Governing Decentralized Complex Queries Through a DAO

Governing Decentralized Complex Queries Through a DAO

Mirko Zichichi
Ontology Engineering Group
Universidad Politécnica de Madrid, Spain
mirko.zichichi@upm.es

Stefano Ferretti
Department of Pure and Applied Sciences
University of Urbino "Carlo Bo", Italy
stefano.ferretti@uniurb.it

Luca Serena
CIRI ICT
University of Bologna, Italy
luca.serena2@unibo.it

Gabriele D'Angelo
Department of Computer Science and Engineering
University of Bologna, Italy
g.dangelo@unibo.it

ABSTRACT

Recently, a new generation of P2P systems capable of addressing data integrity and authenticity has emerged for the development of new applications for a "more" decentralized Internet, i.e., Distributed Ledger Technologies (DLT) and Decentralized File Systems (DFS). However, these technologies still have some unanswered issues, mostly related to data lookup and discovery. In this paper, first, we propose a Distributed Hash Table (DHT) system that efficiently manages decentralized keyword-based queries executed on data stored in DFS. Through a hypercube logical layout, queries are efficiently routed among the network, where each node is responsible for a specific keywords set and the related contents. Second, we provide a framework for the governance of the above network, based on a Decentralized Autonomous Organization (DAO) implementation. We show how the use of smart contracts enables organizational decision making and rewards for nodes that have actively contributed to the DHT. Finally, we provide experimental validation of an implementation of our proposal, where the execution of the same protocol for different logical nodes of the hypercube allows us to evaluate the efficiency of communication within the network.

CCS CONCEPTS

• **Computer systems organization** → **Distributed architectures**; • **Information systems** → **Distributed storage**.

KEYWORDS

Distributed Ledger Technology, Decentralized File Storage, Distributed Hash Table, Keyword Search, Smart Contracts

1 INTRODUCTION

The digitalization process, that has been ongoing over the last decades, has seen data management and delivery become a crucial issue. In order to cope with the increasingly higher number of contents that is demanded through the Web, multiple solutions for an efficient use of Internet have been designed. In particular, thanks to the decentralization of content storage and delivery, it is possible to avoid single points of failures, reduce the workload at data centers and to allow a distribution of data that is closer to the original source. Decentralization also fosters the creation of open systems, where participants can freely join the system and contribute to its functioning.

Recently, Distributed Ledger Technologies (DLTs) and Decentralized File Systems (DFS) have emerged as Peer-to-Peer (P2P) technologies capable of offering interesting features related to data validation and trustfulness [23]. DLTs have gained popularity with the advent of the cryptocurrencies, which allow users to trade crypto-assets without any central entity being involved, ensuring transparency and data integrity. Besides the financial use case, DLTs, and DFS in particular, provide the features of data integrity, authenticity, confidentiality and auditability, used to build novel applications for a "more" decentralized Internet [1].

InterPlanetary File System (IPFS) is one of the most used DFS protocols where files and data are replicated globally on hundreds of nodes in the network [2]. Furthermore, DFS and in general other P2P-based technologies might also have a prominent role against censorship, since shutting down a server will not prevent contents from being available on Internet. An example occurred when Turkey denied the access to the Turkish Wikipedia in 2017, with IPFS being able to guarantee the access through mirroring [17].

One of the aspects that remains still open with respect to these novel technologies, is concerned with the data discovery and lookup. Specifically, data can only be accessed by knowing the respective identifier or location and cannot be searched based on its content. Put in other words, these systems lack a viable (decentralized) data management scheme that enables "complex" queries on top of them.

In this paper, we propose a decentralized system to efficiently manage keyword-based queries to the contents stored in DFS. We make use of a Distributed Hash Table (DHT) structured as a hypercube in order to provide the service of keyword-based queries over IPFS files. The hypercube is a logical layout where there are 2^r nodes, each one labelled with an r bits ID and connected to the r nodes whose ID differs by only one bit. Each node is responsible for a specific keywords set, derived from their ID. The hypercube structure allows to optimize the routing of the queries, by reducing the number of hops needed to locate contents. Moreover, the second main contribution of this paper is the creation of a framework for the organization of nodes operators that host and share information, with the aim of improving the scalability and the decentralization of the system. In fact, usually applications built upon P2P networks are supported by nodes that have no particular incentive to keep them operational, but are only interested in their use, e.g. BitTorrent. In our work, we focus on the case where nodes are interested in keeping the network operational and healthy, and, in general, where node operators act in the context of a sharing economy, e.g. in the same way as Wikipedia editors are interested in contributing to the free encyclopedia [9]. It has been argued that DLTs can "crystallize" the dynamics of a model of socio-economic production in which large numbers of people work cooperatively, i.e. commons-based peer production [15]. P2P networks have this "cooperative vein" intrinsically built into their structure. Therefore, we leverage DLTs to build a Decentralized Autonomous Organization (DAO) [11] around the network of node operators. We envision an approach that is based on the creation of a DAO for those actors who have actively contributed to the functioning of the system, with smart contracts involved in managing rewards and organizational decisions. We then propose different use cases where this network can take form and we provide a possible framework for its governance.

Finally, we report an experimental validation of an implementation of our proposal. In particular, we provide results showing how the size of the hypercube and the number of objects stored in the DHT affect the search procedures. Moreover, we also evaluate the smart contracts, implemented to be executed on the Ethereum blockchain, in terms of operations execution cost.

This paper is structured as follows. Section 2 provides the background on the technologies used. Section 3 presents a description of the hypercube DHT structure, while in Section 4 we argue on how such a system can be governed through a DAO. In Section 5, the system experimental validation is reported, finally, Section 6 provides the concluding remarks.

2 BACKGROUND AND RELATED WORK

2.1 Distributed Hash Tables (DHTs)

A Distributed Hash Table (DHT) is a decentralized system for the distributed storage of contents. The rationale of this approach is to store the information in the various nodes of the system, providing a routing mechanism to easily get which node owns a certain resource [12]. Each local view of the DHT nodes will look like a traditional hash table, with a mapping from a key (i.e. the univocal representation of an item) to values (i.e. addresses of the peers owning such a resource). The association of objects to DHT nodes is obtained through the use of a hash function, a one-way function

which maps any item into a binary sequence of n bits. The idea is to distribute the storage workload among the DHT nodes according to the key (i.e. the n bit string obtained after having applied the hash function) of the objects. Each DHT is identified itself through an n bit ID, which lies in the same ID space used to identify contents. Then, based on its ID, each node is in charge of maintaining information on those contents that are in a specific ID space interval. Lookup of a content x thus becomes looking for the node in the DHT that manages a subset of the ID space that contains x [5].

2.2 Distributed Ledger Technology (DLT)

A DLT is a P2P system where the participants maintain a copy of the ledger, and there is a consensus mechanism that allows all the nodes to have the same view on the stored information. Data written on the ledger are trustworthy, because DLT protocols ensure their integrity, immutability and authenticity. There are multiple DLT implementations, differing mostly for their structure (e.g. blockchain, DAG) and for the features they provide, such as smart contracts.

2.2.1 Smart Contracts and Decentralized Autonomous Organizations. Smart contracts are programs whose execution is performed in a distributed way. In Ethereum [3], all the participants receive the same inputs and perform a computation on the basis of a smart contract code that leads to the same outputs. Each process is thus completely traced and permanently stored on the blockchain. Smart contracts can be used to automatize and supervise the exchange of digital or physical assets, to create tokens, i.e. the representation of physical assets or utilities, and to allow the management of a Decentralized Autonomous Organization (DAO) [11]. In order to enable a decentralized management of a DAO, smart contracts implement transactions, currency flows, rules and rights within the organization. DAO members can make proposals for the management of the organization and also discuss and vote those through transparent mechanisms. Members can also interact through smart contracts and tokens can be sent or received. Usually, tokens grant their holder a certain set of rights within the DAO.

2.3 Decentralized File Storage (DFS)

Decentralized File Storages (DFS) offer an alternative to the traditional client-server models, i.e. where a domain name is provided and is then resolved to an IP address. In Content Based Addressing items are directly queried through the network rather than establishing a connection with a server. In order to know which node in the network own the requested contents, it is possibile to rely on a DHT system that is in charge of mapping the items with the addresses of the peers owning such data. DFS follow this approach and offer higher data availability and resilience thanks to data replication.

2.3.1 IPFS and File Search. The InterPlanetary File System (IPFS) is a DFS and a protocol thought for distributed environments with a focus on data resilience [8]. The P2P network that runs the IPFS protocol, stores and shares files in the form of IPFS objects that are identified by a CID (Content IDentifier). This CID consists in the digest produced when a hash function is applied to a file and it is used to retrieve the referenced IPFS object. However, it provides

no means of searching for a file without owning it, since its hash is required. To overcome this limitation a generic search engine has been developed, namely "ipfs-search" [10]. This solution is rather centralised and does not escape the problem of concentration similar to the conventional web. In response to this, a decentralized solution called Siva [13] has been proposed. An inverted index of keywords is built for the published contents on IPFS and users can search through it, however Siva is proposed as an enhancement of the IPFS public network DHT and does not feature any optimization for a keyword storage structure apart from the use of caching. In terms of aim, a system, similar to the one presented in this paper, is The Graph, a "Decentralized Query Protocol" [18]. The Graph network consists of a system built upon Ethereum and IPFS, that allows to query data stored in these two technologies. The organization of the network is similar to what is referred as DAO, however their method for storing indexes is different from our proposal.

3 MULTIPLE KEYWORDS SEARCH

The hypercube geometric form has been leveraged by Joung et al. [12] to organise the topological structure of a DHT network, by using keywords. Such DHT can be exploited to perform multiple keyword based queries. Let $K \subseteq W$ represent a keywords set in a keyword space W. Such set K can be used to perform a query to search for data contents characterized by keywords contained in K (e.g. as metadata). In particular, let O be a set of data objects referenced in the DHT, these are distributed among all the network nodes based on the keywords they have been associated with, i.e. all the objects $o \in O$ mapped to a keywords set $K_o \subseteq W$ are maintained by the node responsible for such a keywords set K_o. We then consider O as the set of all the CIDs in IPFS and we use the DHT to map keywords to IPFS Objects.

3.1 Hypercube

The DHT takes the form of a r-dimensional hypercube $H_r(V, E)$, where V is set of vertices representing logical network nodes and E is a set of edges that connect pairs of neighbor nodes. The ID of a logical node is given by an r-bit string associated to the keywords set the node is responsible for. Each bit position refers to a specific keyword. Thus, let assume that a given keywords set K contains a given keyword k, which is assigned to the i-th bit of the string by a function. Then, the in the r-bit string representation of K the i-th bit will be set to 1. More formally, keywords are given in input to an uniform hash function $h : W \rightarrow \{0, 1, \ldots, r - 1\}$ that returns positions to set to 1 in r-bit string, i.e. $one(u) = \{h(k) \mid k \in K\}$.

We leverage such a protocol and make use of these r-bit strings to identify logical nodes in our system, e.g. for $r = 4$, a node that has ID 0100 handles all those the objects $o \in O$ associated to keywords sets whose the *one* function returns 0100. In the DHT, connections between nodes, i.e. edges, are created among those nodes whose IDs differ of only one bit, e.g. 1011 and 1010. This link creation method builds an hypercube structured graph.

3.2 Keywords Queries

In our system, the discovery of contents in IPFS is based on the lookup of multiple keywords stored in the DHT. In particular each

logical node will locally store an index table where the CIDs of all the IPFS Objects associated to the keywords set it is responsible for are stored. Given a query for a keywords set K, the associated r-bit string is used to reach the responsible logical node through a routing mechanism based on the hypercube form.

Take, for instance, a keyword space W made of 6 keywords, $W = \{$"Temperature", "PoI", "Wikipedia", "Bologna", "Urbino", "Rome"$\}$. Let consider a query with a keywords set $K = \{$"Wikipedia, Rome"$\}$ and assume that the r-bit query string associated to K will be 001001. Thus, to answer the query the node u with ID 001001 must be contacted. Starting from a node $v \in V$ in the hypercube, the query process consists in passing the request from v to one of its neighbours that is nearer to the destination u. Iterating this process, u will be eventually reached and it will return the CIDs associated to K, in this case the reference to the Wikipedia page of the city of Rome stored in IPFS. This type of punctual query is called Pin Search in [12], i.e. obtaining all and only the objects that are exactly associated with the keywords set K, $\{o \in O \mid K_o = K\}$; in this case data objects are retrieved only from one node.

Another query type is the Superset Search. It is similar to a Pin Search, but in addition it also searches for objects that can be described by keywords sets that include K, i.e., $\{o \in O \mid K_o \supseteq K\}$. In this case, data objects are retrieved from all nodes that are responsible for a superset of K. Then, since the possible outcomes of this search can be quite large, a limit l on the results is set. For instance, a Superset Search using the keywords set of the previous example would include:

- the objects retrieved from the node u through the Pin Search
- plus the objects retrieved from u's neighbors, responsible for keywords sets such as $K_1 = \{$"Wikipedia, Rome, PoI"$\}$
- plus the objects retrieved from u's neighbors' neighbors, with keywords sets such as $K_2 = \{$"Wikipedia, Rome, PoI,Temperature"$\}$
- and so on, until the number of objects is equal to l or no more nodes shall be contacted.

4 DECENTRALIZED AUTONOMOUS ORGANIZATION FRAMEWORK

The contribution presented so far in this paper can be represented as in Figure 1 with the first four layers (bottom-up). To recap:

(1) First layer: the nodes of the IPFS public network running the standard IPFS protocol.
(2) Second layer: all the files that the IPFS nodes keep in their storage. These are indexed using CIDs.
(3) Third layer: the files can be described by keywords, which are then used to execute queries and find the files.
(4) Fourth layer: these keywords are saved, together with the file association via the CID, using the Hypercube DHT.

Before continuing, it is important to point out that this structure is independent from the IPFS implementation, but can be adapted to any type of DFS or DLT for data storage, e.g. IOTA DLT [23].

The main focus of this section is the fifth layer. It consists of technologies and processes that form the governance of the hypercube DHT network, i.e. the DAO. Layers 1 to 4 provide the technological means for implementing a keyword based search over a DFS and this can be enough for offering a complete solution in many cases.
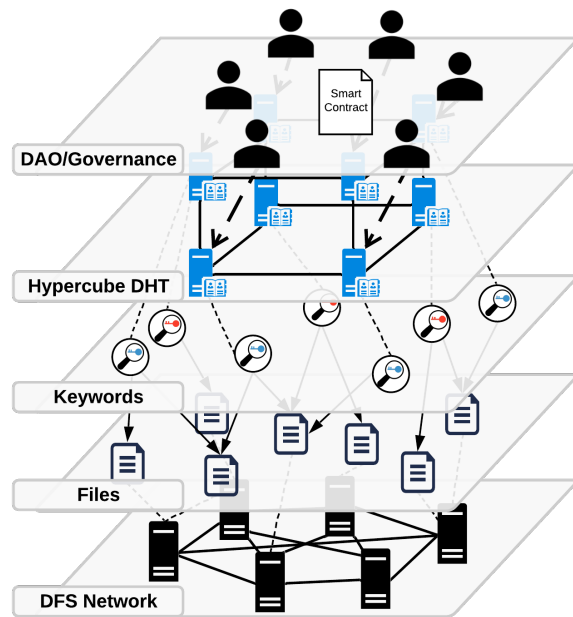
Figure 1: System represented as a layered architecture.

However, we are also interested in the scenario where, in order to orchestrate the operational decisions and rewards, the DHT network nodes operators can form a DAO. The fifth layer is mostly based on the use of smart contracts and the interfaces to those. Smart contracts, indeed, enable the creation of an organization that takes advantage of a token-based economy and decentralized voting. In particular, we use Ethereum smart contracts and we structure the DAO on top of two preliminary research contributions [4, 22]:

- **Token economy** - The DAO is built around the use of a unique token, e.g. "DAOToken", used for transferring value (e.g. users that pay node operators), or for staking purposes (e.g. becoming a DAO member by time-locking a certain amount of tokens). The smart contract used to represent these functions consists in an implementation of the ERC20 interface [6].
- **Members Registry** - A smart contract was developed as a members registry, to allow token holders to time-lock DAO-Tokens and become DAO members. Any account holding any amount of DAOToken can lock some tokens for a desired amount of time through a specific time-lock contract. This time-lock contract will hold these tokens and release them after the date set, and no one will be able to unlock those before that date.
- **General Voting** - A specific smart contract was developed to allow DAO members to call for a vote and then decide on a proposal. This contract allows any member to make a proposal and gives everyone the opportunity to submit a suggestion to vote regarding that proposal. Each proposal has its own debate period and any member can vote a suggestion within that time period. A member vote weight is proportional to the amount of tokens locked until a date that comes after the debate period end.

- **Value Transfer Voting** - Any extension of the previous voting smart contract can be developed to allow a decision taken to directly enact an operation to be executed on-chain (through another smart contract). For instance, DAO members can vote to transfer some staked tokens to a specific account in the case of issuing a bounty.

4.1 Use Cases

In this section we discuss on different, but possibly overlapping, use cases for implementing the DAO framework presented above. For instance, the first use case on DeFi is generally applicable to different DAOs implementations, and can be combined with the second use case to create a complete Hypercube DAO system.

4.1.1 A DeFi-based rewarding system. Decentralized finance (DeFi) is a term that refers to novel P2P financial infrastructures, based on smart contracts, that are non-custodial, permissionless, openly verifiable and composable [19]. With DeFi protocols such as Decentralized Exchanges (DEX), anyone can engage in non-custodial exchange of on-chain digital assets, e.g. tokens. In contrast to traditional finance where an asset's liquidity is based on the bid and ask orders prices, in the most used DeFi protocols, such as Uniswap, usually assets are ERC20 tokens and their liquidity is provided algorithmically through a simple pricing rule within a smart contract [19]. For instance, in the case of the DAOToken, an Uniswap liquidity pool smart contract can be created by locking into it an amount x of DAOTokens and an amount y of another ERC20 token to be exchanged with. The value of a single DAOToken in respect to the other token will be proportional to the ratio $\frac{y}{x}$ and such values will vary based on the tokens that will be stacked in the pool after an exchange, e.g. buying DAOTokens will drain the reserve of the locked DAOTokens and increase the other token's reserve.

In Uniswap, each liquidity provider receives newly minted Liquidity Pool (LP) tokens to represent the share of liquidity they have provided. These LP tokens can then be burn by the providers in order to redeem their share of liquidity (and accrued fees obtained when exchanges happen). This means that when a new ERC20 token, e.g. the DAOToken, is created and initially distributed to its creators, they can easily have a return on their newly created tokens by locking them in a new liquidity pool. This is also a way to let the general investors interested in this token to acquire it. However, the possibility for the creators to redeem at any time the liquidity they have provided, by burning the LP tokens, makes the value of the token highly unstable. At any moment, indeed, the investors can be left with a worthless token due to these "big players" burning LP tokens and draining the reserve.

Based on this, we envision a use case where the DAO is based not on the timelock of the DAOToken directly, but on the timelock of the LP tokens obtained by locking DAOTokens in liquidity pools. From an implementation point of view, in Uniswap no changes are required because LP Tokens are compatible with the ERC20 inteface. This means that the stability of the DAO is directly proportional to the value the DAOToken can take, and that the power exercised by DAO members is directly proportional to the gains/losses they are willing to make through their behavior, making it possible to have a strong incentive to behave correctly.

Smart Contract	Operation	Cost (gas)
ERC20	transfer()	51167
TokenTimelockProxy	lockTokens()	232024
TokenTimelock	release()	25626
Voting	submitProposal()	133501
Voting	submitSuggestion()	114523
Voting	vote()	142848
Voting	executeProposal()	56991

Table 1: DAO smart contracts operations cost in terms of gas.

4.1.2 Unique general purpose DAO vs. DAO islands. The proposal we provide in this work comprises the use case where a unique DHT network is governed by a DAO, with the purpose of assisting "browsable sister network[s] to the internet" [20]. Indeed, apart from IPFS, several DFS (and DLT-backed DFS such as Arweave), are built for the replication of the content that can be found on Internet. In this use case we propose a unique DAO that deals with the maintaining of a DHT that allows to search through keywords in those general purpose file storages.

Opposed to the previous one, we envision a use case where different networks implement their own DHT, resulting in a multitude of "islands" where keywords-based queries are possible for specific topics or platforms. Each island has its own organization and rules, but they are all similar for the querying protocol. It could be the case in which several smart contracts enabled DLTs are used for making the DAO, or, conversely, only one DLT used, with the same token shared among the different DAOs. An example would be a DAO maintaining the hypercube DHT for querying the decentralized version of Wikipedia, or a DAO for maintaining querying for political content shared in social media.

4.1.3 Decentralized ipfs-search. Finally, a possible use case is based on the possibility of combining ipfs-search [10] with our keyword-based hypercube DHT. That is, we consider a protocol where:

- several IPFS nodes crawl the network by monitoring the IPFS logs for files addition;
- these nodes download the files added through the "sniffed" CID;
- for each file the metadata are extracted and transformed into keywords;
- the association between the keywords obtained and the CID is stored in the hypercube DHT.

5 EXPERIMENTAL EVALUATION

In this section, we provide an experimental validation of our work. In particular, we implemented the software that each hypercube DHT logical node runs for maintaining the index table and to answer the queries that it receives. Furthermore, we developed a prototype of the DAO framework.

5.1 DAO Smart Contracts

The smart contracts that implement the framework presented above have been developed in Solidity and stored as Open Source code in Zenodo [21]. In Table 1, we provide the cost execution in terms of gas [3] for the main operations. The most expensive operation is the *lockTokens()* function, that locks a certain amount of an ERC20 Token for a specified amount of time. This is because, following the OpenZeppelin library for secure smart contracts development [14], each lock request creates a new smart contract that locks tokens for a unique account. However, normally the creation and deployment of such a smart contract through the Factory pattern would require at least 501818 gas units. We used the EIP-1167 Minimal Proxy pattern [16], that, instead of deploying a new contract each time such as in the Factory pattern, clones an already deployed contract functionalities by delegating all function calls to it.

5.2 Hypercube

The DHT software is implemented in Python and it exposes the four main nodes actions using the Flask server framework [7], i.e. Insert object, Remove object, Pin search, Superset search. Together with the core logic methods for a logical node, the implementation includes also an interface for communicating with an IPFS node, in order to possibly return files instead of only CIDs. In the same physical nodes, it is possible to host more than one logical node, that might refer to the same (local) IPFS node.

We tested our implementation running the software on a dedicated host (i.e. a quad core CPU, 16GB RAM), by associating several logical nodes to different operative system ports. Each logical node was executed by a dedicated Flask server and after a bootstrap phase, each node was connected to its neighbors based on the hypercube topology. More specifically, we run two different types of tests, one for the Pin Search and one for the Superset Search. In both cases, we tested the network configuration for 8, 16, 32, 64 and 128 nodes and populated the network each time with 10, 100 and 1000 objects generated randomly. Then, we repeated 50 random queries for Pin Search and 50 for Superset Search (with objects limit set to 10). During each query, a node was randomly chosen and it was queried using a random keywords set.

The results for the test that we carried out are reported in Figure 2. These results are obtained averaging the number of hops needed to get from on node in the network to another, for different operations.

5.2.1 Pin Search. Results for the Pin Search (Figure 2, left) show similar average hops when the number of objects varies and an increase from 1.28 to 3.52 when increasing the number of nodes. This was an expected result as the Pin Search average number of hops should theoretically be with the order of the logarithm of the number of logical nodes, i.e. $\frac{\log(n)}{2}$ or $\frac{r}{2}$. For instance, with 128 nodes, the experienced average number of hops was around $\frac{\log(128)}{2} = 3.5$.

5.2.2 Superset Search. In the case of Superset Search results are different from the previous case. As Figure 2 (right) shows, the average number of hops decreases when the number of objects increases, and it increases when the number of nodes increases. The minimum value here is 1.36 for 1000 objects and 8 nodes and the maximum is 20.36 for 10 objects and 128 nodes. Theoretically, the average number of hops should be equal to the average hops required to get to the node responsible for query keywords set K, i.e. Pin Search $\frac{\log(n)}{2}$, plus the average hops to get from that node to all the nodes that that include K, until the limit of objects l (or nodes including K) is reached.
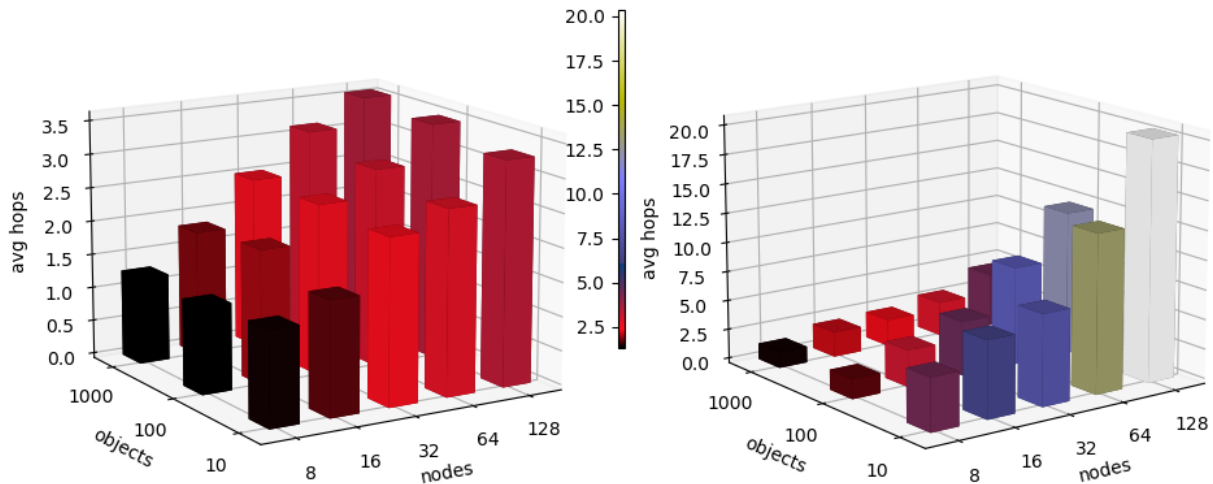
Figure 2: Pin Search results on the left, Superset Search on the right.

6 CONCLUSIONS

In this work, we proposed a decentralized system that manages keyword-based queries for contents stored in IPFS, through the use of an hypercube DHT. The query routing efficiency lies in the traversal of the hypercube which has a maximum number of hops of $\log(number\ of\ nodes) = r$, i.e. the hypercube dimension. Our experimental validation is in line with this number and shows that on average $\frac{r}{2}$ hops are required for the Pin Search. While, in the case of the Superset Search, we experienced the dependence of the number of hops with the ratio between the limit l assigned to the query and the distribution of objects between nodes.

Furthermore, we described the development of a DAO related to the economic sustainability and development of the project, as well as use cases for the government of the above system. The use of Ethereum smart contracts enables the possibility of voting for making organizational decisions. Furthermore, the ability to create ERC20 tokens allows to reward nodes that have actively contributed to the operation of the P2P system.

As a future work, we will focus on two aspects. Firstly, we will investigate on the feasibility of a "pay-per-query" model, where node operators within the DAO are rewarded at the level of granularity of the query. Secondly, we will face load balancing issues that arise when a more realistic contents distribution is put in place and where some nodes might suffer an higher workload due to the popularity of the contents/keywords they store.

REFERENCES

[1] Marianna Belotti, Nikola Božić, Guy Pujolle, and Stefano Secci. 2019. A vademecum on blockchain technologies: When, which, and how. *IEEE Communications Surveys & Tutorials* 21, 4 (2019), 3796–3838.
[2] Juan Benet. 2014. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561* (2014).
[3] Vitalik et al. Buterin. 2013. Ethereum white paper. https://github.com/ethereum/wiki/wiki/White-Paper
[4] Biagio Distefano, Nadia Pocher, and Mirko Zichichi. 2020. MOATcoin: Exploring Challenges and Legal Implications of Smart Contracts Through a Gamelike DApp Experiment. In *Proc. of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems (CryBlock 2020), co-located with the 26th Annual International Conference on Mobile Computing and Networking (MobiCom 2020), ACM.* ACM, 1–6.
[5] Gabriele D'Angelo and Stefano Ferretti. 2017. Highly intensive data dissemination in complex networks. *J. Parallel and Distrib. Comput.* 99 (2017), 28–50.
[6] Vitalik Buterin Fabian Vogelsteller. 2015. EIP-20: ERC-20 Token Standard. https://eips.ethereum.org/EIPS/eip-20
[7] Miguel Grinberg. 2018. *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.".
[8] Barbara Guidi, Andrea Michienzi, and Laura Ricci. 2021. Data Persistence in Decentralized Social Applications: The IPFS approach. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC).* IEEE, 1–4.
[9] Juho Hamari, Mimmi Sjöklint, and Antti Ukkonen. 2016. The sharing economy: Why people participate in collaborative consumption. *Journal of the association for information science and technology* 67, 9 (2016), 2047–2059.
[10] IPFS Community. 2021. Search engine for the InterPlanetary File System. https://github.com/ipfs-search/ipfs-search.
[11] Christoph Jentzsch. 2016. Decentralized autonomous organization to automate governance. *White paper, November* (2016). https://lawofthelevel.lexblogplatformthree.com/wp-content/uploads/sites/187/2017/07/WhitePaper-1.pdf
[12] Yuh-Jzer Joung, Li-Wei Yang, and Chien-Tse Fang. 2007. Keyword search in dht-based peer-to-peer networks. *IEEE Journal on Selected Areas in Communications* 25, 1 (2007), 46–61.
[13] Nawras Khudhur and Satoshi Fujita. 2019. Siva-The IPFS Search Engine. In *2019 Seventh International Symposium on Computing and Networking (CANDAR).* IEEE, 150–156.
[14] OpenZeppelin. 2021. OpenZeppelin website. https://openzeppelin.com/
[15] Alex Pazaitis, Primavera De Filippi, and Vasilis Kostakis. 2017. Blockchain and value systems in the sharing economy: The illustrative case of Backfeed. *Technological Forecasting and Social Change* 125 (2017), 105–115.
[16] Joe Messerman Peter Murray, Nate Welch. 2018. EIP-1167: Minimal Proxy Contract. https://eips.ethereum.org/EIPS/eip-1167
[17] João Santos, Nuno Santos, and David Dias. 2019. DClaims: A censorship resistant web annotations system using IPFS and Ethereum. *arXiv preprint arXiv:1912.03388* (2019).
[18] The Graph. 2020. The Graph Protocol. https://thegraph.com/
[19] Sam M Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J Knottenbelt. 2021. SoK: Decentralized Finance (DeFi). *arXiv preprint arXiv:2101.08778* (2021).
[20] Samuel Williams and William Jones. 2018. Arweave Lightpaper. (2018).
[21] Mirko Zichichi. 2021. *miker83z/HypercubeDAOContracts.* https://doi.org/10.5281/zenodo.4767755
[22] Mirko Zichichi, Michele Contu, Stefano Ferretti, and Gabriele D'Angelo. 2019. LikeStarter: a Smart-contract based Social DAO for Crowdfunding. In *Proc. of the 2st Workshop on Cryptocurrencies and Blockchains for Distributed Systems.*
[23] Mirko Zichichi, Stefano Ferretti, and Gabriele D'Angelo. 2020. A Framework based on Distributed Ledger Technologies for Data Management and Services in Intelligent Transportation Systems. *IEEE Access* (2020).