorithms used to compute the allocation are instances of stable matching. Partial preference orderings are submitted based on metrics like personal experience, expected QoS, and reputation. This reputation could be inferred from previous on-chain transactions, or using off-chain reputation systems (as described in Section 5.4).

So far, the solution is scalable and allows for preference based allocation, but lacks price discovery. To solve this, the second part of the solution uses an oracle contract, which decouples the price discovery from the allocation computation. This way, market function is outsourced based on macro parameters to the smart contract, rather than doing this on a per item basis, making the process more efficient.

Figure 5.3 shows a sequence of events during allocation. Any node in the network can trigger an allocation, after which interested nodes register either as a client or seller. After a threshold of nodes is reached, the billboard contract picks an execution node and sends them a list of nodes in the allocation among other parameters. All clients and sellers compile a partial preference list of the nodes in the opposite set, and submit this to the execution node along with an encrypted payment promise. The execution node then computes the allocation and returns the solution to the clients and sellers, which in turn unlock their payments.

5.5.2 Billboard Contract

The billboard contract orchestrates the beginning phase of the allocation. First, nodes who are interested in participating in an allocation register at the contract as a client or seller. The contract sets parameters to dictate when the allocation is full; this could for example be a time interval (in terms of blocks) or a pre-set number of nodes allowed to join.

Algorithm 2 Register (x, $caller\_address$)

```
Require: $in\_progress$ & ! $waiting\_for\_node$
Require: ! $already\_registered$
if $x == 0$ & ! $c\_full$ then
    $c \leftarrow c+1$
    $clients.add(caller\_address)$
else if $x == 1$ & ! $s\_full$ then
    $s \leftarrow s+1$
    $sellers.add(caller\_address)$
end if
if $c\_Full$ & $s\_Full$ then
    $waiting\_for\_node \leftarrow true$
end if
```

Algorithm 3 ClaimTask ($caller\_address$)

```
Require: $in\_progress$ & $waiting\_for\_node$
Require: ! $already\_registered$
$execution\_node \leftarrow caller\_address$
$in\_progress \leftarrow false$
```

5.5.3 Integration with SGX

After the registration phase, the smart contract chooses an execution node with an available enclave. This can be an execution node claiming a task, or chosen and contacted by the contract based reputation.

The execution node then obtains a nonce, and a list of node addresses in the allocation. The nonce decides which set in the allocation algorithm will be the initiators for that round. This is not known beforehand so all are incentivised to be truthful in their preferences. Clients and sellers send their preference lists directly to the execution node.

It is assumed that nodes are connected to a payment network, and they can send off-chain payments without added cost. The node in the allocation sends the execution node an encrypted payment receipt, which the latter will be able to unlock upon completion.

Next, the enclave runs the algorithm to produce an allocation. This algorithm will be publicly available (e.g. stored on decentralised storage) and should be remotely attested by a subset of nodes to verify correct instantiation. Finally, the enclave sends the results back to the nodes, after which the secret is released and their balance is updated.

The approach described above differs to the centralised approach of resource allocation in a number of ways. Most importantly, the function of the sellers, smart contract, and execution node are all concentrated to a single entity.

5.5.4 SGX Security Analysis

The disadvantages associated with SGX are not a problem for FLOCK. As opposed to auctions, there is little to be gained from attacks on the allocation, as there is no money directly involved. SGX specific attacks and vulnerabilities are now discussed, and specifically how they are unlikely to affect FLOCK.

Side-channel attacks aim to infer information from inside an enclave, but in the case of FLOCK the costs of an attack is much higher than potential gain, as price discovery is decoupled and therefore there should be indifference over allocated nodes. Single-point attacks may be launched as a general attack against a service, aiming to exploit the single point of failure of the enclave. This requires considerable resources, and the smart contract has a number of backup nodes if an enclave fails, and could revert to computing the allocation on-chain.

Attacks from the SGX platform operator are incentivised against as a gas fee is paid to claim the task, which is recovered from off-chain payments. If it acts maliciously it will not recover these funds. Similarly, availability failures should be rare as the platform owner wants to keep the enclave running.

Finally, there is limited protected memory on SGX of 128MB, which poses a constraint on the computation. The required memory is mainly for storing addresses and preference lists. Ethereum addresses consist of 42 hexadecimal characters, which take up 21 bytes. Assuming an average preference list size of 5 per node, the enclave needs to store 6 addresses per node. This means that per node 126 bytes need to be stored, allowing to fit 1,015,873 addresses within the memory limit. This does not take into account storage of the allocation algorithm or nonce, but as not even 10% of this amount is not expected (due to the desired quick timeline of allocations), there should be plenty of space left. Therefore, the memory limit should not be a problem for the computation in FLOCK.

5.5.5 Oracle Contract

The oracle contract is responsible for a global price discovery. The contract derives its state (i.e. the price to be paid for a service) from the underlying billboard contract and other public parameters. The oracle contract can be deployed for different services and incorporate complex financial rules dictating the price. The simple contract used in Section 5.6.1 uses the ratio of clients to sellers and the change of the price of Ethereum to set a price.

The oracle contract updates its price when called by a node. Its pricing mechanism and the parameters from which the price is derived are transparent. Therefore, any node can see if there is a discrepancy between the current price and what it should be, and can call the function to update. This is financially incentivised when the price difference will cause more gain than the cost of calling the contract.

The work in this chapter is aimed at fast allocation of large numbers of nodes, and for this use-case global pricing is an efficient solution. However, it must be noted that there is still a need for other mechanisms like auctions for popular items. A secondary market can be envisioned based on multi-item auctions for the top sellers and others who want to sell spare capacity from the primary market. This can be compared to the spot market approach from Amazon, where flat and on-demand instances of services are sold on the primary market based on a global price, after which spare capacity is offered based on bids on the secondary market, usually at a lower price.

5.6 Evaluation

To evaluate FLOCK its key components have been implemented[3]. On these, extensive simulations have been performed in order to assess latency, cost, scalability and more. Specifically, the smart contracts used are examined in terms of gas costs, and the proposed allocation algorithm is compared to other initiatives. The desired properties of scalability, preference based, and price discovery are analysed by implementation of the smart contracts, as well as the matching algorithms. The

[3]`https://github.com/navinkeizer/FLOCK`

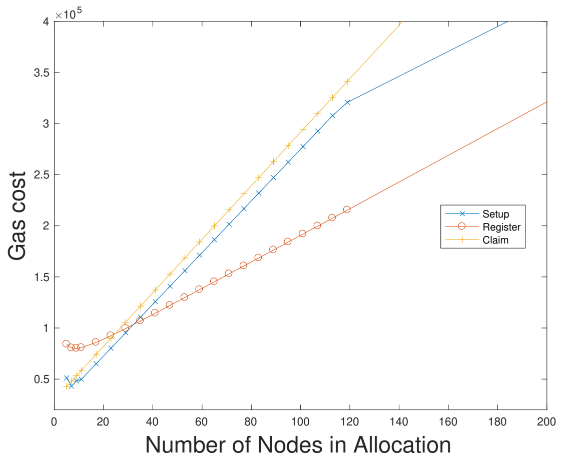

Figure 5.4: Progression of gas costs (GWEI) for functions of the billboard contract using on-chain storage of nodes.

simulations were performed on a local laptop with average specifications.

5.6.1 Smart Contract Cost

The billboard and oracle contracts have been implemented in Solidity, and several simulations have been performed to explore their gas costs. Remix IDE is used as the platform for implementation, along with Ganache virtual blockchain and Ethereum testnets.

Starting with the billboard contract, its main functions are tested: Setup, Register, and Claim. For completeness their pseudo-code has been added. The Setup function (algorithm 1) is used to reset the allocation state and start a new allocation with a new Threshold. Nodes can use Register (algorithm 2) as long as they have not already registered as client or seller, which is checked with an internal function. The number of nodes in either set are registered and updated using count variables. Finally, an execution node can use Claim (algorithm 3) if it is not participating in the allocation. Its gas fee can be seen as collateral which it will regain after successful computation.

Figure 5.4 shows the progression of the gas costs as more nodes participate in the allocation. A clear linear increase of the gas costs can be observed, which quickly become unmanageable. This is due to the simplicity of implementation using on-chain storage for the intermediate states (the list of nodes participating in

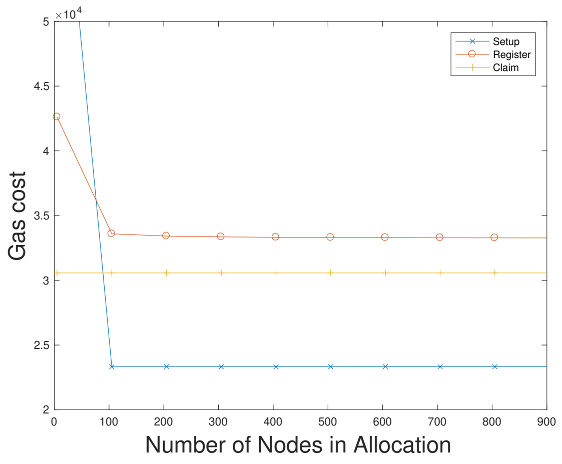

Figure 5.5: Progression of gas costs (GWEI) for functions of the billboard contract using off-chain storage of nodes.

the allocation), and therefore this simple proof-of-concept implementation lacks in performance.

An off-chain storage solution is needed for the contracts, which is discussed further in Section 5.7.1. The billboard contract has been implemented with this assumption of off-chain storage available, and in Figure 5.5 it can be seen that in this case, gas costs are not only an order of magnitude lower (note that the y-axis differs by 10^1), but also remain nearly constant. There will be some interaction needed between the smart contract and the off-chain component, and therefore Figure 5.5 can be taken as a lower bound on gas costs, whilst Figure 5.4 is the absolute upper bound.

Table 5.1 shows how the proposed on-chain solution for a 5 node setup outperforms implementations of PASTRAMI and Trustee, for all actors associated with the allocation/auction. For the on-chain implementation, the gas costs paid by the execution node is assumed to be fully refunded with an added premium by nodes in the allocation through the payment network, and this is reflected in the gas costs. Furthermore, PASTRAMI has an added cost of up to 131,804 in case of misbehaviour.

Table 5.2 shows the gas costs of the oracle contract, along with the conversion

	On-chain	PASTRAMI	Trustee
client	143,827	391,046	1,847,178
seller	92,673	95,934	143,720
execution node	-	7,108	-

Table 5.1: Comparison of individual gas costs (GWEI) of allocations/auctions on blockchain (5 node setup).

Function	Gas Cost	Cost (USD)
deploy contract	213008	$2.8168
setup oracle	71360	$0.9437
update oracle	56931	$0.7529

Table 5.2: Gas costs (GWEI) and USD conversions of oracle contract functions.

to USD[4]. It should be noted that the deployment and setup of the contract are one time costs, after which the update cost is paid by those who to call it, when they notice a price discrepancy. Nodes are incentivised to call this function when their gain of a new price exceeds $0.7529.

With the introduction of layer 2 scalability solutions, as well as the transition to Ethereum 2.0, gas costs are expected to decrease. In fact, gas cost have fallen by about 48 % since the time of performing this analysis. However, at the same time the price of ETH has tripled in the same period, causing the total cost to increase overall. While prices of gas and ETH continue to fluctuate, they will stay in a cost range which is considered manageable for participation in FLOCK.

5.6.2 Allocation Algorithm Performance

To verify the scalability of FLOCK, it is next shown that the proposed allocation algorithm executes with low latency, and achieves a significant improvement over other algorithms that could be used. The focus here is on the second type of services from Section 5.3.2 and this is modelled using an implementation of the hospital/resident (HR) stable matching algorithm[5].

For a comparison with auctions, the VDA implementation from PASTRAMI is used, as well as a simple implementation of a Vickrey auction, which does not

[4]based on the average gas price and Ethereum price on 08-11-2020
[5]https://github.com/daffidwilde/matching

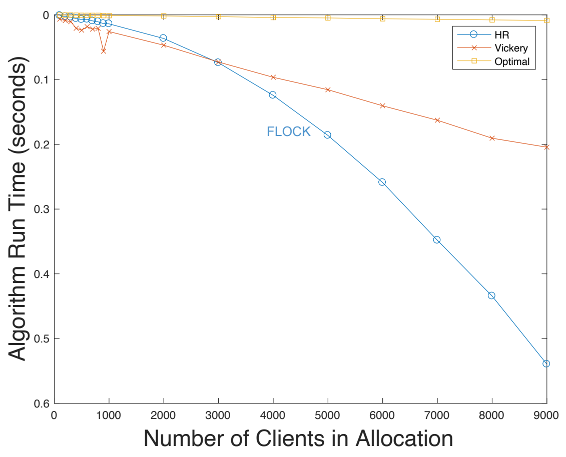

Figure 5.6: Single-item allocation algorithm comparison.

implement the security functions needed for sealed bids. Implementations of VCG auctions were also considered, but as they were significantly slower they have been omitted from the results. Finally, a simple mechanism was implemented to allocate at random without any constraints, which represent an optimally scalable solution (to be used as baseline). Secure enclave overheads have not been considered as these would be similar for all solutions.

First, the algorithm run times are compared for a constant number of sellers (100) and increasing numbers of clients. The assumption is made that there will be more clients of services than sellers, as there are barriers to becoming a seller such as spare capacity available and system requirements.

Figure 5.6 shows the run time for varying clients for the different algorithms in the single-item case —that is, where all sellers only have one item to sell. Note that the y-axis is inverted and therefore a higher line translates to a lower delay. It is evident that until about 3,000 clients, the HR algorithm performs better than the Vickrey. It is important to note that the strength of the HR algorithm lies in allocating large amounts of nodes at the same time, which is not exploited in the single item case.

Therefore, Figure 5.7 is more relevant, as it shows the same simulation for the multi-item case (*i.e.* all sellers can have capacities higher than 1). The HR algorithm clearly outperforms the VDA and remains close to optimal, staying under a second

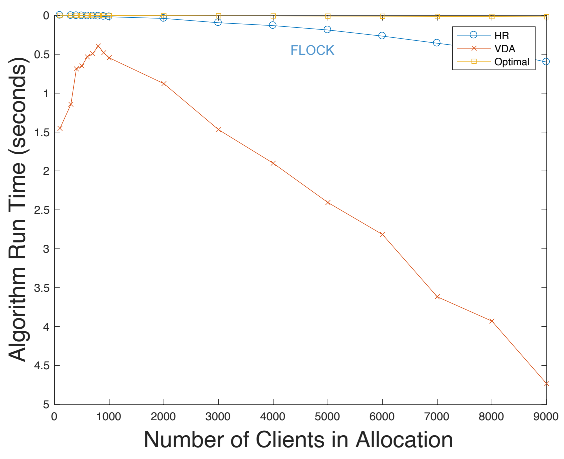

Figure 5.7: Multi-item allocation algorithm comparison (maximum capacity = 3).

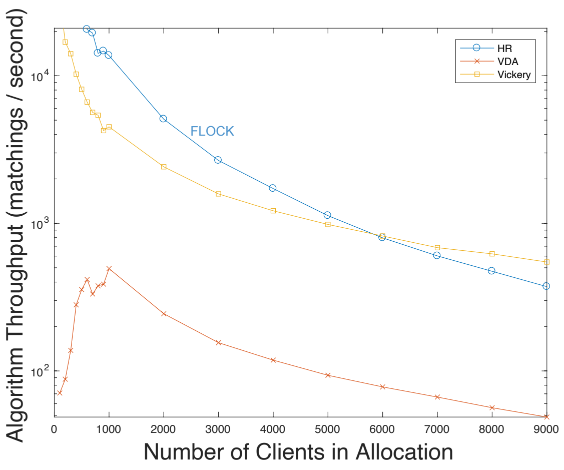

Figure 5.8: Multi-item allocation throughput comparison (maximum capacity = 3).

of run-time when approaching 10,000 clients.

So far, the HR algorithm is much faster than the VDA algorithm. Looking solely at latency however, gives a skewed image in favour of the Vickrey auction (as seen in the single-item case). To inspect this comparison further, the throughput in matching per second is explored next.

Figures 5.8 and 5.9 show the throughput of the evaluated algorithms for increasing numbers of clients. The maximum capacity in terms of items to sell per seller has been varied, as this illustrates the strength of the HR algorithm. Evidently, the HR algorithm outperforms the VDA algorithm, as well as the simple Vickrey auction up until a certain limit. This limit moves from 6,000 clients for a maximum

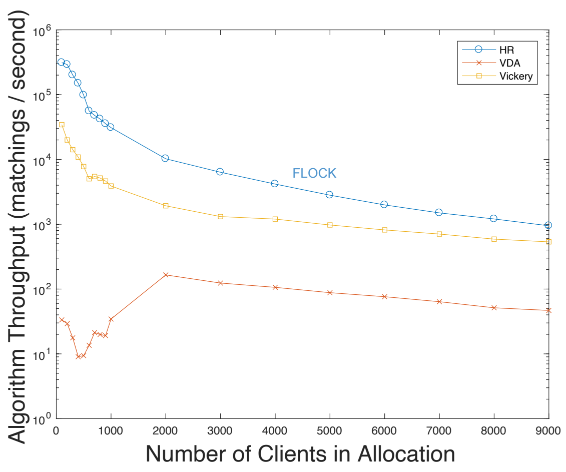

Figure 5.9: Multi-item allocation throughput comparison (maximum capacity = 10).

capacity of 3, to over 10,000 clients for a maximum capacity of 10. This illustrates the scalability of the HR algorithm.

Furthermore, it is important to note that in these simulations the simple Vickrey auction only matches one client per seller rather than within a capacity like the HR, which results in a much lower percentage of clients being matched. A simple solution could be to duplicate the bids for any of the instances of a node, but this fails in practise as a client may only require a single instance or would lower their prices for a subsequent instance as its utility for it is lower. Practical implementation would require more sophisticated bids like the VDA, which has been shown to perform suboptimal compared to the HR algorithm.

Additionally, Figure 5.10 confirms that if the sellers are increased for the Vickrey auction, as to make for a fair comparison with the HR algorithm, its performance lacks behind. This is shown by comparing its runtime of 0.12 seconds to match about 150 sellers to 500 clients, to the runtime of the HR in Figure 5.7, when there are 100 sellers with on average 1.5 items matched to 500 clients, of 0.0066 seconds.

Figure 5.11 shows the increase of the algorithm runtime as the preference list size is increased. The figure shows a linear increase, confirming that small preference lists are preferred for faster execution, which is in line with users not being able to submit full preference lists.

Finally, although this cannot be shown in simulation, it must be noted that the

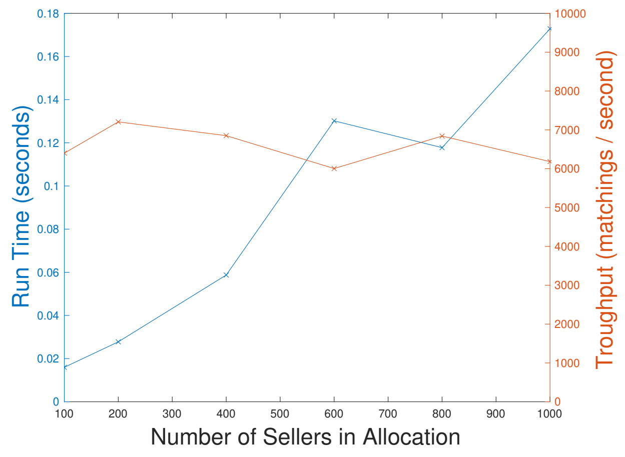

Figure 5.10: Performance of Vickrey auction (clients = 500, maximum capacity = 5).

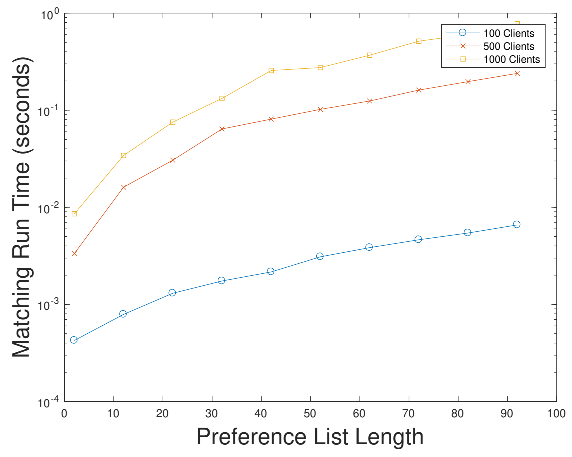

Figure 5.11: Influence of preference list size on performance (for hospital/resident algorithm, maximum capacity = 3).

preparation phase for the auctions are much longer, especially when adding more complex bids. For the HR allocation, the preparation phase can be automated, where all nodes have a list of trusted nodes, from which it submits those that are available to the allocation. Furthermore, as mentioned before, a very primitive version of the Vickrey auction is used in the comparison, which does not implement the security features needed for sealed bid auctions, and therefore its performance can be seen as an ideal case.

5.7 Discussion

This chapter has presented a decentralised service allocation system on blockchain, which supports lightweight and scalable pairings of clients and sellers. The importance of capturing all sides of the allocation triangle was introduced and highlighted, in order for decentralised services to compete with centralised counterparts. The proposed solution captures these aspects, but is by no means the only possible solution. Possible extensions to this work are now discussed, which could improve on its limitations and performance.

5.7.1 Off-Chain Storage for Intermediate States

The simple billboard contract implementation is meant as a proof-of-concept. However, as seen in the evaluation, on-chain storage is not feasible and quickly renders the system unusable. Instead, while collecting the nodes in an allocation round, this intermediate data should be outsourced off-chain. For example, decentralised storage (IPFS, Storj) can be used to temporarily store the user list, as long as retrieval of these nodes remains quick. Another method could be to use another dedicated node with SGX capabilities as temporary storage node. Implementation of this is left to future work.

5.7.2 Decoupled Service Discovery

To further decrease the cost and delays associated with an allocation, a completely off-chain solution may be needed. Smart contracts are used as their public reachability provides an inherent service discovery. This is at a cost, which can be significant if it needs to be repeatedly paid by users. Therefore, a solution may decouple the service discovery completely, and use other mechanism to query the P2P network and find allocations orchestrated in a decentralised manner. Service discovery could also take into account the distance of nodes, and thereby minimise networks delays between allocated nodes.

5.7.3 Real-Time Allocation

Although the performance of the implementation meets scalability and speed constraints, certain applications may need near real-time allocation. The delays including gathering preferences, blockchain consensus, and algorithm execution might be too long. An example of this is content retrieval markets, where nodes requesting some content should not have to wait for the allocation to be ran upon every request.

In this case, the performance of the system can be increased by performing the allocation beforehand, rather than as requests come in. The allocation now is between the content provider and the retrieval miner (seller). This can be done periodically, assigning the task of retrieval of some content to a node. Other nodes can request the content from this node, and payment by clients is still based on the oracle contract. In practise this is an easy extension to the proposed implementation.

5.7.4 Oracle Contract Extensions

In this work, a simple oracle contract was described. This concept can be extended much further. As a start, each decentralised service that is offered may have its own oracle dictating its price. One step further could define different prices for different nodes in the same service, based on parameters such as QoS, reputation (*i.e.* Section 5.4), and location.

The composition of the contract is left to developers. The implementation in this work uses the most basic market dynamics, and lacks financial sophistication. The oracle can be made arbitrarily complex however, and its design may be based on financial incentives and game theory.

Finally, as the complexity of the oracle is increased, its cost will increase too. A solution to this could be to outsource its computation to a secure enclave, similarly to how the allocation is computed. This would however decrease the decentralisation and security of the solution, and would require more complete security analyses as there might be attack vectors with monetary gain, as in this computation there is an important monetary aspect.

5.8 Summary

This chapter presented FLOCK, a decentralised service allocation system on blockchain, which supports lightweight and scalable pairings of clients to sellers. This solution leverages smart contracts, stable matching, payment networks, and

TEEs to achieve the properties of the allocation triangle. General allocation types are distinguished, and use-cases in decentralised storage and retrieval markets are explored.

Evaluation of the system shows that cost and overhead of FLOCK are manageable and much lower compared to systems using auctions, reducing cost by at least 52%. Furthermore, using a hospital/resident stable matching algorithm greatly improves performance over auction algorithms in terms of latency and throughput, staying under 0.5 seconds for 9000 clients. Finally, additional ways to improve on FLOCK's performance and limitations are discussed.

Chapter 6

Decentralised Mechanisms for P2P Networking: NAT Traversal

6.1 Overview

Traversing Network Address Translators (NAT) remains a big issue in P2P networks, which are the basis of DWeb services. Previous attempts to solve this issue do not function without trusted centralised servers. This chapter presents a decentralised, relay-based NAT traversal system, where any reachable node is able to assist an unreachable node in NAT traversal. Smart contracts on the Ethereum blockchain are used to ensure fair rewards. Besides financial incentives, a reputation system based on transactions on-chain is used to mitigate against malicious behaviour, and guide peer discovery.

Evaluation of the proposed system peer discovery and reputation system shows that a combination of historic performance metrics leads to an optimal scoring function, that the system takes little time to reach stability from inception, and that the system is resilient against various attacks. Implementation of the smart contract shows that the cost for participants is manageable.

6.1.1 Introduction

P2P networks have been popular in the design of various DWeb applications due to their decentralised nature and associated benefits. There are however a number of challenges and issues in P2P networking. One main challenge stems from the incompatibility with the current Internet infrastructure, which is tailored towards the client-server model. As most DWeb initiatives like blockchains and DSNs are built on top of P2P networks, they suffer from these same issues.

One of the most prominent is the issue of unreachable peers, which can be caused by NATs, firewalls, VPN's and more (from here these are all generalised as NATs). In a P2P network, a node must be able to act as a client and a server at the same time. However, NATs make it impossible to receive requests from the network if the node itself has not already set up a connection with the other node.

In the early days of P2P networks research focused on fixing this problem of NAT traversal [336, 337, 338, 339, 340], and this produced a number of protocols such as STUN, TURN, hole punching, connection reversal, port number prediction and more. The reason a large suite of protocols originated is the lack of standardisation in NATs, making protocols only suitable for a subset of NATs.

One of the main issues with the proposed solutions is the need for a set of centralised servers assisting in the network by relaying messages or maintaining connections with various nodes. These servers are prone to various attacks [339], require complete trust in the central infrastructure, and can be situated far from the node using its services, thus incurring high latencies. While this is acceptable for certain applications, many latency-sensitive applications would incur significant performance degradation, *e.g.* P2P video calls, P2P live video streaming (such as in P2P gaming), blockchain based computation offloading for latency sensitive tasks, and block propagation in a blockchain network.

This chapter proposes a decentralised, relay-based NAT traversal system for P2P networks with a built-in reward system for relays who assist nodes in their communication. In this system, any reachable node in the P2P network can become a relay, which requires incentivisation of honest behaviour and stronger security measures, as these nodes are untrusted and potentially malicious.

The proposed system is realised by a combination of *i)* dual-path routing, producing a *Proof of Timely Relay* (Section 6.4.1), which ensures a client of honest behaviour of relay nodes, and *ii)* smart contracts (Section 6.4.2) on the Ethereum blockchain, which ensure fair exchange for the work done by relays. Nodes are incentivised to behave properly in order to claim maximum rewards.

A reputation system (Section 6.4.3) is incorporated on the blockchain to protect nodes from potential malicious nodes, and increase the chance of being used as a relay when behaving properly. The *reputation scoring policy* is determined locally based on historical transactions on the blockchain (as a simplified version of the idea presented in Section 5.4), and assists in the *peer discovery* process (Section 6.4.4).

The evaluation (Section 6.5) explores different reputation scoring policies and their ability to avoid contracting malicious relays, yielding a low malicious peer pick ratio of 5% using two rounds of peer discovery for up to 30% of malicious nodes in the network. Furthermore, it shows that the reputation system is able to quickly stabilise from inception (within 5-10 cycles). Finally, the cost overhead and performance of the smart contract implementation is tested, in order to assess performance and feasibility of the approach.

The main contributions of this chapter are as follows:

- A reward system for relays assisting in decentralised relay-based NAT traversal, incorporated using smart contracts. Incentives are used to promote honest behaviour, and rewards are based on the relay performance.
- A node-centric, open, on-chain reputation system, where nodes locally decide a scoring policy, and which is used to aid in peer discovery.

6.1.2 Related Works

There are a number of works which implement NAT traversal solutions in a way which is highly related to this chapter. Several works propose rewards for relaying connections and files. Ghosh et al. [88] propose the use of TorCoins to reward users offering bandwidth for relaying of TOR connections. A proof of bandwidth is used to verify that the bandwidth claimed for rewards is the correct bandwidth transferred, and contributors in a circuit claim rewards proportionally to the end-to-end QoS measured over the whole path. Goyal et al. [341] introduce a system for decentralised content delivery incorporating secure incentivisation in P2P networks.

This system focuses on content providers, who start smart contracts for a file they want to distribute, and pay relay peers for content delivery, based on a proof of delivery chain.

Norton and Simanavicius [342] propose a smart contract based segment routing WAN system, based on a centralised controller. Users provide or consume spare bandwidth, forming a network of segment routers on user machines, used to find less congested network paths. Rewards are claimed through smart contracts, which are updated by tickets submitted to the centralised controller. The work in this chapter is distinct from these approaches as a mechanism is implemented for rewarding based on QoS measured, combined with a reputation system to prevent contracting malicious peers in peer discovery, as well as smart contract micro-payments to ensure fair exchange.

Jiang et al. [343] present a hybrid architecture for VoIP calls using either default paths or a managed overlay solution, comprised of relays in data-centres managed by a centralised controller. This improves performance and QoE by deciding relay paths based on prediction guided exploration. The work in this chapter could extend this to public relays, exchanging spare resources for rewards with distributed control.

6.2 Preliminaries: NAT Traversal

This section presents background on the issue of NAT traversal and mentions solutions from early P2P literature. NATs perform both port and address translation at the boundaries of public and private networks [336], enabling a much larger number of nodes in the private networks to be connected to the public Internet through a single public IP address. Public ports are assigned to specific private IP/port combinations when connections are going outward, leading to connectivity issues when an outside peer wants to connect to a node behind a NAT in a P2P network [338]. The additional public address space introduced by IPv6 will most likely not solve this problem, as NATs will be used to communicate between the two versions, and firewalls and VPN's will continue to be present. Therefore, it is essential for P2P networks to succeed that NAT traversal is taken into careful consideration.

There have been numerous proposed protocols to aid in traversing NATs. In general, there are two cases when NATs create issues for P2P networks for communication between two nodes: *i)* both peers are unreachable (behind a NAT), and *ii)* only one of the two peers is unreachable. In the first case, connection in both directions will fail, whereas in the second case only the connection from the public node to the unreachable node will fail.

For the first case, STUN and TURN are the most notable solutions. STUN uses a set of central servers to help peers behind a NAT with discovering their perceived address from the public Internet. TURN leverages a set of centralised servers to relay entire connections and transmissions to the unreachable node. Although the performance of TURN is lower comparatively, the reliability is high [338].

For the second case, connection reversal is most reliable, as it uses a centralised server to initiate the connection to the unreachable node, after which a direct connection is set up. All the above protocols besides TURN have specific requirements in order to succeed.

The centralised approaches described above have a number of issues associated with them. From a performance perspective, the latencies are not acceptable for certain real-time and latency sensitive applications. Furthermore, there are a number of well-known security threats and attacks, such as the man-in-the-middle attack, where a malicious node can insert itself in the path of all packets going to the victim [338, 339]. By pretending to be the NAT to the client and vice versa, the malicious node can in this case drop packets, alter content, delay packets, and learn about network topology and other vulnerabilities of the victim. There is also communication hijacking, denial of service attacks, and partitioning attacks which can isolate and overflow the victim.

Besides these centralised approaches, Libp2p [344] has proposed decentralised versions of STUN and TURN, where any peer in the network can replace the centralised server. This approach, however, lacks peer discovery and bootstrapping. More importantly, the Libp2p implementation lacks required security measures, which are essential since any peer can provide this service, and is intrinsically untrusted. Launching an attack as an adversary in the position of network assisting (relay) node becomes trivial without incentivisation of proper behaviour and identification of malicious nodes.

6.3 Threat Model and Assumptions

The P2P network is assumed to be divided into two sets of nodes: reachable and unreachable peers. Reachable peers can act as relays, whereas unreachable peers need a relay to participate in the network. Reachable peers are peers that are able to function as a client and server at the same time.

In the system, unreachable peers act as clients and reachable peers can decide to act as relays. Both are assumed to be potentially malicious actors, but rational. In this case, a malicious actor is one who may exploit the system for monetary gains and information, but also a node which oversubscribes its limited (bandwidth) resources and is thus unable to provide the good relay service promised. The latter is assumed to be more common, as there are no direct financial incentives for launching an attack, but there are for serving more nodes as a relay (oversubscription attack).

Because the clients and relays mutually distrust one another, smart contracts are deployed on a blockchain to behave as a trusted, mediating third-party. One disadvantage of the smart contract approach is the requirement of network synchrony, and constant blockchain monitoring, for example to avoid a timeout event as an honest node. All peers are assumed to be connected to the Ethereum blockchain (based on previous relays or as light nodes). Furthermore, the majority of network links are assumed to be stable and thus this should not add large additional latency.

The proposed system aims to present a secure NAT traversal mechanism for P2P networks, where malicious attacks are irrational and malicious nodes are easily identified, while ensuring honest nodes a fair exchange of rewards for their resources. These goals are achieved through incorporating the following:

- Financial incentives through smart contracts to stimulate good behaviour and mitigate against malicious attacks.

- Reputation system on-chain to identify potential malicious nodes and predict the performance of a node.

- Peer discovery based on nearby peers, their reputation score (computed locally based on transaction history on-chain), and possibly a blind auction to receive the highest bids (from client perspective in terms of promised minimum QoS).

Besides security, the system should have an increased performance in the best case (when an honest nearby node is contracted) over centralised solutions. The system and smart contract should not add significant overhead (processing, network traffic, and cost) compared to the centralised solution either.

In terms of the actual NAT traffic relaying, this work builds on current NAT traversal techniques (*e.g.* Libp2p) and adds a layer on top of them to facilitate decentralised NAT traversal. Because of this, the rest of this chapter focuses on the latter, and leaves NAT traversal details out of the scope.

6.4 Architecture Design

The goals mentioned in the previous section are reached through a combination of Proof of Timely Relay (PoTR), which guarantees the unreachable peer that data has been relayed correctly and timely, and a smart contract which governs the rewards, ensures fair exchange, and implements a reputation system through mutual scoring.

The unreachable client nodes contact a number of relays to help become reachable in the network. This agreement is stored on the smart contract on the blockchain, and includes parameters such as the contract duration, QoS promised by the relay, and payment rate.

Depending on the situation, the relay node either uses connection reversal (in the case the peer dialling the unreachable peer is reachable in the network) or sets up a circuit relay if both parties are unreachable. This work focuses mostly on the latter since it has a higher reliability, and since there is no prior knowledge of

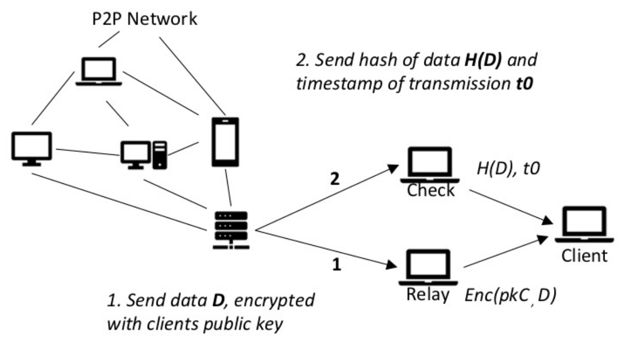
P2P Network
2. Send hash of data H(D) and timestamp of transmission t0
2
1
Check
H(D), t0
Client
Relay
Enc(pkC, D)
1. Send data D, encrypted with clients public key

Figure 6.1: Proof of Timely Relay (PoTR).

network topology, the worst case should be assumed. The mechanism is similar to that in the centralised case, but instead of using the central server, the peer acts as the intermediary.

6.4.1 Proof of Timely Relay

Proof of timely relay (PoTR), as shown in figure 6.1, is the mechanism which ensures the unreachable client node that the relay they have contracted is not acting maliciously. It allows the client communicating with a peer to verify that the data has been relayed without any additional delay higher than promised in the contract, without the data being altered, and without a large packet drop rate.

In order for the client node c to have this verification, they pick two relay nodes in the P2P network: one as a *check node (R_c)* and one as the *"true" relay node (R_r)*. R_r acts as a standard relay, where a peer p (who is communicating with the relay's client) would send its data to R_r (using end-to-end encryption to prevent tampering) which the relay node R_r in turn forwards it to c. At the same time, the peer p sends a hash of the data and timestamp of the start of transmission of the file to the check node R_c, which in turn forwards it to c. This information then allows c to assess the performance of R_r.

The client and the relay nodes both keep a record of their view of the performance of R_r in their local performance log, based on PoTR for c and based on forwarding delays and approximate network delays for R_r. Every evaluation inter-

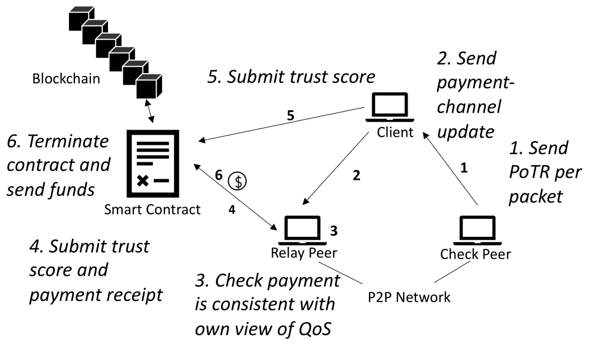
Blockchain
5. Submit trust score
5
2. Send payment-channel update
Client
6. Terminate contract and send funds
1. Send PoTR per packet
1
2
6
$
4
Smart Contract
3
Relay Peer
Check Peer
4. Submit trust score and payment receipt
3. Check payment is consistent with own view of QoS
P2P Network

Figure 6.2: Relay operation and smart contract termination.

val (6.4.2), both parties calculate the average performance over the interval they perceive and use this to settle payments.

6.4.2 Smart Contract

The smart contract is used to document the agreement between unreachable clients and relays on the blockchain, and govern payments by c for the services provided by R_r without the need of a trusted third party. The smart contracts allows for conditional payments based on the QoS provided by the R_r, and allows for micro-payments ensuring fair exchange.

The proposed smart contract is implemented in Solidity and deployed on the Ethereum blockchain. When a client starts a new agreement with a relay, it submits the following negotiated parameters to the blockchain:

1. Minimum QoS score promise, as any statistic of:
   - Latency
   - Packet drop rate
   - Throughput
2. Evaluation interval time
3. Deposit
4. Payment rate function parameters

5. Timeout duration

The minimum QoS score promise is the lowest QoS score that is acceptable for c, and for which it is willing to reward R_r. The QoS score is based on performance metrics such as latency, drop rate, and throughput. This composition is envisioned to be different for different applications, and the implementation is left to application developers.

The evaluation interval represents the interval when c looks into its performance log and compares the performance in this interval to what was promised (minimum) by R_r. If it is insufficient, it will terminate the agreement. If the QoS has been satisfactory, it sends a signed payment update to R_r, according to the QoS score and the payment rate function. The interval can be set larger to avoid the overhead of calculating scores frequently, but there is a trade-off with security as smaller intervals allow malicious behaviour on either side to be identified early.

Upon receiving the payment update, R_r compares this to what it expects, based on its own observed performance, to check c is not underpaying. If it notices malicious behaviour from c or has other reasons to terminate the agreement (such as oversubscription) it can close the payment channel any time by submitting the latest signed payment channel update to the smart contract and triggering the termination process.

A client submits a deposit which will act as their relay credit, forming a pay-as-you-go structure. The deposit will set the duration of the contract, based on the maximum payment rate. A contract can be extended by additional deposits.

The termination process can be triggered by c or R_r to end the agreement and settle the payment. First, the node sends a termination message to the other, submits a trust score to the smart contract of the perceived trustworthiness of the other peer (explained in Section 6.4.3) and in the case of R_r also the payment receipt. Next, the other node submits the trust score they observed to the contract (and payment receipt if R_r), after which the payment is made to R_r and the rest of the deposit is returned to the c. The smart contract will only pay the funds when both scores have been received by the contract, and thus incentivises a terminating node to notify

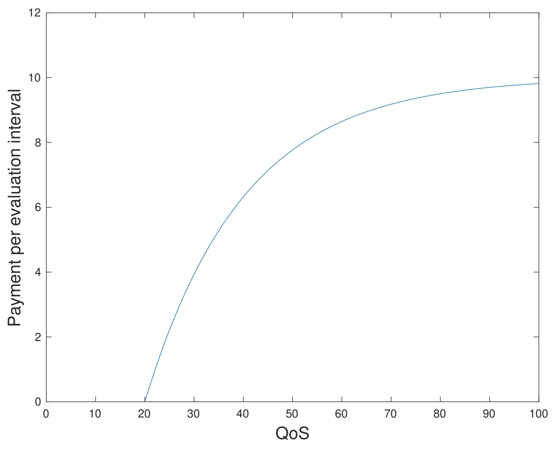

Figure 6.3: Example of payment function with p_{max} = 10 and QoS_{min} = 20.

the other node, and it incentivises both nodes to participate in the scoring. If one node becomes unresponsive and holds the other node hostage, a timeout event can be called if the duration has expired to give the calling node their payment and automatically assigns a negative trust score to the unresponsive node.

The *payment rate function* is another parameter which is negotiated between c and R_r, and it describes the payment rate for different levels of QoS score. This incentivises R_r to perform better to receive better pay, and should be chosen to deter oversubscription attacks. One example of this could use a negative exponential function, with a minimum promised QoS score acting as a lower bound for receiving rewards. A maximum payment bound is also introduced to prevent infinite rewards, both dictated by supply and demand. To simplify, the QoS score is quantified as a number from 0-100, which in reality will be based on the latency, throughput, and drop rate (and may be normalised to this range). Using these constraints, this function could be formed as follows:

$$P = p_{max} * (1 - e^{-\frac{1}{20} * (Q - QoS_{min})}) \tag{6.1}$$

Where P is the payment per interval, p_{max} is the maximum payment rate, Q is the QoS score over the interval, and QoS_{min} is the minimum QoS score promised. Figure 6.3 illustrates this payment rate function with p_{max} set at 10, and QoS_{min} at 20. As a QoS of 20 is reached, there are increasing rewards until the negotiated

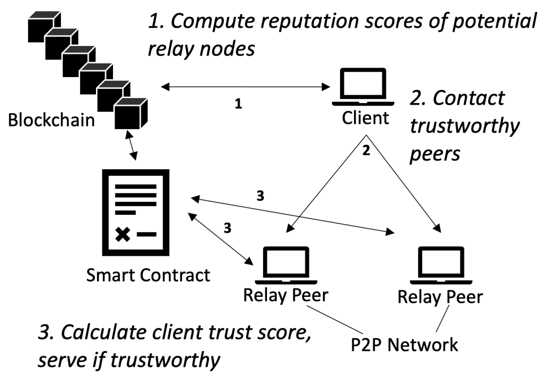
1. Compute reputation scores of potential relay nodes
Blockchain
1
Client
2. Contact trustworthy peers
2
3
3
Smart Contract
Relay Peer
Relay Peer
P2P Network
3. Calculate client trust score, serve if trustworthy

Figure 6.4: Peer discovery based on on-chain transactions.

level of 10.

6.4.3 Reputation System

The smart contract records the trust scores given to the peers involved in the contract for each agreement. As the blockchain is open and anyone can query the ledger to get the previous transaction data including the trust scores, this makes for an open, decentralised and node-centric reputation system. Nodes can locally decide their own policy for calculating reputation scores of nodes in the network (similar to the idea proposed in Section 5.4). This can be a pure indication of trust scores assigned in history, or go further and incorporate metrics such as contract duration and promised QoS scores. They can also opt to use public information only, or combine it with private information based on its own previous agreements.

The trust scores given to each node is either positive (i.e. 1), or negative (i.e. 0). Re-join attacks are avoided by setting negative scores to 0, as now nodes will not gain from rejoining the network after receiving a negative score, because they will still start at 0 (the same as before).

6.4.4 Peer Discovery

In order for an unreachable client to discover potential relays, both the reputation of nodes and the expected performance are important. Malicious nodes should be avoided in general, even though they can easily be picked out in operation through PoTR and eliminated through the smart contract, as they may still damage honest nodes. First, the time spent initiating a contract, waiting for an evaluation interval, and terminating a contract will delay the transmissions of files in the network and will reduce the overall QoS. Second, instantiating the smart contract and subsequent functions on it requires gas, which is paid for in Ether, leading to a monetary loss. Last, packets might be lost due to the confusion, and this may affect the clients perceived trust by the network.

Contracting R_r with close proximity to c will likely lead to a better expected performance. Furthermore, the R_c would need to be close to ensure timely delivery of the PoTR. For this reason, peer discovery is proposed as follows.

First, c finds nodes providing relaying services nearby. There are various proposed ways this can be done, such as flooding the network and waiting for the first response, registering nodes at servers based on location, and using application level anycast [345], although this raises the question if globally reachable nodes are required. Further details of this is left for future work. After obtaining the set of nearby nodes, c calculates their reputation scores, in order to filter out potential malicious nodes. Calculating only the scores of these nodes rather than all nodes in the network increases performance and decreases overhead.

In the next part, the top N peers are contacted. If this number is low, c could initiate a direct relaying request to the peer(s) and negotiate the parameters. If N is large, c can set up a blind auction (possibly on-chain [317]) to receive bids for the highest QoS/Payment function parameters. The contacted peers will locally calculate the trust score of the client to decide whether to participate or not. A contract is set up with the highest bidder by the client, and if all parameters are correct R_r starts relaying. Implementation and evaluation of this is left for future work.

6.5 Evaluation

Different aspects of the proposed system are now evaluated by running simulations, and this shows how the design goals outlined in Section 6.3 are met. The simulations were performed on a local laptop with average specifications.

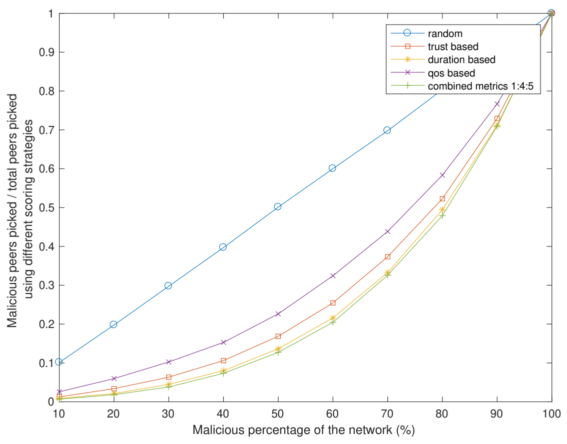

Figure 6.5: Malicious node pick ratio for different scoring functions (against malicious percentage in network).

6.5.1 Peer discovery and Reputation System

In order to evaluate the performance of peer discovery using reputation metrics the PeerSim P2P simulator [346] in Java was used. A network of 100,000 nodes was created, all with an ID and boolean variable to indicate maliciousness, thus dividing the network into a set of malicious and honest nodes. All nodes can act both as a relay and a client. Historic transactions are next generated on the blockchain including the number of previous transactions, trust, duration, and QoS scores associated with peer identities based on a Zipf distribution with different exponents for malicious and honest peers.

The assumption is made that most malicious peers will start with a lower number of historical transactions, because increase in usage would lead nodes to be more invested into the system due to the built up reputation. Only a minority of malicious nodes will have a larger number of historic transactions (such as oversubscribed honest peers), and this is modelled by setting the number of transactions using a higher exponent Zipf distribution for malicious nodes compared to honest nodes.

Every simulation cycle, each node randomly connects to N nodes (representing neighbours found), performs a scoring system on their data, and chooses the highest of them. A range of different scoring functions and parameters were tested in the simulation, beginning with scoring based only on the trust score, QoS, or duration.

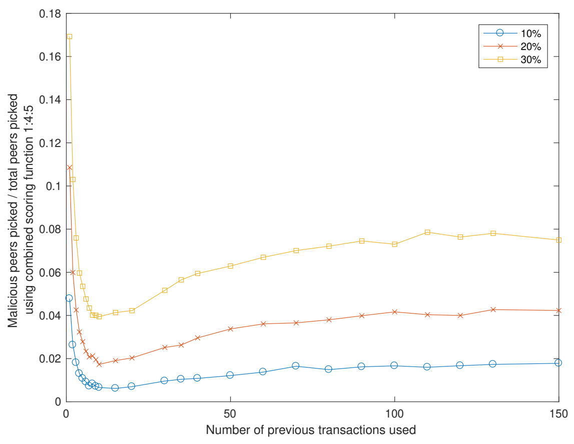

Figure 6.6: Malicious node pick ratio for different number of transactions used in 1:4:5 combined scoring.

Functions combining these parameters were also evaluated.

First, the effectiveness of the different types of scoring policies made by the local nodes is measured. Figure 6.5 shows the performance in terms of malicious node pick ratio (how many malicious nodes are picked as the best relay option based on the scoring function, as a fraction of the total nodes picked). The duration based and parameters combined scoring functions perform the best, followed by the trust based, QoS based, and random picking. Adjusting the ratios of the combined function a 1:4:5 ratio of QoS:trust:duration is found to outperform all strategies.

Next, the impact of the amount of previous transactions used in scoring is assessed, for different malicious percentages in the network. Figure 6.6 shows that the combined function improves with a higher number of transactions used. There is however a trade-off in performance and precision to be considered, as for more than about 10-20 transactions used, the malicious node pick ratio starts increasing again. This is due to the loss of accuracy of current behaviour as it is averaged over a longer period. As 10 transactions achieves a good result and trade-off, this is set constant for the next simulations.

Last, the effect of the amount of peers chosen for the second part of negotiation of peer discovery (as discussed in 6.4.4) on the malicious node pick ratio is examined. There is a trade-off between financial gain and security. The more peers are

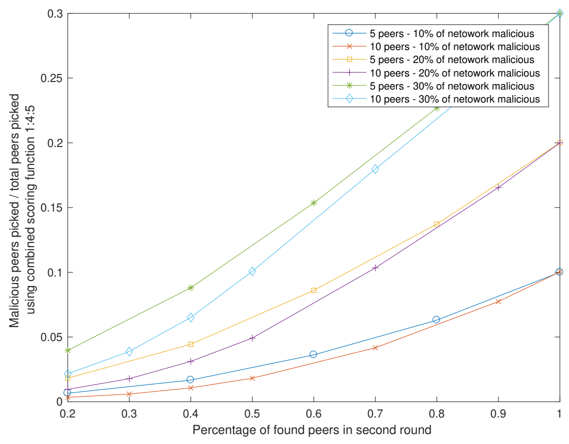

Figure 6.7: Malicious node pick ratio for varying number of nodes invited to second round (of nearby found peers).

invited for auction, the higher the chance of a higher bid, however, this will lower the quality of potential relays as lower scored are permitted, and thereby lower the security guarantee. Figure 6.7 shows the expected upward trend in malicious pick ratio with more peers in the second round, with the steepness of the curve increasing after 40 %. This suggest that the amount of peers chosen for auction should be lower than 40% of discovered peers.

6.5.2 Stabilisation Time Reputation System

As discussed above, the solution for peer discovery based on reputation scores works well in eliminating malicious peers from being chosen as a relay. This is however based on the assumption that all or most peers have had previous transactions on the blockchain, indicating their ability to provide a good and trusted service.

At the system inception, no node will have this data available and therefore some delay can be expected in convergence to a stable reputation system. This period should be as small as possible, as users might be discouraged from using the system, as there are lower guarantees of avoiding malicious peers and incurring unwanted fees.

To evaluate this, two simulations were performed in PeerSim. First, a network of 10000 nodes was used, with malicious percentages of 20%, 30%, and 40% per

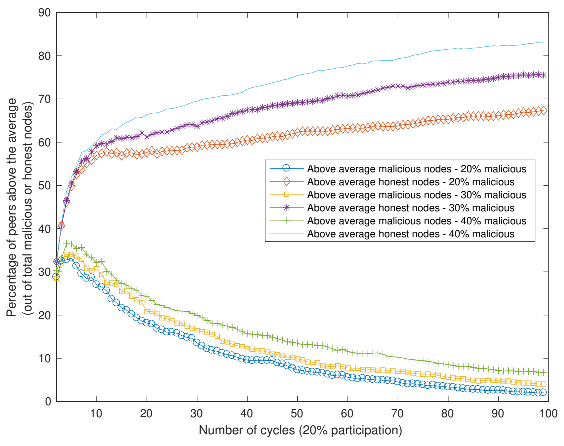

Figure 6.8: Percentage of malicious / honest peers above average score, for different percentages of malicious nodes in the network.

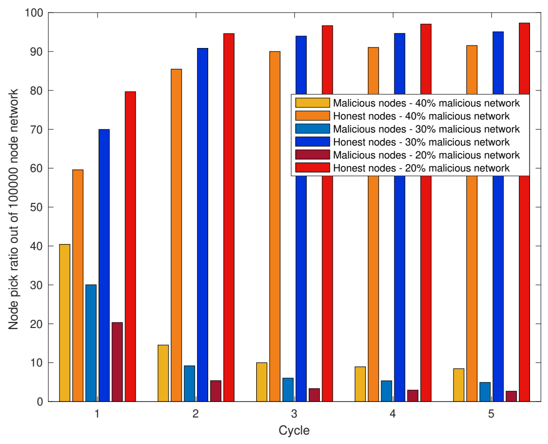

Figure 6.9: Node pick ratio's over time for different percentages of malicious nodes in the network (remembering all previous unacceptable peers).

run (*i.e.* 2000-4000 malicious nodes). Each node starts with no previous transactions and a score of 0. Every simulation cycle, a subset of the network (set at 20%) gets scored, based on different Zipf distributions in order to model malicious and honest nodes. Every cycle the average score is calculated of all nodes in the network. It can then be observed which percentage of malicious and honest nodes are above the average.

Figure 6.8 shows that the time it takes for 90% of malicious nodes to be below the average is between 40 and 75 cycles, depending on the malicious percentage in the network. At this point, between 60% and 80% of honest nodes are above the

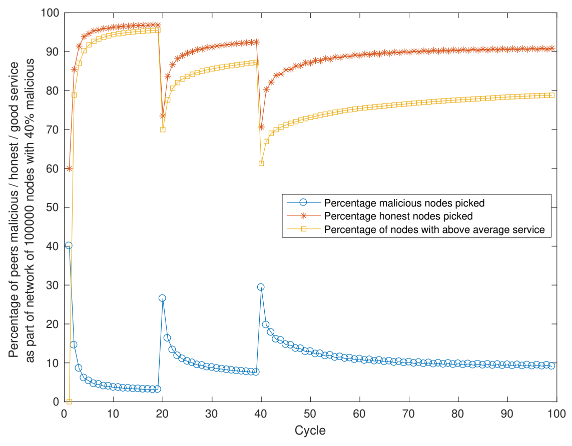

Figure 6.10: Node pick ratio's and ratio of nodes with good service, redistributing maliciousness at 20% and 40% (remembering only last unacceptable peer).

average, with these percentages improving with more cycles. What is important to note is that this stabilisation time is between 80,000 and 150,000 transactions. To put this into context, the Ethereum blockchain has over 1 million transactions per day[1]. The percentage of honest nodes above the average improves and stabilises more quickly, within 10-20 cycles.

Although this seems like a large number of transactions, it can be argued that in practise this won't be the case as all nodes only have a small number of peers out of which they choose the top ones. To substantiate this, the network is set up as before, and all nodes choose their best peers (set to top 5) as a relay, using the combined metric scoring function. All nodes can act both as a relay and a client.

If a peer's score is above a threshold (set so all metrics are above 50%), the peer is defined as acceptable. All nodes are scored every round based on maliciousness, until an honest peer is found with acceptable service. If an unacceptable peer is chosen, it is remembered in a nodes local log, and is omitted from consecutive picking rounds. Figure 6.9 shows the malicious and honest node pick ratios, progressing with more cycles, for different percentages of malicious nodes in the network. It can be observed that within a couple rounds, a malicious node pick ratio of under 10% is achieved for varying levels of malicious nodes in the network.

[1]https://etherscan.io/chart/tx

The strategy of remembering unacceptable peers works well, but falls short when there are a small number of peers nearby which have previously provided unacceptable service, but now may be able to provide adequate service, or when an honest node becomes oversubscribed. Therefore, another strategy is explored where instead of remembering all previous unacceptable peers, a node only remembers the latest one. The network is set as before, but at cycle 20 and 40 the malicious variable is redistributed randomly so that some nodes will change from malicious to honest and vice versa. Figure 6.10 shows how the second strategy is able to largely recover and readjust quickly (within 5-10 cycles) to a low malicious node pick ratio and a high percentage of nodes with good service.

6.5.3 Security Analysis

The security of the proposed system is now examined, after which mitigations against various attacks are discussed.

First, the situation is considered when both c and R_r behave honestly, but due to network events such as failures, the service provided is inadequate, leading to both nodes mutually giving each other negative scores. Although this negatively impacts honest nodes, such network-level failure events are assumed to happen infrequently, and therefore this will not impact the overall trust score of a node significantly. Furthermore, the proposed combined metric scoring uses more metrics besides trust, which leads to a small impact of one-off incorrect scoring. In the negotiation phase, setting a low minimum service requirement also ensures that temporal network congestion does not immediately lead to contract termination, allowing R_r to recover and provide better service.

Collude attacks are considered next between actors in the system. When R_r and c collude to try to boost scores with short forged transactions, they will not be able to impact the overall score by submitting trust scores, as the duration and QoS metrics will be controlled by the smart contract, where duration is measured and QoS inferred from micro-payments. Attempting this attack would also be expensive as every transaction on-chain would incur costs. It is easier and more financially attractive to boost scores by serving nodes properly, while earning rewards.

Function	**Gas Cost**	**Cost slow**	**Cost medium**	**Cost fast**
deploy contract	1738464	0.0086923 ($1.33861)	0.0132123 ($2.03469)	0.0173846 ($2.67723)
start relay agreement	231292	0.0011565 ($0.1781)	0.0017578 ($0.2707)	0.0023129 ($0.35619)
terminate agreement client	48440	0.0002422 ($0.0373)	0.0003681 ($0.05669)	0.0004844 ($0.0746)
terminate agreement relay	78980	0.0003949 ($0.06081)	0.0006002 ($0.09243)	0.0007898 ($0.12163)
timeout	62281	0.0003114 ($0.04796)	0.0004733 ($0.07289)	0.0006228 ($0.09591)
submit trust score client	65173	0.0003259 ($0.05019)	0.0004953 ($0.07628)	0.0006517 ($0.10036)
submit trust score relay	95970	0.0004799 ($0.0739)	0.0007294 ($0.11233)	0.0009597 ($0.14779)
extend relay agreement	34219	0.0001711 ($0.02635)	0.0002601 ($0.04006)	0.0003422 ($0.0527)

Table 6.1: Smart contract costs per function, with different transaction speeds.

Collusion between R_r and R_c to keep c in the dark is also irrational as relaying packets properly would provide financial rewards. Sending dummy messages to c to maximise rewards will be noticed if it is not expecting this traffic, which makes this type of attack unfeasible as well.

Sybil attacks are largely ineffective as the attacker will have finite bandwidth, needed for participation in the system. All new nodes entering the network start with a zero score, and will not be able to consistently serve as R_r unless it provides satisfactory service, due to the peer discovery mechanism. Therefore, creating new identities or rejoining the network with new identities has no benefit for a malicious party. Using sybil identities to boost scores is also mitigated against since collude attacks are ineffective.

The final type of attack discussed is a dishonest scoring attack, where nodes score dishonestly to lower competition in the network. This attack stems from the absence of direct loss when scoring dishonestly. However, when a node is scored incorrectly, it can keep the identity of the dishonest peer in its log and avoid contracting them in the future, and share this information with other peers, which will judge the information based on the nodes score, potentially costing the malicious peer future business. Thus, assuming this attack to be rare, this will not greatly impact the overall score. Further exploration of this mechanism is left for future work.

	terminated by relay	timeout	with 5 extends, terminated by client
Client	296465 ($0.37473)	342013 ($0.4323)	450827 ($0.56984)
Relay	78980 ($0.09982)	141261 ($0.17856)	95970 ($0.12131)

Table 6.2: Smart contract cost for client and relay for different scenarios.

6.5.4 Smart Contract

To evaluate the smart contract performance and costs, the code was implemented in Solidity and deployed on a private virtual blockchain on Ganache. The Remix IDE was also used to interact with the accounts and contract functions.

Table 6.1 shows the costs associated with all functions based on the gas used. This is calculated using the GWEI/GAS estimates from the Ethereum Gas Station and taking the USD conversion rate in April 2020. It is important to reiterate that gas fees can greatly vary, and the overall costs have increased since performing this work. Assuming that layer 2 scaling solutions and Ethereum 2.0 fix the high gas fees in the future, these results can be seen as a worst case scenario.

The different speeds have different costs, and should be used based on the time sensitivity of the function in the contract. Starting a new agreement may be more instantly, as well as submitting or extending before the timeout occurs.

From this data, it can be concluded that client cost of a contract would be around $0.40 - $0.60, and relay cost between $0.10 - $0.20. This is concretely shown in Table 6.2. These costs may become more significant if there is a high frequency of calling relay agreements, but this is not expected as clients should stay with a trusted relay over time.

Out of all contract functions, deployment is the most expensive. This event should happen rarely however, as all agreements are stored on one contract. Contracts can be envisioned to be deployed per application using the relays and the cost paid by the application developer, and potentially have different contracts for geographic regions.

6.6 Summary and Discussion

This chapter presented a reward system for decentralised NAT traversal, where nodes are incentivised to be honest due to financial rewards. To ensure fair exchange, off-chain micro-payments and a smart contract on Ethereum are used. Relays are rewarded proportionally to their performance, and clients are protected from malicious attacks by relays through a Proof of Timely Relay. A two-way reputation system is used to discover nearby honest peers, based on previous open data on the blockchain, and peers can enforce local scoring policies.

The evaluation shows the performance of the reputation system for different scoring policies with different parameters, yielding a low malicious peer pick ratio of 5% using two rounds of peer discovery for up to 30% of malicious nodes in the network. A strategy has been defined which handles changes in maliciousness of nodes, and the system reaches stability within 5-10 cycles of inception. The costs of the interaction with the smart contract were reasonable for both client and relay, given a long term interaction.

While this chapter focused on NAT traversal using relays to send files, this work can be extended to more general relay and bandwidth sharing applications (*e.g.* decentralised content delivery such as real time communication and live-video streaming).

Another direction which extends this work is to set up multiple contracts with peers, creating a pool of relays, allowing clients to have a larger guarantee of being served if one relay is busy. In a file system, this could include per file negotiation, where the client sends expected traffic of retrieving a file and a relay can pledge to serve this file.

Chapter 7

Conclusions and Future Research Directions

7.1 Overview

The DWeb aims to tackle challenges in the current Web infrastructure stemming from centralisation of control, governance, and management. Recent technologies like blockchains and decentralised storage networks (DSN) have introduced methods for establishing trust and storing content without a single root-of-trust. This thesis has explored whether these novel techniques can be leveraged to create decentralised counterparts to centralised Web services, especially in the area of content search and retrieval, without compromising on key features like performance and security.

In order to verify the hypotheses derived in Section 1.5, which state that a DWeb can be realised (although special care needs to be taken to QoS, security, privacy, and performance objectives, as well as novel emerging challenges), this thesis presents an extensive background discussion, as well as a number of novel architectures which solve particular challenges in the DWeb.

First, a framework is proposed for studying related works in the area. This is then used to perform an extensive analysis of both novel industry and research initiatives, as well as building blocks from the P2P era. The three main areas of search engines, name-registries, and file systems are discussed and in each area a number of open issues are defined.

Next, search on the DWeb is explored, first yielding a novel experimental IPFS search mechanism, which uniquely targets decentralised keyword-search using an index-based approach. However, this mechanism is insufficient and therefore informed the design of a novel search mechanism based on similarity search. This mechanism uses locality sensitive hashing (LSH) to extract similarity signatures and records from content, and stores these on a DHT. Various underlying content networks and types are supported, and the system can be used in a wide range of use-cases.

A critical component of the DWeb is collaborative network resource sharing by peers (NRS service). To achieve this, providers and consumers of the resources need a way of finding each other, in a mechanism that captures the following properties simultaneously: scalability, preference based, and price discovery. To achieve this, an allocation system is introduced which is highly scalable, fast, and lightweight, and allows nodes to indicate their preference for clients/sellers without needing to submit bids by using stable matching algorithms. A smart contract is used to decouple price discovery, as well as to implement a billboard orchestrating an allocation and service discovery.

A big issue in P2P networks is NAT traversal, which presents connectivity problems for peers to participate in the network as full peers. Previous solutions to solve this all suffer from the challenges of centralisation. To combat this, a decentralised, relay-based NAT traversal system is presented, where any full node can serve as a relay, for which it is rewarded using off-chain payments through a smart contract. To ensure satisfactory performance, a QoS check and reputation system are added.

These works have addressed, tested, and confirmed the initial hypotheses. *H1* –which states that decentralised network resource sharing services can be designed using blockchain-based incentives– has been addressed in the works on allocation and NAT traversal. *H2* –which states that a decentralised Web can be realised with the aid of network resource sharing services, blockchains, and decentralised storage networks– has been addressed in the proposed framework, as well as in the sections on search and allocation. Finally, *H3* –which states that it is not trivial to ensure satisfactory QoS/QoE, security, privacy, and performance in a decentralised Web– has been explored throughout the chapters.

To summarise, the work in this thesis has concluded that implementing decentralised alternatives to the current Web model is feasible, but non-trivial. Tools like blockchains and DSN are useful, but require careful considerations in incentivisation, security vulnerabilities, and scalability. Furthermore, most works in the area, including this thesis, are early stage research, and therefore the assumptions, design, and results should be verified and tested in future work.

7.2 Future Work

This thesis has explored the foundations of realising a DWeb, especially in content retrieval and delivery. However, as mentioned above, this research area is relatively young, and therefore a number of future directions are discussed below which directly extent the work done in this thesis.

7.2.1 Decentralised Search Implementations

Besides Ditto, few projects aim to provide fully decentralised search on decentralised data, and many still rely on centralised back-end or gateway servers. Using a traditional crawled index approach raises a number of unsolved issues when translated to a decentralised setting, for example due to the immutability of the top level index. Other known challenges include key management, integration with name-registries, as well as meeting performance goals.

These issue should be explored further, as well as alternative workflows like those based on similarity search, producing complete systems. A number of claims regarding the benefits of decentralisation of search should also be explored further to verify they hold in practise.

7.2.2 File System Privacy

While file systems based on DSN have come a long way, they remain vulnerable when it comes to privacy, both in the reader and writer case. These vulnerabilities can be exploited to monitor the entire network traffic, and therefore obfuscation techniques should be prioritised to improve user privacy, without degrading performance.

Another important open question remains around the unclear legal implications of storing content. The legal implications of adding illegal or copyrights content to an index are currently unclear, and requires cross disciplinary work. This also applies to retrieving harmful content from search engines, registering a disputed domain name and more, as there is a lack of moderation techniques available.

7.2.3 Generalised Relays for Bandwidth Sharing

Chapter 6 described using network relays to assist in NAT traversal. This idea may be generalised in order to have a larger impact as a solution to situations where shared bandwidth is needed as a general NRS service. Users who have spare bandwidth may sell this to users who require extra bandwidth, and in turn receive rewards proportional to their work in cryptocurrency. Adding the monetary incentives secures the system against malicious attacks.

There are many use-cases for decentralised bandwidth sharing. First of all, such a system may be used in order to create decentralised VPN's. These differentiate themselves from services such as TOR by offering cryptocurrency rewards. Second of all, bandwidth sharing schemes can enable decentralised CDN's, as well as allow for multi-casting in the case of live video or video conferencing. Finally, improved routing paths may be found compared to the default network path. This way, a premium is paid by a client to receive better QoS.

7.2.4 Reputation System

One of the main benefits of blockchains is that there is no need for a trusted central party, due to the distribution of the shared ledger and the consensus algorithms. However, when sharing services directly from one peer to another, this property of the blockchain cannot be leveraged, as either peer may be selfish and malicious. Therefore, another method of applying trust is needed, as discussed in Section 2.5

Historically, reputation systems have largely been centralised. Distributed reputation systems have been attempted as well, but they failed to reach adoption, due to their complexity and security issues. On top of this, reputation as a metric is difficult, as the view of someones reputation is subjective, and changes in different contexts. The AI-based, personalised mechanism described in Section 5.4 provides a promising direction, but should be explored further in future work.

7.2.5 Decentralised Blockchain CDN

Although a number of projects have started working on decentralised CDN's powered by blockchain, they largely remain in an early stage without much public specifications available. As this is the case, many open questions remain in this area. First of all, studies are needed to assess the feasibility of decentralised CDN's, and how their performance would compare to the current infrastructure, as well as hybrid PA-CDN's.

Content placement is another open challenge, as traditionally in DHT's content is placed at the user with the closest ID, but in a CDN awareness and consideration is required of the location of the nodes where content is cached. For a commercial party to use a decentralised CDN, they would need assurance that the content is widely available and quickly accessible, which would require their content to be available at a number of geographically spread out nodes. The question of how often data is replicated also arises.

Another open question is that of incentivisation and who will pay for decentralised nodes to cache content. A node should also earn more rewards as their QoS provided is larger, but should not be penalised if network failures occur.

Bibliography

[1] Anutosh Banerjee, Robert Byrne, Ian De Bode, and Matt Higginson. Web3 beyond the hype. Technical report, McKinsey, 2022.

[2] Haytham Yassine, Ed Crouch, Kestas Sereiva, Sesh Iyer, and Mark Abraham. Web3 opens new paths to customer loyalty. Technical report, BCG, 2023.

[3] Number of identity-verified cryptoasset users from 2016 to november 2022 (in millions) [graph]. `https://www.statista.com/statistics/1202503/global-cryptocurrency-user-base/`, 2022.

[4] Matthieu Nadini, Laura Alessandretti, Flavio Di Giacinto, Mauro Martino, Luca Maria Aiello, and Andrea Baronchelli. Mapping the nft revolution: market trends, trade networks, and visual features. *Scientific reports*, 11(1):20902, 2021.

[5] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach (7th Edition)*. Pearson, 7th edition, 2017.

[6] David Clark. The design philosophy of the darpa internet protocols. *SIGCOMM Comput. Commun. Rev.*, 18(4):106–114, August 1988.

[7] Simon Forrest Chris Evans, Julian Issa. The sustainable future of video entertainment. Technical report, Cisco, 2020.

[8] Eric K. Clemons. Business models for monetizing internet applications and web sites: Experience, theory, and predictions. *Journal of Management Information Systems*, 26(2):15–41, 2009.

[9] Tharam Dillon, Chen Wu, and Elizabeth Chang. Cloud computing: issues and challenges. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 27–33. Ieee, 2010.

[10] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[11] Rajkumar Buyya, Mukaddim Pathan, and Athena Vakali. *Content delivery networks*, volume 9. Springer Science & Business Media, 2008.

[12] Behrouz Zolfaghari, Gautam Srivastava, Swapnoneel Roy, Hamid R. Nemati, Fatemeh Afghah, Takeshi Koshiba, Abolfazl Razi, Khodakhast Bibak, Pinaki Mitra, and Brijesh Kumar Rai. Content delivery networks: State of the art, trends, and future roadmap. *ACM Comput. Surv.*, 53(2), apr 2020.

[13] George Pavlou and Ioannis Psaras. The troubled journey of qos: from atm to content networking, edge-computing and distributed internet governance. *Computer Communications*, 07 2018.

[14] Ricky K. P. Mok, Edmond W. W. Chan, and Rocky K. C. Chang. Measuring the quality of experience of http video streaming. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 485–492, 2011.

[15] Vili Lehdonvirta. *Cloud Empires: How Digital Platforms Are Overtaking the State and How We Can Regain Control.* MIT Press, 2022.

[16] Martin Moore. *Tech Giants and Civic Power.* CMCP, Policy Institute, King's College London, April 2016.

[17] Fact sheet how breaking up amazon can empower small business. In *Institute for Local Self-Reliance*, 2022.

[18] Google and facebook ad deal. In *New York Times*, 2021.

[19] Amy Orben, Tobias Dienlin, and Andrew K. Przybylski. Social media's enduring effect on adolescent life satisfaction. *Proceedings of the National Academy of Sciences*, 116(21):10226–10228, 2019.

[20] House of Commons Science and Technology Committee. Impact of social media and screen-use on young people's health. Technical report, House of Commons, 2019.

[21] Florian Rehbein, Daniel L King, Andreas Staudt, Tobias Hayer, and Hans-Jürgen Rumpf. Contribution of game genre and structural game characteristics to the risk of problem gaming and gaming disorder: A systematic review. *Current Addiction Reports*, 8(2):263–281, 2021.

[22] Eli Pariser. *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Group , The, 2011.

[23] Daniel Allington. *Conspiracy Theories, Radicalisation and Digital Media*. Global Network on Extremism and Technology, February 2021.

[24] Russell Brandom and Jay Peters. Russia says it's blocking facebook in alarming new censorship push. In *The Verge*, 2022.

[25] Adam Minter. China is cranking up its global propaganda machine. In *Bloomberg*, 2022.

[26] The all conquering quaver. In *The Economist*, 2022.

[27] Jake Swearingen. When amazon web services goes down, so does a lot of the web. *New York Magazine*, 2018.

[28] Anthony Cuthbertson. Facebook down: Users report issues with messenger and instagram. *The Independent*.

[29] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23, October 2009.

[30] Mahadev Satyanarayanan. The emergence of edge computing. *IEEE Computer*, 50(1):30–39, 2017.

[31] Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proceedings of the 4th International Conference on Peer-to-Peer Systems*, IPTPS'05, page 205–216, Berlin, Heidelberg, 2005. Springer-Verlag.

[32] Pedro García López, Alberto Montresor, and Anwitaman Datta. Please, do not decentralize the internet with (permissionless) blockchains! *CoRR*, abs/1904.13093, 2019.

[33] Lian Jian and Jeffrey K. MacKie-Mason. Why share in peer-to-peer networks? In *Proceedings of the 10th International Conference on Electronic Commerce*, ICEC '08, New York, NY, USA, 2008. Association for Computing Machinery.

[34] Jeffrey Shneidman and David C. Parkes. Rationality and self-interest in peer to peer networks. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II*, pages 139–148, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[35] Ke Zhang et al. Mobile-Edge Computing for Vehicular Networks. *IEEE Vehicular Technology Magazine*, 12(June):2–10, 2017.

[36] Eve M. Schooler, David Zage, Jeff Sedayao, Hassnaa Moustafa, Andrew Brown, and Moreno Ambrosin. An architectural vision for a data-centric iot: Rethinking things, trust and clouds. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1717–1728, 2017.

[37] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `http://www.bitcoin.org/bitcoin.pdf`, 2009.

[38] Vitalik Buterin. A next-generation smart contract and decentralized application platform. 2015.

[39] Arvind Narayanan and Jeremy Clark. Bitcoin's academic pedigree: The concept of cryptocurrencies is built from forgotten ideas in research literature. *Queue*, 15(4):20–49, aug 2017.

[40] Adam Back. Hashcash - a denial of service counter-measure. "`http://diyhpl.us/~bryan/papers2/bitcoin/Hashcash%20-%20a%20denial-of-service%20counter-measure.pdf`", 09 2002.

[41] David Chaum. Blind signatures for untraceable payments. In *Annual International Cryptology Conference*, 1982.

[42] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14:352, 10 2018.

[43] Klaus Schwab. *The Fourth Industrial Revolution*. Crown Publishing Group, New York, NY, USA, 2017.

[44] Juan Benet and Nicola Greco. Filecoin: A decentralized storage network. *Protoc. Labs*, pages 1–36, 2018.

[45] The Golem Project. Golem whitepaper. `https://golem.network/crowdfunding/Golemwhitepaper.pdf`, 2016.

[46] Mysterium network project whitepaper. `https://mysterium.network/whitepaper.pdf`, October 2017.

[47] Jake S. Cannell, Justin Sheek, Jay Freeman, Greg Hazel, Jennifer Rodriguez-Mueller, Eric Hou, Brian J. Fox, and Steven Waterhouse. Orchid: A decentralized network routing market. `https://www.orchid.com/assets/whitepaper/whitepaper.pdf`, November 2019.

[48] A blockchain framework for building decentralized vpn applications. https://sentinel.co/whitepaper.pdf, 2020.

[49] Erik Daniel and Florian Tschorsch. Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys & Tutorials*, 24(1):31–52, 2022.

[50] Juan Benet. Ipfs-content addressed, versioned, p2p file system. https://arxiv.org/abs/1407.3561, 2014.

[51] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Borje Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.

[52] Navin V. Keizer, Onur Ascigil, Michał Król, Dirk Kutscher, and George Pavlou. A survey on content retrieval on the decentralised web. In *Journal in Submission*, 2023.

[53] Navin V. Keizer and Puneet Bindlish. Deece search: Decentralised search for ipfs. https://github.com/navinkeizer/Deece, 2021.

[54] Navin V. Keizer, Onur Ascigil, Michał Król, and George Pavlou. Ditto: Towards decentralised similarity search for web3 services. In *Conference in Submission*, 2023.

[55] Navin V. Keizer, Onur Ascigil, Ioannis Psaras, and George Pavlou. Flock: Fast, lightweight, and scalable allocation for decentralized services on blockchain. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9, 2021.

[56] Navin V. Keizer, Fan Yang, Ioannis Psaras, and George Pavlou. The case for ai based web3 reputation systems. In *2021 IFIP Networking Conference (IFIP Networking)*, pages 1–2, 2021.

[57] Navin V. Keizer, Onur Ascigil, Ioannis Psaras, and George Pavlou. Rewarding relays for decentralised nat traversal using smart contracts. In *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, Mobihoc '20, page 309–314, New York, NY, USA, 2020. Association for Computing Machinery.

[58] Andrew S. Tanenbaum and David Wetherall. *Computer Networks*. Prentice Hall, Boston, 5 edition, 2011.

[59] Dimitrios K. Vassilakis and Vasilis Vassalos. Modelling real p2p networks: The effect of altruism. In *Seventh IEEE International Conference on Peer-to-Peer Computing (P2P 2007)*, pages 19–26, 2007.

[60] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings First International Conference on Peer-to-Peer Computing*, pages 99–100, 2001.

[61] Ken Birman. The promise, and limitations, of gossip protocols. *SIGOPS Oper. Syst. Rev.*, 41(5):8–13, oct 2007.

[62] Nathaniel Leibowitz, Matei Ripeanu, and Adam Wierzbicki. Deconstructing the kazaa network. In *Proceedings the Third IEEE Workshop on Internet Applications. WIAPP 2003*, pages 112–120, 2003.

[63] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM 2004*, volume 1, page 130, 2004.

[64] Liang Wang, Suzan Bayhan, Jörg Ott, Jussi Kangasharju, and Jon Crowcroft. Understanding scoped-flooding for content discovery and caching in content networks. *IEEE Journal on Selected Areas in Communications*, 36(8):1887–1900, 2018.

[65] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In Peter Druschel, Frans Kaashoek,

and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 53–65, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[66] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev.*, 31(4):149–160, August 2001.

[67] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In Rachid Guerraoui, editor, *Middleware 2001*, pages 329–350, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[68] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. *SIGCOMM Comput. Commun. Rev.*, 31(4):161–172, aug 2001.

[69] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, USA, 2001.

[70] Zied Trifa and Maher Ali Khemakhem. Taxonomy of structured p2p overlay networks security attacks. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 6:470–476, 2012.

[71] Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. Consensus in the age of blockchains. *CoRR*, abs/1711.03936, 2017.

[72] Arthur Gervais, Ghassan O. Karame, Karl Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. On the security and performance of proof of work blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 3–16, New York, NY, USA, 2016. Association for Computing Machinery.

[73] Daniel Fullmer and A. Stephen Morse. Analysis of difficulty control in bitcoin and proof-of-work blockchains. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 5988–5992, 2018.

[74] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Advances in Cryptology–CRYPTO 2017: 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part I*, pages 357–388. Springer, 2017.

[75] Cong T. Nguyen, Dinh Thai Hoang, Diep N. Nguyen, Dusit Niyato, Huynh Tuong Nguyen, and Eryk Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019.

[76] Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of financial studies*, 34(3):1156–1190, 2021.

[77] Hoang Giang Do and Wee Keong Ng. Blockchain-based system for secure data storage with private keyword search. In *2017 IEEE World Congress on Services (SERVICES)*, pages 90–93, 2017.

[78] Nazanin Zahed Benisi, Mehdi Aminian, and Bahman Javadi. Blockchain-based decentralized storage networks: A survey. *Journal of Network and Computer Applications*, 162:102656, 2020.

[79] Stefano De Angelis, Leonardo Aniello, Roberto Baldoni, Federico Lombardi, Andrea Margheri, and Vladimiro Sassone. Pbft vs proof-of-authority: Applying the cap theorem to permissioned blockchain. 2018.

[80] Hanaa Abbas, Maurantonio Caprolu, and Roberto Di Pietro. Analysis of polkadot: Architecture, internals, and contradictions. In *2022 IEEE International Conference on Blockchain (Blockchain)*, pages 61–70, 2022.

[81] Alfonso Cevallos and Alistair Stewart. A verifiably secure and proportional committee election rule. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, AFT '21, page 29–42, New York, NY, USA, 2021. Association for Computing Machinery.

[82] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[83] Amirmohammad Pasdar, Young Choon Lee, and Zhongli Dong. Connect api with blockchain: A survey on blockchain oracle implementation. *ACM Comput. Surv.*, oct 2022. Just Accepted.

[84] Hamda Al-Breiki, Muhammad Habib Ur Rehman, Khaled Salah, and Davor Svetinovic. Trustworthy blockchain oracles: Review, comparison, and open research challenges. *IEEE Access*, 8:85675–85685, 2020.

[85] Giulio Caldarelli. Understanding the blockchain oracle problem: A call for action. *Information*, 11(11):509, Oct 2020.

[86] R. Aroul Canessane, N. Srinivasan, Abinash Beuria, Ashwini Singh, and B. Muthu Kumar. Decentralised applications using ethereum blockchain. In *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, volume 1, pages 75–79, 2019.

[87] Eugenia Politou, Fran Casino, Efthimios Alepis, and Constantinos Patsakis. Blockchain mutability: Challenges and proposed solutions. *IEEE Transactions on Emerging Topics in Computing*, 9(4):1972–1986, 2021.

[88] Mainak Ghosh, Miles Richardson, Brian Ford, and Rob Jansen. A torpath to torcoin: Proof-of-bandwidth altcoins for compensating relays. 2014.

[89] Xixun Yu and Athanasios Vasilakos. A survey of verifiable computation. *Mobile Networks and Applications*, 22:1–16, 06 2017.

[90] Juan Benet, David Dalrymple, and Nicola Greco. Proof of replication technical report (wip). Technical report, Protocol Labs, 2017.

[91] Elli Androulaki, Ghassan O. Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security*, pages 34–51, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[92] Mohammad Dabbagh, Kim-Kwang Raymond Choo, Amin Beheshti, Mohammad Tahir, and Nader Sohrabi Safa. A survey of empirical performance evaluation of permissioned blockchain platforms: Challenges and opportunities. *Computers & Security*, 100:102078, 2021.

[93] Christine V Helliar, Louise Crawford, Laura Rocca, Claudio Teodori, and Monica Veneziani. Permissionless and permissioned blockchain diffusion. *International Journal of Information Management*, 54:102136, 2020.

[94] Manlu Liu, Kean Wu, and Jennifer Jie Xu. How will blockchain technology impact auditing and accounting: Permissionless versus permissioned blockchain. *Current Issues in auditing*, 13(2):A19–A29, 2019.

[95] James Clavin, Sisi Duan, Haibin Zhang, Vandana P. Janeja, Karuna P. Joshi, Yelena Yesha, Lucy C. Erickson, and Justin D. Li. Blockchains for government: Use cases and challenges. *Digit. Gov.: Res. Pract.*, 1(3), nov 2020.

[96] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, and et al. Hyperledger fabric. *Proceedings of the Thirteenth EuroSys Conference*, Apr 2018.

[97] Gianmaria Del Monte, Diego Pennino, and Maurizio Pizzonia. Scaling blockchains without giving up decentralization and security: A solution to the blockchain scalability trilemma. In *Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, CryBlock '20, page 71–76, New York, NY, USA, 2020. Association for Computing Machinery.

[98] Amani Altarawneh, Tom Herschberg, Sai Medury, Farah Kandah, and Anthony Skjellum. Buterin's scalability trilemma viewed through a state-change-based classification for common consensus algorithms. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0727–0736, 2020.

[99] Mauro Conti, E. Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.

[100] Rui Zhang, Rui Xue, and Ling Liu. Security and privacy on blockchain. *ACM Comput. Surv.*, 52(3), jul 2019.

[101] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[102] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptol.*, 7(1):1–32, December 1994.

[103] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. Gossipsub: Attack-resilient message propagation in the filecoin and eth2.0 networks, 2020.

[104] Gleb Naumenko, Gregory Maxwell, Pieter Wuille, Alexandra Fedorova, and Ivan Beschastnikh. Bandwidth-efficient transaction relay for bitcoin. *CoRR*, abs/1905.10518, 2019.

[105] Peter Fairley. Ethereum will cut back its absurd energy use. *IEEE Spectrum*, 56(1):29–32, 2019.

[106] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in*

Financial Technologies, AFT '19, page 41–61, New York, NY, USA, 2019. Association for Computing Machinery.

[107] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. In Joseph Bonneau and Nadia Heninger, editors, *Financial Cryptography and Data Security*, pages 201–226, Cham, 2020. Springer International Publishing.

[108] Louis Tremblay Thibault, Tom Sarry, and Abdelhakim Senhaji Hafid. Blockchain scaling using rollups: A comprehensive survey. *IEEE Access*, 10:93039–93054, 2022.

[109] Jeff Burdges, Alfonso Cevallos, Peter Czaban, Rob Habermeier, Syed Hosseini, Fabio Lama, Handan Kilinc Alper, Ximin Luo, Fatemeh Shirazi, Alistair Stewart, et al. Overview of polkadot and its design considerations. *arXiv preprint arXiv:2005.13456*, 2020.

[110] Gavin Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White paper*, 21(2327):4662, 2016.

[111] Stefan Dziembowski, Sebastian Faust, and Kristina Hostáková. General state channel networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 949–966, New York, NY, USA, 2018. Association for Computing Machinery.

[112] Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. Sok: Layer-two blockchain protocols. Cryptology ePrint Archive, Report 2019/360, 2019. `https://eprint.iacr.org/2019/360`.

[113] Yotam Sali and Aviv Zohar. Optimizing off-chain payment networks in cryptocurrencies, 2020.

[114] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. pages 106–123, 05 2019.

[115] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, January 2016.

[116] Frederik Armknecht, Ghassan O Karame, Avikarsha Mandal, Franck Youssef, and Erik Zenner. Ripple: Overview and outlook. In *Trust and Trustworthy Computing: 8th International Conference, TRUST 2015, Heraklion, Greece, August 24-26, 2015, Proceedings 8*, pages 163–180. Springer, 2015.

[117] Yuhui Zhang and Dejun Yang. Robustpay+: Robust payment routing with approximation guarantee in blockchain-based payment channel networks. *IEEE/ACM Transactions on Networking*, 29(4):1676–1686, 2021.

[118] Vibhaalakshmi Sivaraman, Shaileshh Bojja Venkatakrishnan, Kathleen Ruan, Parimarjan Negi, Lei Yang, Radhika Mittal, Giulia Fanti, and Mohammad Alizadeh. High throughput cryptocurrency routing in payment channel networks. In *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation*, NSDI'20, page 777–796, USA, 2020. USENIX Association.

[119] Stefanie Roos, Pedro Moreno-Sanchez, Aniket Kate, and Ian Goldberg. Settling payments fast and private: Efficient decentralized routing for path-based transactions. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[120] Krzysztof Pietrzak, Iosif Salem, Stefan Schmid, and Michelle Yeo. Lightpir: Privacy-preserving route discovery for payment channel networks. In *2021 IFIP Networking Conference (IFIP Networking)*, pages 1–9, 2021.

[121] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, and Matteo Maffei. Silentwhispers: Enforcing security and privacy in decentralized credit networks. In *Proceedings of the 24th Annual Symposium on Network and Distributed System Security (NDSS '17)*, February 2017.

[122] Giulio Malavolta, Pedro Moreno-Sanchez, Aniket Kate, Matteo Maffei, and Srivatsan Ravi. Concurrency and privacy with payment-channel networks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, page 455–471, New York, NY, USA, 2017. Association for Computing Machinery.

[123] Felix Engelmann, Henning Kopp, Frank Kargl, Florian Glaser, and Christof Weinhardt. Towards an economic analysis of routing in payment channel networks. In *Proceedings of the 1st Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, SERIAL '17, New York, NY, USA, 2017. Association for Computing Machinery.

[124] Peng Wang, Hong Xu, Xin Jin, and Tao Wang. Flash: Efficient dynamic routing for offchain networks. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, CoNEXT '19, page 370–381, New York, NY, USA, 2019. Association for Computing Machinery.

[125] Yiannis Psaras and David Dias. The interplanetary file system and the filecoin network. In *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, pages 80–80, 2020.

[126] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, March 2007.

[127] Minaxi Gupta, Paul Judge, and Mostafa Ammar. A reputation system for peer-to-peer networks. In *Proceedings of the 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, NOSSDAV '03, page 144–152, New York, NY, USA, 2003. Association for Computing Machinery.

[128] Debora Donato, Mario Paniccia, Maddalena Selis, Carlos Castillo, Giovanni Cortese, and Stefano Leonardi. New metrics for reputation management in p2p networks. In *Proceedings of the 3rd International Workshop on Adversarial Information Retrieval on the Web*, AIRWeb '07, page 65–72, New York, NY, USA, 2007. Association for Computing Machinery.

[129] Tim Moreton and Andrew Twigg. Trading in trust, tokens, and stamps. In *In Proc. of the First Workshop on Economics of Peer-to-Peer Systems*, 2003.

[130] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, page 640–651, New York, NY, USA, 2003. Association for Computing Machinery.

[131] Arash Molavi Kakhki, Chloe Kliman-Silver, and Alan Mislove. Iolaus: Securing online content rating systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 919–930, 2013.

[132] Haifeng Yu, Chenwei Shi, Michael Kaminsky, Phillip B Gibbons, and Feng Xiao. Dsybil: Optimal sybil-resistance for recommendation systems. In *2009 30th IEEE Symposium on Security and Privacy*, pages 283–298. IEEE, 2009.

[133] Ammar Battah, Youssef Iraqi, and Ernesto Damiani. Blockchain-based reputation systems: Implementation challenges and mitigation. *Electronics*, 10, 01 2021.

[134] Emanuele Bellini, Youssef Iraqi, and Ernesto Damiani. Blockchain-based distributed trust and reputation management systems: A survey. *IEEE Access*, 8:21127–21151, 2020.

[135] Michal Król, Alberto Sonnino, Mustafa Al-Bassam, Argyrios G. Tasiopoulos, and Ioannis Psaras. Proof-of-prestige: A useful work reward system for unverifiable tasks. *CoRR*, abs/1905.03016, 2019.

[136] Richard Dennis and Gareth Huw Owenson. Rep on the roll: a peer to peer reputation system based on a rolling blockchain. *International Journal for Digital Society*, 7(1):1123–1134, 3 2016.

[137] B. Wellman and C. Haythornthwaite. *The Internet in Everyday Life*. Information Age Series. Wiley, 2008.

[138] John H Aldrich, Rachel K Gibson, Marta Cantijoch, and Tobias Konitzer. Getting out the vote in the social media era: Are digital tools changing the extent, nature and impact of party contacting in elections? *Party Politics*, 22(2):165–178, 2016.

[139] Jari Arkko, Brian Trammell, Mark Nottingham, Christian Huitema, Martin Thomson, Jeff Tantsura, and Niels ten Oever. Considerations on internet consolidation and the internet architecture. Technical report, IETF, 2020.

[140] Carlos Castillo. Fairness and transparency in ranking. *SIGIR Forum*, 52(2):64–71, jan 2019.

[141] Bernhard Debatin, Jennette P Lovejoy, Ann-Kathrin Horn, and Brittany N Hughes. Facebook and online privacy: Attitudes, behaviors, and unintended consequences. *Journal of computer-mediated communication*, 15(1):83–108, 2009.

[142] April Glaser. How apple and amazon are aiding chinese censors. `https://slate.com/technology/2017/08/apple-and-amazon-are-helping-china-censor-the-internet.html`, 2018.

[143] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, and Kiran Nagaraja. Peer-to-peer computing. Technical report, 2002.

[144] Jim Waldo. A hitchhiker's guide to the blockchain universe. *Commun. ACM*, 62(3):38–42, February 2019.

[145] Radha Mookerjee, Subodha Kumar, and Vijay S Mookerjee. Optimizing performance-based internet advertisement campaigns. *Operations Research*, 65(1):38–54, 2017.

[146] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. Design and evaluation of ipfs: A storage layer for the decentralized web. In *Proceedings of the ACM SIGCOMM 2022 Conference*, SIGCOMM '22, page 739–752, New York, NY, USA, 2022. Association for Computing Machinery.

[147] T. Doan, Y. Psaras, J. Ott, and V. Bajpai. Toward decentralized cloud storage with ipfs: Opportunities, challenges, and future considerations. *IEEE Internet Computing*, 26(06):7–15, nov 2022.

[148] Mei Kobayashi and Koichi Takeda. Information retrieval on the web. *ACM Computing Surveys (CSUR)*, 32(2):144–173, 2000.

[149] Xi Li, Zehua Wang, Victor C. M. Leung, Hong Ji, Yiming Liu, and Heli Zhang. Blockchain-empowered data-driven networks: A survey and outlook. *ACM Comput. Surv.*, 54(3), apr 2021.

[150] Nazanin Zahed Benisi, Mehdi Aminian, and Bahman Javadi. Blockchain-based decentralized storage networks: A survey. *J. Netw. Comput. Appl.*, 162:102656, 2020.

[151] Till Neudecker and Hannes Hartenstein. Network layer aspects of permissionless blockchains. *IEEE Communications Surveys Tutorials*, 21(1):838–857, 2019.

[152] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys Tutorials*, 7(2):72–93, 2005.

[153] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM computing surveys (CSUR)*, 36(4):335–371, 2004.

[154] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell. A survey of peer-to-peer storage techniques for distributed file systems. In *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II*, volume 2, pages 205–213 Vol. 2, 2005.

[155] George Xylomenos, Christopher N. Ververidis, Vasilios A. Siris, Nikos Fotiou, Christos Tsilopoulos, Xenofon Vasilakos, Konstantinos V. Katsaros, and George C. Polyzos. A survey of information-centric networking research. *IEEE Communications Surveys Tutorials*, 16(2):1024–1049, 2014.

[156] Behrouz Zolfaghari, Gautam Srivastava, Swapnoneel Roy, Hamid R. Nemati, Fatemeh Afghah, Takeshi Koshiba, Abolfazl Razi, Khodakhast Bibak, Pinaki Mitra, and Brijesh Kumar Rai. Content delivery networks: State of the art, trends, and future roadmap. *ACM Comput. Surv.*, 53(2), apr 2020.

[157] Milad Ghaznavi, Elaheh Jalalpour, Mohammad A. Salahuddin, Raouf Boutaba, Daniel Migault, and Stere Preda. Content delivery network security: A survey. *IEEE Communications Surveys Tutorials*, 23(4):2166–2190, 2021.

[158] Pavel Mach and Zdenek Becvar. Mobile edge computing: A survey on architecture and computation offloading. *IEEE Communications Surveys Tutorials*, 19(3):1628–1656, 2017.

[159] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys Tutorials*, 19(4):2322–2358, 2017.

[160] Nasreen Anjum, Dmytro Karamshuk, Mohammad Shikh-Bahaei, and Nishanth Sastry. Survey on peer-assisted content delivery networks. *Computer Networks*, 116:79–95, 2017.

[161] Qingmin Jia, Renchao Xie, Tao Huang, Jiang Liu, and Yunjie Liu. The collaboration for content delivery and network infrastructures: A survey. *IEEE Access*, 5:18088–18106, 2017.

[162] Ruizhe Yang, F. Richard Yu, Pengbo Si, Zhaoxin Yang, and Yanhua Zhang. Integrated blockchain and edge computing systems: A survey, some research issues and challenges. *IEEE Communications Surveys Tutorials*, 21(2):1508–1532, 2019.

[163] Michael Walfish, Hari Balakrishnan, and Scott Shenker. Untangling the web from dns. In *NSDI*, volume 4, pages 17–17, 2004.

[164] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7:1–32, 1994.

[165] Swarm. Swarm: Storage and communication infrastructure for a self-sovereign digital society. `https://www.ethswarm.org/swarm-whitepaper.pdf`, 2021.

[166] Storj: A decentralized cloud storage network framework. `https://storj.io/storj.pdf`, October 2018.

[167] David Vorick. Skynet. `https://blog.sia.tech/skynet-bdf0209d6d34`.

[168] David Vorick and Luke Champine. Sia: Simple decentralized storage. *Nebulous Inc*, 2014.

[169] A decentralized video delivery and streaming network powered by a new blockchain. `https://s3.us-east-2.amazonaws.com/assets.thetatoken.org/Theta-white-paper-latest.pdf`, 2018.

[170] Theta mainnet 3.0 whitepaper. `https://s3.us-east-2.amazonaws.com/assets.thetatoken.org/Theta-white-paper-3-0-latest.pdf`, 2020.

[171] Doug Petkanics and Eric Tang. Livepeer whitepaper: Protocol and economic incentives for a decentralized live video streaming network. `https://github.com/livepeer/wiki/blob/master/WHITEPAPER.md`.

[172] David Mazières. Self-certifying file system. `https://pdos.csail.mit.edu/~ericp/doc/sfs-thesis.ps`, 2000.

[173] Zooko Wilcox-O'Hearn. Names: Decentralized, secure, human-meaningful: Choose two. `https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html`, 2001.

[174] Presearch. Presearch whitepaper. `https://www.presearch.io/uploads/WhitePaper.pdf`, 2017.

[175] YaCy. Yacy decentralized web search. `https://yacy.net`, 2004.

[176] Brave Software. Basic attention token (bat) blockchain based digital advertising. `https://basicattentiontoken.org/static-assets/documents/BasicAttentionTokenWhitePaper-4.pdf`, 2021.

[177] Nebulas. Nebulas technical white paper. `https://nebulas.io/docs/NebulasTechnicalWhitepaper.pdf`, 2018.

[178] Brandon Ramirez. The graph network in depth. `https://thegraph.com/blog/the-graph-network-in-depth-part-1`, 2020.

[179] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[180] Eric K. Clemons. Business models for monetizing internet applications and web sites: Experience, theory, and predictions. *Journal of Management Information Systems*, 26(2):15–41, 2009.

[181] Jake Swearingen. When amazon web services goes down, so does a lot of the web. 2018.

[182] Nawras Khudhur and Satoshi Fujita. Siva-the ipfs search engine. In *2019 Seventh International Symposium on Computing and Networking (CANDAR)*, pages 150–156. IEEE, 2019.

[183] Mingyu Li, Jinhao Zhu, Tianxu Zhang, Cheng Tan, Yubin Xia, Sebastian Angel, and Haibo Chen. Bringing decentralized search to decentralized services. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, pages 331–347, 2021.

[184] Mirko Zichichi, Luca Serena, Stefano Ferretti, and Gabriele D'Angelo. Governing decentralized complex queries through a dao. In *Proceedings of the Conference on Information Technology for Social Good*, pages 121–126, 2021.

[185] Liyan Zhu, Chuqiao Xiao, and Xueqing Gong. Keyword search in decentralized storage systems. *Electronics*, 9(12):2041, 2020.

[186] Feng Wang and Yanjun Wu. Keyword search technology in content addressable storage system. In *2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 728–735. IEEE, 2020.

[187] Steve Waterhouse. Jxta search: Distributed search for distributed networks. 2001.

[188] Yi Qiao and Fabián E. Bustamante. Structured and unstructured overlays under the microscope: A measurement-based view of two p2p systems that people use. In *2006 USENIX Annual Technical Conference (USENIX ATC 06)*, Boston, MA, May 2006. USENIX Association.

[189] Wesley W. Terpstra, Jussi Kangasharju, Christof Leng, and Alejandro P. Buchmann. Bubblestorm: Resilient, probabilistic, and exhaustive peer-to-peer search. *SIGCOMM Comput. Commun. Rev.*, 37(4):49–60, August 2007.

[190] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th International Conference on Supercomputing*, ICS ’02, page 84–95, New York, NY, USA, 2002. Association for Computing Machinery.

[191] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM ’03, page 407–418, New York, NY, USA, 2003. Association for Computing Machinery.

[192] Faroo. Faroo. `https://web.archive.org/web/20150914205049/http://www.faroo.com/hp/p2p/p2p.html`, 2007.

[193] Seeks. Seeks faq. `https://github.com/beniz/seeks/wiki/FAQ`, 2014.

[194] Sebastian Michel, Peter Triantafillou, and Gerhard Weikum. Minerva ∞: A scalable efficient peer-to-peer search engine. In *Proceedings of the ACM/IFIP/USENIX 6th International Conference on Middleware*, Middleware’05, page 60–81, Berlin, Heidelberg, 2005. Springer-Verlag.

[195] Kai-hsiang Yang and Jan-ming Ho. Proof: A dht-based peer-to-peer search engine. In *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI’06)*, pages 702–708, 2006.

[196] Tim Lu, Shan Sinha, and Ajay Sudan. Panache: A scalable distributed index for keyword search. 01 2003.

[197] Patrick Reynolds and Amin Vahdat. Efficient peer-to-peer keyword searching. In *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, Middleware ’03, page 21–40, Berlin, Heidelberg, 2003. Springer-Verlag.

[198] Blaise Gassend, Thomer M. Gil, and Bin Song. Dinx: A decentralized search engine. 2001.

[199] Herwig Unger and Markus Wulff. Towards a decentralized search engine for p2p-network communities. In *Eleventh Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2003. Proceedings.*, pages 492–499, 2003.

[200] Boon Thau Loo, Ryan Huebsch, Ion Stoica, and Joseph M. Hellerstein. The case for a hybrid p2p search infrastructure. In *Proceedings of the Third International Conference on Peer-to-Peer Systems*, IPTPS'04, page 141–150, Berlin, Heidelberg, 2004. Springer-Verlag.

[201] Michael Herrmann, Ren Zhang, Kai-Chun Ning, Claudia Diaz, and Bart Preneel. Censorship-resistant and privacy-preserving distributed web search. In *14-th IEEE International Conference on Peer-to-Peer Computing*, pages 1–10, 2014.

[202] ipfs search. ipfs-search documentation. `https://ipfs-search.readthedocs.io/en/latest/index.html`, 2021.

[203] Sebastian A. Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. Mapping the interplanetary filesystem. *2020 IFIP Networking Conference (Networking)*, pages 289–297, 2020.

[204] Leonhard Balduf, Sebastian A. Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. Monitoring data requests in decentralized data storage systems: A case study of ipfs. *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 658–668, 2021.

[205] Hien Tran, Tarek Menouer, Patrice Darmon, Abdoulaye Doucoure, and François Binder. Smart contracts search engine in blockchain. In *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, ICFNDS '19, New York, NY, USA, 2019. Association for Computing Machinery.

[206] Shuai Wang, Yong Yuan, Xiao Wang, Juanjuan Li, Rui Qin, and Fei-Yue Wang. An overview of smart contract: Architecture, applications, and future trends. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 108–113, 2018.

[207] A. Raza, K. Han, and S. O. Hwang. A framework for privacy preserving, distributed search engine using topology of dlt and onion routing. *IEEE Access*, 8:43001–43012, 2020.

[208] Sebastian Henningsen, Sebastian Rust, Martin Florian, and Björn Scheuermann. Crawling the ipfs network. In *2020 IFIP Networking Conference (Networking)*, pages 679–680, 2020.

[209] Almonit. Almonit. `https://almonit.eth.link/`, 2021.

[210] Vijay S. Mookerjee and Yong Tan. Analysis of a least recently used cache management policy for web browsers. *Operations Research*, 50(2):345–357, 2002.

[211] Shuai Wang, Wenwen Ding, Juanjuan Li, Yong Yuan, Liwei Ouyang, and Fei-Yue Wang. Decentralized autonomous organizations: Concept, model, and applications. *IEEE Transactions on Computational Social Systems*, 6(5):870–878, 2019.

[212] Satoshi Fujita. Similarity search in interplanetary file system with the aid of locality sensitive hash. *IEICE TRANSACTIONS on Information and Systems*, 104(10):1616–1623, 2021.

[213] Aameek Singh, Mudhakar Srivatsa, Ling Liu, and Todd Miller. Apoidea: A decentralized peer-to-peer architecture for crawling the world wide web. In *Distributed Multimedia Information Retrieval*, 2003.

[214] Accelerating content routing with bitswap: A multi-path file transfer protocol in ipfs and filecoin. 2021.

[215] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Biersack. A global view of kad. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, page 117–122, New York, NY, USA, 2007. Association for Computing Machinery.

[216] Nebulas Research. Yellow paper: Nebulas rank. `https://github.com/nebulasio/nr-report/blob/master/en/main.pdf`, 2019.

[217] Qian Li, Tao Zhou, Linyuan Lv, and Duanbing Chen. Identifying influential spreaders by weighted leaderrank, 2013.

[218] Torsten Suel, Chandan Mathur, Jo-wen Wu, Jiangong Zhang, Alex Delis, Mehdi Kharrazi, Xiaohui Long, and Kulesh Shanmugasundaram. Odissea: A peer-to-peer architecture for scalable web search and information retrieval. 06 2003.

[219] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, page 21, USA, 2004. USENIX Association.

[220] Ziliang Lai, Chris Liu, Eric Lo, Ben Kao, and Siu-Ming Yiu. Decentralized search on decentralized web. *CoRR*, abs/1809.00939, 2018.

[221] Sébastien Lahaie, David Pennock, Amin Saberi, and Rakesh Vohra. Sponsored search auctions. *Algorithmic Game Theory*, 01 2007.

[222] Tao Qin, Wei Chen, and Tie-Yan Liu. Sponsored search auctions: Recent advances and future directions. *ACM Trans. Intell. Syst. Technol.*, 5(4), jan 2015.

[223] Alex Berke and Dan Calacci. Privacy limitations of interest-based advertising on the web: A post-mortem empirical analysis of google's floc. *arXiv preprint arXiv:2201.13402*, 2022.

[224] Marc Langheinrich. To floc or not? *IEEE Pervasive Computing*, 20(2):4–6, 2021.

[225] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. A survey on federated learning systems: Vision, hype and reality for data privacy and protection, 2021.

[226] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2021.

[227] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. Obliviad: Provably secure and practical online behavioral advertising. In *2012 IEEE Symposium on Security and Privacy*, pages 257–271, 2012.

[228] Saikat Guha, Bin Cheng, and Paul Francis. Privad: Practical privacy in online advertising. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, Boston, MA, March 2011. USENIX Association.

[229] Imdad Ullah, Salil S. Kanhere, and Roksana Boreli. Privacy-preserving targeted mobile advertising: A blockchain-based framework for mobile ads. *CoRR*, abs/2008.10479, 2020.

[230] Matti Pärssinen, Mikko Kotila, Rubén Cuevas Rumin, Amit Phansalkar, and Jukka Manner. Is blockchain ready to revolutionize online advertising? *IEEE Access*, 6:54884–54899, 2018.

[231] Dongxiao Liu, Cheng Huang, Jianbing Ni, Xiaodong Lin, and Xuemin Shen. Blockchain-based smart advertising network with privacy-preserving accountability. *IEEE Transactions on Network Science and Engineering*, 8(3):2118–2130, 2021.

[232] Dominic Deuber, Nico Döttling, Bernardo Magri, Giulio Malavolta, and Sri Aravinda Krishnan Thyagarajan. Minting mechanisms for blockchain – or – moving from cryptoassets to cryptocurrencies. Cryptology ePrint Archive, Report 2018/1110, 2018. `https://ia.cr/2018/1110`.

[233] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. `http://arxiv.org/abs/1710.09437`, 2017.

[234] SteemIt. `https://steemit.com/`, 2022.

[235] Paul V Mockapetris. RFC 1035: Domain names-implementation and specification, 1987.

[236] Jonathan Weinberg. Icann and the problem of legitimacy. *Duke LJ*, 50:187, 2000.

[237] Vasileios Pappas, Dan Massey, and Lixia Zhang. Enhancing dns resilience against denial of service attacks. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pages 450–459, 2007.

[238] Pratik Satam, H Alipour, Youssif Al-Nashif, and Salim Hariri. Anomaly behavior analysis of dns protocol. *J. Internet Serv. Inf. Secur*, 5, 01 2015.

[239] U. Steinhoff, Alexander Wiesmaier, and Roberto Araújo. The state of the art in dns spoofing, 2006.

[240] Maciej Korczyński, Michał Król, and Michel van Eeten. Zone poisoning: The how and where of non-secure dns dynamic updates. In *Proceedings of the 2016 Internet Measurement Conference*, pages 271–278, 2016.

[241] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose. Dns security introduction and requirements. RFC 4033, RFC Editor, March 2005. `http://www.rfc-editor.org/rfc/rfc4033.txt`.

[242] Wilson Lian, Eric Rescorla, Hovav Shacham, and Stefan Savage. Measuring the practical impact of DNSSEC deployment. In *22nd USENIX Security Symposium (USENIX Security 13)*, pages 573–588, Washington, D.C., August 2013. USENIX Association.

[243] Marvin Theimer and Michael B. Jones. Overlook: scalable name service on an overlay network. In *Proceedings 22nd International Conference on Distributed Computing Systems*, pages 52–61, 2002.

[244] Russ Cox, Athicha Muthitacharoen, and Robert Morris. Serving dns using a peer-to-peer lookup service. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, page 155–165, Berlin, Heidelberg, 2002. Springer-Verlag.

[245] Marwan Abu-Amara, Farag Azzedin, Fahd A. Abdulhameed, Ashraf Mahmoud, and Mohammed H. Sqalli. Dynamic peer-to-peer (p2p) solution to counter malicious higher domain name system (dns) nameservers. In *2011 24th Canadian Conference on Electrical and Computer Engineering(CCECE)*, pages 001014–001018, 2011.

[246] Mark Handley and Adam Greenhalgh. The case for pushing dns. 01 2005.

[247] Xiangui Wang, Kedan Li, Hui Li, Yinghui Li, and Zhiwei Liang. Consortiumdns: A distributed domain name service based on consortium chain. In *2017 IEEE 19th International Conference on High Performance Computing and Communications; IEEE 15th International Conference on Smart*

City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pages 617–620, 2017.

[248] Xinan Duan, Zhiwei Yan, Guanggang Geng, and Baoping Yan. Dnsledger: Decentralized and distributed name resolution for ubiquitous iot. In *2018 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–3, 2018.

[249] Wenfeng Liu, Yu Zhang, Lu Liu, Shuyan Liu, Hongli Zhang, and Binxing Fang. A secure domain name resolution and management architecture based on blockchain. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7, 2020.

[250] NameCoin. Namecoin. `https://www.namecoin.org/`, 2011.

[251] Nxt Community. Nxt whitepaper. `https://nxtdocs.jelurida.com/Nxt_Whitepaper`, 2016.

[252] Emercoin. Emerdns. `https://emercoin.com/en/emerdns`, 2021.

[253] Handshake. Handshake whitepaper. https://handshake.org/files/handshake.txt, 2018.

[254] Namebase. Namebase. `https://learn.namebase.io`, 2021.

[255] Shi Yin, Yu Teng, Ning Hu, and Xu Dong Jia. Decentralization of dns: Old problems and new challenges. In *Proceedings of the 2020 International Conference on Cyberspace Innovation of Advanced Technologies*, CIAT 2020, page 335–341, New York, NY, USA, 2020. Association for Computing Machinery.

[256] HU Wei-hong, AO Meng, SHI Lin, XIE Jia-gui, and LIU Yang. Review of blockchain-based dns alternatives. 3(3):71, 2017.

[257] Scarlett Gourley and Hitesh Tewari. Blockchain backed dnssec. In *Lecture Notes in Business Information Processing*. Springer, 07 2018.

[258] Brendan Benshoof, Andrew Rosen, Anu G. Bourgeois, and Robert W. Harrison. Distributed decentralized domain name service. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1279–1287, 2016.

[259] Wondeuk Yoon, Indal Choi, and Daeyoung Kim. Blockons: Blockchain based object name service. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 219–226, 2019.

[260] Jingqiang Liu, Bin Li, Lizhang Chen, Meng Hou, Feiran Xiang, and Peijun Wang. A data storage method based on blockchain for decentralization dns. In *2018 IEEE Third International Conference on Data Science in Cyberspace (DSC)*, pages 189–196, 2018.

[261] Enis Karaarslan and Eylul Adiguzel. Blockchain based dns and pki solutions. *IEEE Communications Standards Magazine*, 2(3):52–57, 2018.

[262] Faizan Safdar Ali and Alptekin Küpçü. Improving pki, bgp, and DNS using blockchain: A systematic review. *CoRR*, abs/2001.00747, 2020.

[263] Unstoppable Domains. Architecture overview. `https://docs.unstoppabledomains.com/domain-registry-essentials/architecture-overview`, 2020.

[264] Philip Saunders. Nebulis. `https://medium.com/@Physes/how-to-build-a-more-secure-domain-system-5efed811409e`, 2016.

[265] Ethereum Name Service. Ens documentation. `https://docs.ens.domains`, 2021.

[266] Pengcheng Xia, Haoyu Wang, Zhou Yu, Xinyu Liu, Xiapu Luo, Guoai Xu, and Gareth Tyson. Challenges in decentralized name management: The case of ens. In *Proceedings of the 22nd ACM Internet Measurement Conference*,

IMC '22, page 65–82, New York, NY, USA, 2022. Association for Computing Machinery.

[267] Muneeb Ali. Stacks 2.0: Apps and smart contracts for bitcoin. `https://www.stacks.co`, 2020.

[268] Stacks. Blockchain naming system. `https://docs.stacks.co/build-apps/references/bns`, 2021.

[269] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J Freedman. Blockstack: A global naming and storage system secured by blockchains. In *2016 {USENIX} Annual Technical Conference ({USENIX} {ATC} 16)*, pages 181–194, 2016.

[270] Constantinos Patsakis, Fran Casino, Nikolaos Lykousas, and Vasilios Katos. Unravelling ariadne's thread: Exploring the threats of decentralised dns. *IEEE Access*, 8:118559–118571, 2020.

[271] Yuwei Zeng, Zang Tianning, Yongzheng Zhang, Xunxun Chen, and Yipeng Wang. A comprehensive measurement study of domain-squatting abuse. pages 1–6, 05 2019.

[272] Harry A. Kalodner, Miles Carlsten, Paul Ellenbogen, Joseph Bonneau, and Arvind Narayanan. An empirical study of namecoin and lessons for decentralized namespace design. In *14th Annual Workshop on the Economics of Information Security, WEIS 2015, Delft, The Netherlands, 22-23 June, 2015*, 2015.

[273] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

[274] Malte Möser, Ittay Eyal, and Emin Gün Sirer. Bitcoin covenants. In Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 126–141, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[275] Bryan Ford, Jacob Strauss, Chris Lesniewski-Laas, Sean Rhea, Frans Kaashoek, and Robert Morris. Persistent personal names for globally connected mobile devices. In *3rd USENIX Workshop on Real, Large Distributed Systems (WORLDS 06)*, Seattle, WA, November 2006. USENIX Association.

[276] Jakub Czyz, Mark Allman, Jing Zhang, Scott Iekel-Johnson, Eric Osterweil, and Michael Bailey. Measuring ipv6 adoption. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, page 87–98, New York, NY, USA, 2014. Association for Computing Machinery.

[277] Stephen Herwig, Christina Garman, and Dave Levin. Achieving keyless cdns with conclaves. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 735–751, 2020.

[278] Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. The bittorrent p2p file-sharing system: Measurements and analysis. In *Proceedings of the 4th International Conference on Peer-to-Peer Systems*, IPTPS'05, page 205–216, Berlin, Heidelberg, 2005. Springer-Verlag.

[279] Alexandre M. Mateus and Jon M. Peha. Quantifying global transfers of copyrighted content using bittorrent (september 24, 2011). In *TPRC 2011 - The 39th Research Conference on Communication, Information and Internet Policy*, 2011.

[280] Jingting Xue, Chunxiang Xu, and Lanhua Bai. Dstore: A distributed system for outsourced data storage and retrieval. *Future Generation Computer Systems*, 99:106–114, 2019.

[281] Wei Liang, Yongkai Fan, Kuan-Ching Li, Dafang Zhang, and Jean-Luc Gaudiot. Secure data storage and recovery in industrial blockchain network environments. *IEEE Transactions on Industrial Informatics*, 16(10):6543–6552, 2020.

[282] Sarven Capadisli, Tim Berners-Lee, Ruben Verborgh, Kjetil Kjernsmo, Justin Bingham, Dmitri Zagidulin, and Aaron Coburn. Solid protocol draft version 0.9.0. `https://solidproject.org/TR/protocol`, 2021.

[283] Tomas Isdal, Michael Piatek, Arvind Krishnamurthy, and Thomas Anderson. Privacy-preserving p2p data sharing with oneswarm. *ACM SIGCOMM Computer Communication Review*, 40(4):111–122, 2010.

[284] Miti Mazmudar, Stan Gurtler, and Ian Goldberg. Do you feel a chill? using pir against chilling effects for censorship-resistant publishing. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, pages 53–57, 2021.

[285] Eugene Y Vasserman, Victor Heorhiadi, Nicholas Hopper, and Yongdae Kim. One-way indexing for plausible deniability in censorship resistant storage. In *FOCI*, 2012.

[286] Nick Lambert and Benjamin Bollen. The safe network: a new, decentralised internet. 2014.

[287] Guillaume Michel. Double-hashing as a way to increase reader privacy. `https://www.youtube.com/watch?v=VBlx-VvIZqU`, 2022.

[288] Bram Cohen. Incentives build robustness in bittorrent. `https://www.bittorrent.org/bittorrentecon.pdf`, 2003.

[289] Landon P Cox and Brian D Noble. Samsara: Honor among thieves in peer-to-peer storage. *ACM SIGOPS Operating Systems Review*, 37(5):120–132, 2003.

[290] Sam Williams, Viktor Diordiiev, Lev Berman, India Raybould, and Ivan Uemlianin. Arweave: A protocol for economically sustainable information permanence. 2019.

[291] Michael Backes, Ian Goldberg, Aniket Kate, and Tomas Toft. Adding query privacy to robust dhts. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 30–31, 2012.

[292] Dongsheng Li, Wanxin Zhang, Siqi Shen, and Yiming Zhang. Ses-lsh: Shuffle-efficient locality sensitive hashing for distributed similarity search. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 822–827, 2017.

[293] Yu Hua, Bin Xiao, Dan Feng, and Bo Yu. Bounded lsh for similarity search in peer-to-peer file systems. In *2008 37th International Conference on Parallel Processing*, pages 644–651, 2008.

[294] I. Bhattacharya, S.R. Kashyap, and S. Parthasarathy. Similarity searching in peer-to-peer databases. In *25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pages 329–338, 2005.

[295] Alexander Smirnov and Andrew Ponomarev. A hybrid peer-to-peer recommendation system architecture based on locality-sensitive hashing. In *Proceedings of 15th Conference of Open Innovations Association FRUCT*, pages 119–125, 2014.

[296] Parisa Haghani, Sebastian Michel, and Karl Aberer. Distributed similarity search in high dimensions using locality sensitive hashing. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 744–755, 2009.

[297] Bahman Bahmani, Ashish Goel, and Rajendra Shinde. Efficient distributed locality sensitive hashing. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 2174–2178, New York, NY, USA, 2012. Association for Computing Machinery.

[298] Rodolfo da Silva Villaca, Luciano Bernardes de Paula, Rafael Pasquini, and Maurício Ferreira Magalhaes. Hamming dht: Taming the similarity search.

In *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*, pages 7–12, 2013.

[299] Bo Yuan, Jiahui Peng, Chunpei Li, and Wangjie Qiu. Ppirb:achieving an privacy-preserving image retrieval scheme based on blockchain. In *2022 5th International Conference on Data Science and Information Technology (DSIT)*, pages 1–6, 2022.

[300] Yiannis Psaras and Jorge M. Soares. Decentralising the internet with ipfs and filecoin (di2f) — a report from the trenches. Technical report, Protocol Labs, 2021.

[301] Omid Jafari, Preeti Maurya, Parth Nagarkar, Khandker Mushfiqul Islam, and Chidambaram Crushev. A survey on locality sensitive hashing algorithms and their applications. *CoRR*, abs/2102.08942, 2021.

[302] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, page 47–57, New York, NY, USA, 1984. Association for Computing Machinery.

[303] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, page 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[304] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, page 380–388, New York, NY, USA, 2002. Association for Computing Machinery.

[305] Qin Lv, William Josephson, Zhe Wang, Moses Charikar, and Kai Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, page 950–961. VLDB Endowment, 2007.

[306] Mayank Bawa, Tyson Condie, and Prasanna Ganesan. Lsh forest: Self-tuning indexes for similarity search. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, page 651–660, New York, NY, USA, 2005. Association for Computing Machinery.

[307] Andrei Broder. On the resemblance and containment of documents. *Proceedings of the International Conference on Compression and Complexity of Sequences*, 06 1997.

[308] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. Hashing for similarity search: A survey. *CoRR*, abs/1408.2927, 2014.

[309] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Finding Similar Items*, page 68–122. Cambridge University Press, 2 edition, 2014.

[310] Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. ICKS '08, page 131–136, USA, 2008. IEEE Computer Society.

[311] Rui Yuan, Yubin Xia, Haibo Chen, Binyu Zang, and Jan Xie. Shadoweth: Private smart contract on public blockchain. *J. Comput. Sci. Technol.*, 33(3):542–556, 2018.

[312] Mustafa Al-Bassam, Alberto Sonnino, Michal Król, and Ioannis Psaras. Airtnt: Fair exchange payment for outsourced secure enclave computations. *CoRR*, abs/1805.06411, 2018.

[313] Michal Król and Ioannis Psaras. SPOC: secure payments for outsourced computations. *CoRR*, abs/1807.06462, 2018.

[314] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah M. Johnson, Ari Juels, Andrew Miller, and Dawn Song. Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contract execution. *CoRR*, abs/1804.05141, 2018.

[315] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 839–858, 2016.

[316] Hung Dang, Dat Le Tien, and Ee-Chien Chang. Fair marketplace for secure outsourced computations. *CoRR*, abs/1808.09682, 2018.

[317] Alberto Sonnino, Michal Król, Argyrios G. Tasiopoulos, and Ioannis Psaras. Asterisk: Auction-based shared economy resolution system for blockchain. *CoRR*, abs/1901.07824, 2019.

[318] B. Enkhtaivan, T. Takenouchi, and K. Sako. A fair anonymous auction scheme utilizing trusted hardware and blockchain. In *2019 17th International Conference on Privacy, Security and Trust (PST)*, pages 1–5, 2019.

[319] H. Desai, M. Kantarcioglu, and L. Kagal. A hybrid blockchain architecture for privacy-enabled and accountable auctions. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 34–43, 2019.

[320] Hisham S. Galal and A. Youssef. Verifiable sealed-bid auction on the ethereum blockchain. In *IACR Cryptol. ePrint Arch.*, 2018.

[321] Hisham Galal and Amr Youssef. *Succinctly Verifiable Sealed-Bid Auction Smart Contract: ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings*, pages 3–19. 09 2018.

[322] Hisham S. Galal and Amr M. Youssef. Trustee: Full privacy preserving vickrey auction on top of ethereum. *CoRR*, abs/1905.06280, 2019.

[323] Michał Król, Alberto Sonnino, Argyrios Tasiopoulos, Ioannis Psaras, and Etienne Rivière. Pastrami: Privacy-preserving, auditable, scalable & trustworthy auctions for multiple items, 2020.

[324] Debasis Mishra and David C. Parkes. Multi-item vickrey-dutch auctions. *Games and Economic Behavior*, 66(1):326 – 347, 2009.

[325] Fabrice Benhamouda, Shai Halevi, and Tzipora Halevi. Supporting private data on hyperledger fabric with secure multiparty computation. *IBM J. Res. Dev.*, 63(2–3), March 2019.

[326] David Cerezo Sánchez. Raziel: Private and verifiable smart contracts on blockchains. *CoRR*, abs/1807.09484, 2018.

[327] V. Costan and S. Devadas. Intel sgx explained. *IACR Cryptol. ePrint Arch.*, 2016:86, 2016.

[328] Johannes Winter. Trusted computing building blocks for embedded linux-based arm trustzone platforms. STC '08, page 21–30, New York, NY, USA, 2008. Association for Computing Machinery.

[329] Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Trusted execution environments on mobile devices. In *2013 ACM SIGSAC conference on Computer & communications security, 4-8 November 2013, Berlin, Germany*, pages 1497–1498. ACM, 2013.

[330] Software guard extensions programming reference, revision 2. `https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf`, 2014.

[331] Zijian Bao, Qinghao Wang, Wenbo Shi, Lei Wang, Hong Lei, and Bangdao Chen. When blockchain meets sgx: An overview, challenges, and open issues. *IEEE Access*, 8:170404–170420, 2020.

[332] Seongmin Kim, Juhyeng Han, Jaehyeong Ha, Taesoo Kim, and Dongsu Han. Sgx-tor: A secure and practical tor anonymity network with sgx enclaves. *IEEE/ACM Transactions on Networking*, 26(05):2174–2187, sep 2018.

[333] David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[334] Alvin E. Roth and Elliott Peranson. The redesign of the matching market for american physicians: Some engineering aspects of economic design. *American Economic Review*, 89(4):748–780, September 1999.

[335] Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.

[336] Bryan Ford, Pyda Srisuresh, and Dan Kegel. Peer-to-peer communication across network address translators. *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, 2005.

[337] Yangyang Liu and Jianping Pan. The impact of nat on bittorrent-like p2p systems. *2009 IEEE Ninth International Conference on Peer-to-Peer Computing*, pages 242–251, 2009.

[338] Pyda Srisuresh, Bryan Ford, and Dan Kegel. State of peer-to-peer (p2p) communication across network address translators (nats). RFC 5128, March 2008.

[339] Henning Schulzrinne, Enrico Marocco, and Emil Ivov. Security issues and solutions in peer-to-peer systems for realtime communications. RFC 5765, February 2010.

[340] Ari Keranen, Christer Holmberg, and Jonathan Rosenberg. Interactive connectivity establishment (ice): A protocol for network address translator (nat) traversal. RFC 8445, July 2018.

[341] Prateesh Goyal, Ravi Netravali, Mohammad Alizadeh, and Hari Balakrishnan. Secure incentivization for decentralized content delivery. *CoRR*, abs/1808.00826, 2018.

[342] William B. Norton and Jonas Simanavicius. A blockchain-backed internet segment routing wan (sr-wan). Technical report, NOIA Network, 2019.

[343] Junchen Jiang, Rajdeep Das, Ganesh Ananthanarayanan, Philip A. Chou, Venkata Padmanabhan, Vyas Sekar, Esbjorn Dominique, Marcin Goliszewski, Dalibor Kukoleca, Renat Vafin, and Hui Zhang. Via: Improving internet telephony call quality using predictive relay selection. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, page 286–299, New York, NY, USA, 2016. Association for Computing Machinery.

[344] libp2p: Concepts. `https://docs.libp2p.io/concepts/`.

[345] Eleni Mykoniati, Lawrence Latif, Raul Landa, Ben Yang, Richard Clegg, David Griffin, and Miguel Rio. Distributed overlay anycast tables using space filling curves. In *Proceedings of the 28th IEEE International Conference on Computer Communications Workshops, INFOCOM'09*, pages 19–24, 2009.

[346] Alberto Montresor and Márk Jelasity. PeerSim: A scalable P2P simulator. In *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, pages 99–100, Seattle, WA, September 2009.

[347] Srinivasan Keshav. Paradoxes of internet architecture. *IEEE Internet Computing*, 22:96–102, 01 2018.

[348] Gaurish Korpal and Drew Scott. Decentralization and web3 technologies. 2022.

[349] Qin Wang, Rujia Li, Qi Wang, Shiping Chen, Mark Ryan, and Thomas Hardjono. Exploring web3 from the view of blockchain, 2022.

[350] Ghada Almashaqbeh and Ravital Solomon. Sok: Privacy-preserving computing in the blockchain era. Cryptology ePrint Archive, Paper 2021/727, 2021. `https://eprint.iacr.org/2021/727`.

[351] Ghada Almashaqbeh. *CacheCash: a cryptocurrency-based decentralized content delivery network.* Columbia University, 2019.

[352] Barjini Barjini, Mohamed Othman, Hamidah Ibrahim, and Nur Udzir. Shortcoming, problems and analytical comparison for flooding-based search techniques in unstructured p2p networks. *Peer-to-Peer Networking and Applications*, 5:1–13, 03 2012.

[353] Kose John, Maureen O'Hara, and Fahad Saleh. Bitcoin and beyond. *Annual Review of Financial Economics*, 14:95–115, 2022.

[354] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. An empirical analysis of traceability in the monero blockchain, 2018.

[355] Carl Worley and Anthony Skjellum. Blockchain tradeoffs and challenges for current and emerging applications: Generalization, fragmentation, sidechains, and scalability. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1582–1587, 2018.

[356] Darko Čapko, Srdan Vukmirović, and Nemanja Nedić. State of the art of zero-knowledge proofs in blockchain. In *2022 30th Telecommunications Forum (TELFOR)*, pages 1–4, 2022.

[357] Barbara Guidi, Andrea Michienzi, and Laura Ricci. Evaluating the decentralisation of filecoin. In *Proceedings of the 3rd International Workshop on Distributed Infrastructure for the Common Good*, DICG '22, page 13–18, New York, NY, USA, 2022. Association for Computing Machinery.

[358] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4):460–473, 2007.

[359] Michael Herrmann, Kai-Chun Ning, Claudia Díaz, and Bart Preneel. Description of the yacy distributed web search engine. 2014.

[360] Alesha Serada, Jori Grym, and Tanja Sihvonen. The economy of attention on blockchain in the brave browser. In *Futures of Journalism: Technology-stimulated Evolution in the Audience-News Media Relationship*, pages 49–62. Springer, 2022.

[361] Qiuhong Zheng, Yi Li, Ping Chen, and Xinghua Dong. An innovative ipfs-based storage model for blockchain. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 704–708, 2018.

[362] Jack Goldsmith. Who controls the internet? illusions of a borderless world. *Strategic Direction*, 23(11), 2007.

[363] Erik-Oliver Blass and Florian Kerschbaum. *Strain: A Secure Auction for Blockchains: 23rd European Symposium on Research in Computer Security, ESORICS 2018, Barcelona, Spain, September 3-7, 2018, Proceedings, Part I*, pages 87–110. 08 2018.