Received: 31 December 2021 | Revised: 8 February 2022 | Accepted: 15 March 2022

IET Networks

DOI: 10.1049/ntw2.12037

The Institution of Engineering and Technology WILEY

ORIGINAL RESEARCH

Complex queries over decentralised systems for geodata retrieval

Mirko Zichichi[1] | Luca Serena[2] | Stefano Ferretti[3] | Gabriele D'Angelo[2]

[1]Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain

[2]Department of Computer Science and Engineering, University of Bologna, Bologna, Italy

[3]Department of Pure and Applied Sciences, University of Urbino, Urbino, Italy

Correspondence

Mirko Zichichi, Departamento de Inteligencia Artificial (DIA), Universidad Politécnica de Madrid, ETS de Ingenieros Informáticos (ETSIINF), Campus de Montegancedo s/n, Boadilla del Monte (Comunidad de Madrid), 28660, Spain.
Email: mirko.zichichi@upm.es

Funding information

Università degli Studi di Urbino Carlo Bo, Grant/Award Number: Bit4Food; H2020 Marie Sklodowska-Curie Actions, Grant/Award Number: 814177

Abstract

Decentralised systems have been proved to be quite effective to allow for trusted and accountable data sharing, without the need to resort to a centralised party that collects all the information. While complete decentralisation provides important advantages in terms of data sovereignty, absence of bottlenecks and reliability, it also adds some issues concerned with efficient data lookup and the possibility to implement complex queries without reintroducing centralised components. In this paper, we describe a system that copes with these issues, thanks to a multi-layer lookup scheme based on Distributed Hash Tables that allows for multiple keyword-based searches. The service of peer nodes participating in this discovery service is controlled and rewarded for their contribution. Moreover, the governance of this process is completely automated through the use of smart contracts, thus building a Decentralised Autonomous Organization (DAO). Finally, we present a use case where road hazards are collected in order to test the goodness of our system for geodata retrieval. Then, we show results from a performance evaluation that confirm the viability of the proposal.

1 | INTRODUCTION

The digitalisation process, which has been ongoing over the last decades, has seen data management and data delivery become crucial issues. The transformation brought about by digital technologies has data at its core and it had a significant impact on economies and societies around the world. The ability to easily get hold of data has the potential to create several new services based on data and new markets where more and more users are consumers and providers at the same time. However, obtaining large amounts of data that is not from a dubious (or false) origin is often a challenge. In order to cope with the increasingly higher number of content that is demanded through the Web, multiple solutions for efficient use of the Internet have been designed. In particular, thanks to the decentralisation of content storage and delivery, it is possible to avoid the single point of failure, while reducing the workload at data centres and allowing a distribution of data that remains 'closer' to the original data producer. Decentralisation also fosters the creation of open systems, where participants can freely join the system and then contribute to its functioning.

Recently, Distributed Ledger Technologies (DLTs) and a realm of decentralised systems, for example, Decentralised File Storages (DFS), have emerged as Peer-to-Peer (P2P) technologies capable of offering interesting features related to data validation and trustfulness [2, 3]. DLTs have gained popularity with the advent of cryptocurrencies, which allow users to trade crypto-assets without any central entity being involved, ensuring transparency and data integrity. By creating a common, decentralised and trustless infrastructure, it will be possible for data consumers and providers to interact and collaborate in P2P interactions [4, 5]. Benefits often cited of DLTs, indeed, include the enabling of secure transactions between untrusted parties through consensus mechanisms, high availability, and the ability to automate and enforce processes through smart contracts [6]. Besides the financial use case, DLTs and DFS provide some properties such as data integrity, authenticity, confidentiality and auditability, which are used to

An early version of this work appeared in [1]. This paper is an extensively revised and extended version where more than 50% is new material.

build novel applications for a 'more' decentralised Internet [7]. 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 the Internet. A relevant example is the shutdown of Wikipedia that happened in 2017 within certain countries, with the InterPlanetary File System (IPFS) that was still able to guarantee access through mirroring [8].

One of the concerns, that is still open with respect to these novel technologies, is related to the data discovery and lookup operations. Data inserted in DLTs and DFS are usually unstructured and no efficient mechanisms are available to query certain kinds of information such as, for instance, data generated by sensors in a given geographical area. Thus, even if anyone can run public DLTs and DFS nodes, such as Ethereum [9], IOTA [10] and IPFS [11], data lookup can be very slow and expensive. In fact, data are rarely stored in a format that can be consumed directly and they need to be filtered and indexed before executing any complex query. Data are referenced through addresses or indexes that, most of the time, are not related to the content of the data and thus are not useful for their categorisation. Specifically, data can only be accessed by knowing their respective identifier or location and cannot be searched based on the specific content. Put in other words, these systems lack a viable (and decentralised) data management scheme that enables the efficient execution of 'complex' queries on top of them.

In this paper, we propose a decentralised system for key-value metadata-based lookup, which allows for retrieving contents stored in DLTs and/or DFS. Our approach relies on the use of a Distributed Hash Table (DHT) as a layer placed on top of the DLTs/DFS, which offers the possibility to perform multiple keyword searches. The DHT is essentially a peer-to-peer hypercube overlay structure [12]. Each domain of the hypercube corresponds to a keyword and specific DHT nodes are responsible for a certain portion of the ID-space. Through navigation inside the hypercube, it is possible to query the node that maintains information about contents stored in the DLT/DFS that have certain keywords as metadata.

The first contribution of this paper consists in describing the design of such a hypercube-based DHT architecture on top of DLTs and/or DFS. The hypercube is a logical layout in which there are 2^r nodes, each one labelled with a r-bit string identifier (ID) and connected to the r nodes whose ID differs by only one bit [12]. Each node is responsible for a specific keywords set, derived from their ID. An interesting aspect of the specific hypercube-based DHT is that it allows the user to efficiently search for objects matching a specific keywords set K. Moreover, it allows for searching the supersets of K, thus enabling the construction of queries that are more complex than a single <keyword, value> lookup. The hypercube structure allows for optimising the routing of the queries, by limiting the number of hops needed to locate contents. Even if our approach is independent of the underlying DLT and it can be easily extended to other distributed ledgers and DFS, we provide a specific system design that is tailor-made for IOTA [10]. To the best of our knowledge, at the time of writing, this is the first example of using such a solution over DLTs and/or DFS.

The second main contribution is about the creation of a framework for the governance of the layer-two P2P architecture, to improve the scalability and the decentralisation of the system. 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, for example, BitTorrent. Therefore, we leverage DLTs to build a Decentralised Autonomous Organisation (DAO) [13] for the governance of the DHT network. 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 organisational decisions.

The third main contribution is about a case study in which the proposed architecture is used for geodata storing and retrieval. We show how this setup can significantly benefit from Hypercube DHT specific characteristics. Geodata is defined as data holding an implicit or explicit association with a relative location on Earth, that is, a geographic location or geographic position [14]. One of the most common representations of geodata is in the form of latitude and longitude coordinates, but generally, geodata can take the form of vectors, rasters, multi-temporal points or different types of geotagged data [15]. More specifically, the use-case we present consists of a service that is designed for road hazard detection in intelligent transportation systems. This setup will demonstrate how useful the hypercube structure can be for managing geodata, for example, for the creation of Location Based Services [15].

Finally, we report an experimental validation of the implementation of the proposed architecture. 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. We also provide results obtained from a detailed simulation analysis, which confirm that our system allows for multiple keyword searches in a reasonable time (i.e. in the order of the logarithm of the number of hypercube nodes). Finally, we evaluate the smart contracts used for the formation of the DAO, implemented to be executed on the Ethereum blockchain, in terms of operations execution cost.

The remainder of this paper is structured as follows. Section 2 provides the background and related works. In Section 3, we present the system architecture for the decentralised system based on the Hypercube DHT and in Section 4 a technical solution for its governance is discussed. In Section 5, an experimental evaluation is provided and then discussed in Section 6. In Section 7, we then provide the final remarks.

2 | BACKGROUND

In this Section, we introduce background notions needed for the rest of the paper and then we discuss the related works.

2.1 | Distributed Hash Table

A Distributed Hash Table (DHT) is a decentralised system for the distributed storage of contents that provides the functionalities of a hash table, that is, a data structure that efficiently maps 'keys' into 'values'. 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). In addition, each node stores a partial view of the entire network, with which it communicates for routing information. To reach nodes from one part of the network to another, a routing procedure typically traverses several nodes, approaching the destination at each hop. The association of objects to DHT nodes is obtained through the use of a hash function, a one-way function that 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. The 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 [16]. This type of infrastructure has been used as a key element to implement complex and decentralised services, such as Content-Addressable Networks (CANs) [17], DFS [11], cooperative web caching, multicast, and domain name services.

2.2 | Distributed Ledger Technologies (DLTs)

A Distributed Ledger Technology (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 of the stored information. Consensus mechanisms are implemented in order to enable two parties to transact directly without the need for a third-party. The main peculiarity of DLTs is that they ensure untampered data availability. Data written on the ledger are trustworthy, because DLT protocols ensure their integrity, immutability, and authenticity. Thus, they promote the development of trustful and reliable service applications [18, 19]. There are different implementations of DLTs, each one with its pros and cons. One of the main distinctions lies in the support of smart contracts, for example, Ethereum [6]. This feature is quite often in contrast with other key features, related to the level of scalability and responsiveness of the system [20]. Conversely, some implementations are thought to provide better scalability at the expense of lacking some features, for example, based on Direct Acyclical Graphs (DAGs).

While layer-one protocols and technologies in DLTs define the form of the ledger, its distribution, consensus mechanism and features, layer-two solutions are built on top of layer one without changing its trust assumptions, that is, the consensus mechanism, or the structure [21]. Layer-two protocols allow users to communicate through private channels, reducing the transaction load on the underlying DLT.

2.2.1 | IOTA

IOTA is a DLT that allows hosts in a network to transfer immutable data among each other. In the IOTA ledger, that is, the Tangle [10], the vertices of a DAG represent transactions and edges represent validations to previous transactions. The validation approach is thought to address two major issues of traditional blockchain-based DLTs, that is, latency and fees. IOTA has been designed to offer fast validation, and no fees is required to add a transaction to the Tangle [22]. When a new transaction is to be issued, two previous transactions must be selected (i.e. tips selection) and approved by referencing those in the transaction. The result is represented using directed edges in the Tangle. To validate a transaction a Proof-of-Work (PoW) is performed (in order to deter denial of service attacks and other service abuses). In IOTA, the transactions are called messages and are referenced by a message ID (we will use these terms through the paper).

2.2.2 | Smart contracts and their use cases (Decentralised Autonomous Organisation and decentralised finance)

Smart contracts are programs whose execution is performed in a distributed way. In Ethereum [9], 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 automatise and supervise the exchange of digital or physical assets, to create tokens, that is, the representation of physical assets or utilities such as ERC20 Tokens [23], and to allow the management of a DAO [13]. In order to enable the decentralised management of a DAO, smart contracts implement transactions, currency flows, rules and rights within the organisation. DAO members can make proposals for the management of the organisation and also discuss and vote on 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.

Decentralised 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 [24]. With DeFi protocols such as Decentralised Exchanges (DEX), anyone can engage in non-custodial exchange

of on-chain digital assets, for example, tokens. In contrast to traditional finance where an asset's liquidity is based on the bid and ask order 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 [24].

2.3 | Decentralised file storage

Decentralized File Storages (DFS) offer an alternative to the traditional client-server models, that is, 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 has the requested contents, it is possible to rely on a DHT system that is in charge of mapping the items with the addresses of the peers owning such data. DFS follows this approach and offers high data availability and resilience using data replication.

The IPFS is a DFS and a protocol thought for distributed environments with a focus on data resilience [25]. The P2P network that runs the IPFS protocol, stores and shares files in the form of IPFS objects that are identified by a CID (i.e. Content IDentifier). This CID consists of 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 CID (i.e., hash digest) is required.

2.4 | Related works

The popularity of IoT devices and smartphones, and the associated generation of large amounts of data derived from their sensors, have resulted in the interest of individuals in the production and consumption of data via a data marketplace [26]. Making data (which are mostly personal) available for access and trade is expected to become a part of the data-driven digital economy [27]. As introduced earlier, the use of DLTs has been proposed for the implementation of data marketplaces [28, 29] to take advantage of: (i) reliance on third party platforms not needed; (ii) better resilience against network partitioning and single points of failure; (iii) privacy-preserving mechanisms [19]. Most of the related works investigate the data distribution through DLTs, focussing in particular on the use of off-chain storage based on DFS with data links referenced in DLTs [3, 18, 19]. In [2], the authors provide the implementation of a data marketplace based on the use of DFS for storing data and a payment protocol that exploits Ethereum smart contracts [6, 30]. Similarly, in [4, 5], the proposed systems are based on P2P interactions and smart contracts to reach an agreement, while also integrating other components such as the IOTA DLT.

On the other hand, decentralised data search on DLT and DFS is a broader field that has been addressed by both scholars and developers. The Graph is one of the first protocols with the aim to provide a 'Decentralised Query Protocol' [31]. The Graph network consists of a system built upon Ethereum and IPFS, which allows users to query data stored by means of these two technologies. The Graph users can query several indexers nodes by paying for their metred usage, within a query market. The organisation of the network is similar to what is referred to as DAO, however their method for storing indexes is different from our proposal. Indeed, instead of using a DHT network storing data, the Graph P2P network is based on the use of a Service Addressable Network used to locate nodes capable of providing a particular service, which can be any arbitrary computational work.

Specifically for IPFS, in order to overcome the file search limitation, a generic search engine has been developed, namely 'ipfs-search' [32]. This solution is rather centralised and does not escape the problem of concentration similar to the conventional web. In response to this, a decentralised solution called Siva [33] 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 optimisation for a keyword storage structure apart from the use of caching. In [34], the authors propose a layer-one keyword search scheme that implements oblivious keyword search in DFS. Their protocol is based on keywords search with authorisation for maintaining privacy with retrieval requests stored as a transaction in a blockchain (i.e. layer one). Finally, a layer-two solution for keywords search in DFS has been proposed in [35], where a combination of a decentralised B+Tree and HashMaps is used to index IPFS objects [11].

3 | SYSTEM ARCHITECTURE

Our contribution focuses on the design of a decentralised system that allows for queries based on multiple keyword searches. We refer to it as the Hypercube DHT and to its governance. This solution can be leveraged in several cases, especially when decentralisation is required. In general, the kind of applications that can use our proposed approach is the one that depends on data stored on DLTs and DFS. Examples are of decentralised applications are those that validate smart city crowdsensed data through DLTs [3] or distributed collections mirrors such as the Wikipedia over IPFS [8].

Figure 1 shows a layered view for the system architecture of our decentralised data lookup service. It is based on the following components:

0. A network of nodes underlying a decentralised system, for example, DFS or DLT network.
1. A decentralised system that acts as layer one of a distributed application. This can be either:
   - a DFS, used to store data/files in an encrypted form and offering high availability [36];
   - a DLT, which provides a ledger for data indexing and validation in the form of hash pointers [3, 37].

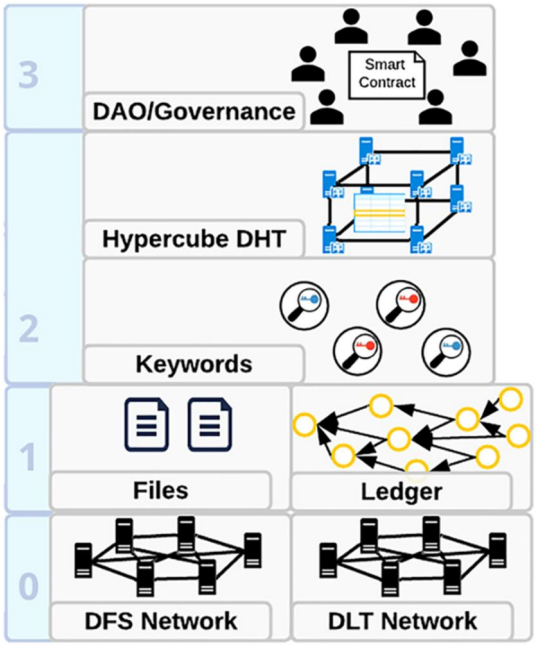

FIGURE 1 Layers in the context of Distributed Ledger Technologies (DLTs). Layer zero consists of the DLT (or Decentralised File Storages (DFS)) network, while layer one is the set of software frameworks run by the network nodes, for example, the ledger (or the file storage). Layer-two solutions are the ones that leverage layer one for providing other services, that is, the Distributed Hash Table (DHT) Keyword Search in our case. Layer three consists of the set of technologies and processes that form the governance of the layer-two solution, that is, the DAO in our case

2. The Hypercube DHT (described in Section 3.2) is a layer-two solution on top of the previous layers. A mapping of keywords to DFS/DLT identifiers consists of the link between the layer one and layer two. The Hypercube DHT thus stores these keywords and provides a distributed mechanism for the search of data in the underlying decentralised systems.
3. An upper layer embodies a set of smart contracts, technologies and processes forming the governance of the Hypercube DHT network (see Section 4).

3.1 | Decentralised system for data sharing

Before going into the detail of the Hypercube DHT specification, we also explain its interoperability with other decentralised systems, such as DLTs and DFS. It is important to emphasise the fact that the Hypercube DHT solution is agnostic to any underlying system. The only requirement that the underlying system must have is to make use of a unique id to refer to a specific piece of information. The Hypercube DHT represents 'just' the means for binding specific keywords to a specific id, in a decentralised fashion.

Among the many possible implementations of DFS and DLTs we decided to focus on the Hypercube DHT interoperability with the IPFS DFS and with the IOTA DLT (both introduced in Section 2).

The rationale behind the choice of IPFS is twofold: (i) this implementation of a DFS is the most referenced and deployed in both the academic research and 'decentralised web' applications; (ii) it uses data identifiers that do not focus on where the content is stored (e.g. web urls), but it forms a kind of address based on the content itself, that is, the CID, thus paving the way to further data validation and efficiency management. This kind of identification supports InterPlanetary Linked Data formats, enabling decentralised data structures to be universally addressable and linkable and thus allowing the creation of several addressable decentralised applications [38].

In choosing the DLT, our choice was primarily motivated by the specific use case considered in this paper. This consists of data sharing in general and geodata in particular (with a focus on vehicular geodata in Section 5). While blockchains such as Ethereum and EOS offer the means to build general-purpose decentralised applications with good success [39], we have opted for the IOTA DLT in our work, because of its highly compatible features with data sharing scenarios. In fact, the low latencies for inserting transactions and the absence of fees to add data into the Tangle make IOTA particularly suitable for our case study. While other DLTs, such as Steem [40], offer feeless transactions, IOTA has been designed for use cases involving the exchange of data between (IoT) devices. In our case, the IOTA's feature of treating data messages as single non-payable and feeless transactions is favourably exploited by the Hypercube DHT mechanisms. While several solutions are possible, we consider three principal decentralised data management instances:

DFS: Taking IPFS as a reference example, an IPFS Object, for example, a file, is uniquely identified through a CID. When data are uploaded to IPFS and referenced using CID, then, the Hypercube DHT will store the mapping between keywords and data's CIDs (Figure 2, left).

DLT: Taking IOTA as a reference example, a Hypercube DHT implementation would store a DLTAddr, that is, a generic reference to a DLT address or transaction identifier (see Figure 2, right).

DLT+DFS: The combination of DLTs and DFS is a prominent solution when data sharing is involved [36]. Data can be stored in a DFS and then referenced in a DLT, for example, using the CID in IPFS. Storing data into a DFS usually requires lower latency with respect to those that can be experienced using DLTs, while the latter provide validation through the publication of the data digest into the ledger and the use of smart contracts. Moreover, this solution is better suited for personal data in order to comply with some regulations, for example, the right to be forgotten [36]. In this DLT-DFS 'mixed' case, the Hypercube DHT is used for only storing one reference, that is, the DLTAddr of an object in the ledger that subsequently refers to an object in the DFS using a hash pointer (Figure 2, bottom).

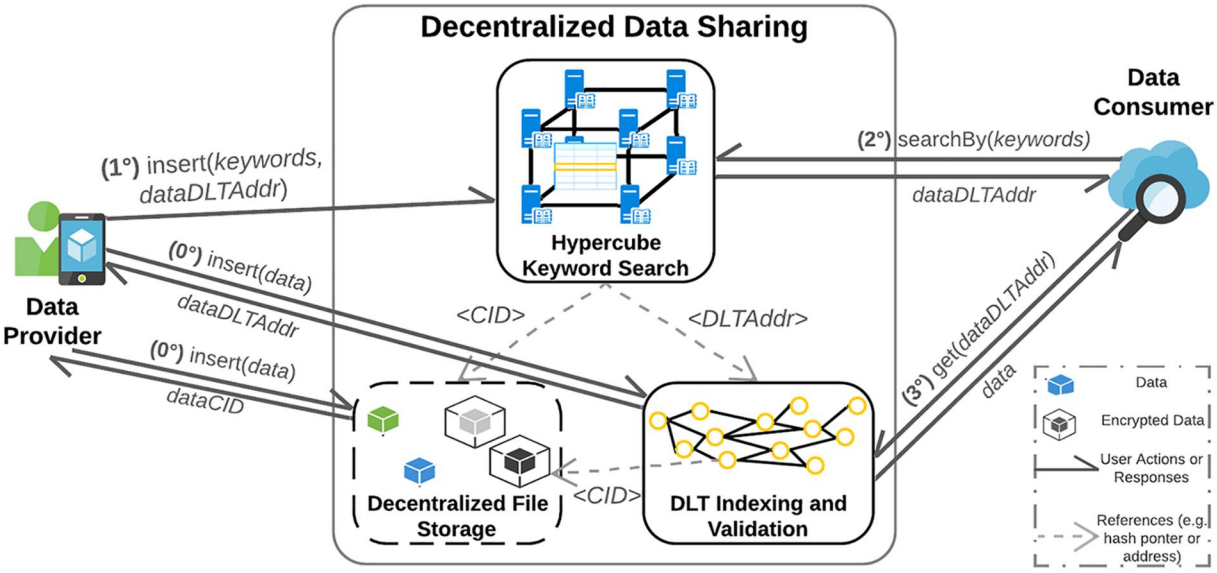

FIGURE 2 Overview of the decentralised architecture including the Hypercube Distributed Hash Table (DHT), a Distributed Ledger Technologie (DLT) and a Decentralised File Storages (DFS), that depicts the main relations and operations for data sharing and querying

From now on we will only refer to the IOTA DLT and we will describe how the mapping of keywords to references in IOTA is done, before continuing with the queries description. In particular, we refer to the use of IOTA messages (i.e. transaction) to enable data sharing in the form of message streams. Based on this model, thus, the retrieval of contents is based on lookup for references to IOTA messages. These messages contain data itself (in an encrypted form) or a reference to the data stored in a DFS, that is, hash pointers.

Without the use of the Hypercube DHT, in order to obtain information from a IOTA message, it is necessary to know the exact address of the message, that is, the message id. This id becomes what we have generally called the reference in a DLT or DLTAddr. No mechanisms are provided by IOTA for the discovery based on the content of certain data available in the Tangle, since the message id value does not provide any information related to the type and kind of a message. In the following section, we will describe how a single message included in IOTA is indexed by a keyword set and then how such a keyword set is exploited to lookup for specific kinds of contents.

3.2 | Hypercube distributed hash table

Considering O as the set of all messages in IOTA, the idea is to map each object $o \in O$ to a keyword set $K_o \subseteq W$, where W is the keyword space, that is, the set of all keywords considered. Thus, in general, a keyword set $K \subseteq W$ can be associated with a data content (i.e. the metadata associated with it) or a query (i.e. we are looking to some content associated with a specific metadata). By using a uniform hash function $h : W \rightarrow \{0, 1, \ldots, r-1\}$, a keyword set K can be represented by the result of such function, that is, a string of bits u where the 1s is set in the positions given by $one(u) = \{\text{mod}_r(h(k)) | k \in K\}$. In other words, each $k \in W$ has a fixed position in the r-bit string given by $h(k)$ and that position can be associated with more than one k (i.e. hash collision). Then, every keyword set K is represented by a r-bit string where the positions are 'activated', that is, are set to 1, by all the $k \in K$. We use these r-bit strings to identify logical nodes in a DHT network, for example, for $r = 4$ a node id can take values such as 0100 or 1110. Inspired by [12], we refer to the geometric form of the hypercube to organise the topological structure of such a DHT network. $H_r(V, E)$ is a r-dimensional hypercube, with a set of vertices V and a set of edges E connecting them. Each of the 2^r vertices represents a logical node, whilst edges are formed when two vertices differ by only one bit, for example, 1011 and 1010 share an edge. In the DHT, the network node represented by a vertex u is directly connected, that is, neighbour, to a node represented by a vertex v that shares an edge with u. The Hamming distance can be used to find out how far apart two vertices u and v are within the hypercube, that is, $Hamming(u, v) = \sum_{i=0}^{r-1}(u_i \oplus v_i)$, where $\oplus$ is the Exclusive OR operation and u_i is the bit at the ith position of the u string, for example, for $u = 1011$ and $v = 1010$, we have $Hamming(u, v) = 1$.

Having this Hypercube DHT set in place, the data sharing process works as follows (see Figure 2). Upon creation of a data object o, it is associated with a set of keywords K_o describing it. In particular, the Data Provider firstly inserts a piece of data in the IOTA DLT (Figure 2, step 0) and obtains the id of a message o (dataDLTAddr in the figure). The reference to object o is then inserted in the Hypercube DHT together with a set of keywords K_o that describe the piece of data (step 1). The contacted Hypercube DHT node will use the associated r-bit string (step 2) to reach the logical node responsible for K by means of a routing mechanism (similar to the one shown in Algorithm 1), in order to obtain the set of objects such that their related set of keywords includes the all the keywords in K_o (step 3).

3.3 | Keyword-based complex queries

In the Hypercube DHT system, contents can be discovered through queries that are based on the lookup of multiple keywords, associated with data. Such queries are processed by the DHT-based indexing scheme described in the previous section. For instance, let's say that $W = \{$*'Bologna', 'San Donato', 'Temperature', 'Celsius'*$\}$. We associate a r-bit string to these keywords, in this order. Thus, as an example 1010 represents the keyword set $K = \{$*'Bologna, Temperature'*$\}$. Let's say that $u \in V$ is the node with id equal to 1010; then u is responsible for K and it maintains a list of message ids containing the temperatures measured in the city of Bologna.

3.3.1 | Multiple keywords search

The system that we propose provides two functions for performing queries based on multiple keywords:

- **Pin Search** - this procedure aims at obtaining all and only the objects associated exactly with a keyword set K, that is, $\{o \in O | K_o = K\}$. Upon request, the responsible node returns to the requester all the message ids of the corresponding objects that it keeps in its table, associated with K.
- **Superset Search** - this procedure is similar to the previous one, but in addition, it also searches for objects that can be described by keywords sets that include K, that is, $\{o \in O | K_o \supseteq K\}$. Since the possible outcomes of this search can be quite large, a limit l is set to the number of returned results.

In the Pin Search, we need to retrieve objects only from one node. Whilst, for Superset Search, we need to retrieve objects from all the nodes that are responsible for a Superset of K. Such nodes are contained in the sub-hypercube $SH(S, F)$ induced by the node u responsible for K, where S includes all the nodes $s \in V$ that 'contain' u, that is, $u_i = 1 \Rightarrow w_i = 1$, while F includes all the edges $e \in E$ between such nodes. Thus, during a Superset Search, the induced sub-hypercube is computed and then only nodes in such sub-hypercube are queried using a spanning binomial tree as described in [12] (definition 4.2). More specifically, the l limit is a query parameter that indicates the maximum number of objects to return when traversing the spanning binomial tree.

3.3.2 | The query routing mechanism

Queries can be injected into the system by users external to the DHT to any $v \in V$ DHT node. Through a routing mechanism, a query q with a defined keywords set k will reach a node $u \in V$ that is responsible for that keyword set. If q is of type Superset Search, then the query will reach all the nodes that are responsible for keyword sets that include K, until the limit of objects l is reached. This process is described in detail in Algorithm 1. This algorithm is executed by each node, every time it gets a new query from a neighbour or from a user. The routing mechanism from a node $v \in V$ receiving a query q from a user, with a keyword set K and a limit l is as follows:

1. If node v is not responsible for K, that is, $\{h(k) | k \in K\} = one(u) \neq one(v)$, then it computes, for all its neighbour nodes, the Hamming distance to node u;
2. Node v broadcasts the query q to the neighbour w with the lowest distance to u;
3. These two steps are repeated by w and by the subsequent nodes, until the query q reaches u;
4. If the query q is of type Pin Search u returns the objects references associated with K, that is, $\{o \in O | K_o = K\}$;
5. Else (in the case of a Superset Search) u computes its children nodes in the spanning binomial tree of the induced sub-hypercube;
6. Then node u broadcasts q to the children nodes until the limit of objects l is reached;
7. The children nodes will repeat the process from step 5 with their children and then return the objects;
8. Finally node u returns the aggregated objects references associated with different supersets of K, that is, $\{o \in O | K_o \supseteq K\}$.

Algorithm 1 QueryRoutingMechanism

Input: q query, K keyword set, l limit
Data: v node string, $one(v)$, $neighbors(v)$
Result: $\{o \in O \mid K_o \supseteq K\}$

```
one(u) ← {h(k) | k ∈ K}
if one(u) ≠ one(v) ∧ From(q) = "User" then
    w ← {n | n ∈ neighbors(v) ∧ Min(Hamming(n, u))}
    return QueryRoutingMechanism(w, q, K, l)
else
    if Type(q) = "PinSearch" then
        return GetObjectsFromIndexTable(K, −1)
    else if one(u) ⊆ one(v) then // i.e. SupersetSearch
        objectsList ← GetObjectsFromIndexTable(K, l)
        l ← l − Length(objectsList)
        From(q) ← "Node"
        while l > 0 do
            c ← GetNextSBTreeChild(u)
            cList ← QueryRoutingMechanism(c, q, K, l)
            objectsList ← objectsList + cList
            l ← l − Length(cList)
        end
        return objectsList
    end
end
```

3.4 | Geoposition-based keywords encoding

In this Sub-Section, we consider aspecific keywords encoding concerned with geolocation data, that is, where the metadata that is associated with a datum and that represents the

geoposition where the datum has been generated (geodata) is as important as the datum itself. In such cases, a mechanism can be employed to automatically associate a given geographical position to a keywords set for the Hypercube DHT. To this aim, we employ a conversion based on a double encoding.

The first encoding consists in simplifying the form of the geoposition data, converting them into *Open Location Code (OLC)* [41]. This code represents highly accurate street addresses, similar in length to telephone numbers, which can be shortened to just four or six digits. The fewer the digits, the larger the squared area represented, and vice versa. For instance, a 4-digit code such as 6*P*23 identifies a particular squared area with a side of 110 km.

The second encoding is a particular type of logical transformation that allows for the OLC of a certain geographical location to be converted into a set of keywords, that is, positions in the r-bit. For example, given the OLC '6P0000+', one can associate each character (or a set of characters) to a certain position in the r-bit string. Associating respectively the characters '6' and 'P' with r-bit strings '000001' and '000010', then the conversion of '6P0000+' containing both will be the union of both, that is, '000011'. Moreover, the association between the characters and their position in the OLC must be univocal, meaning that encoding an OLC representing a smaller area included in a larger area must result in a r-bit string including the r-bit string of the OLC representing the larger area. For instance, an OLC representing a squared area of side 2200 km, for example, '6P000000+', is included in the OLC representing a squared area of side 3.5 m, for example, '6PH57VP3+PR'; thus, assuming '000011' r-bit string encoded from '6P0000+' a r-bit string such as '101011' can encode '6PH57VP3+PR' because $one(101011) \supset one(000011)$.

Take as an example the case of Figure 3, where an object $o \in O$ is stored in an IOTA message and its related geoposition is encoded with OLC, that is, $OLC_o = 6PH57VP3 + PR$. The OLC_o is then split into a number of pieces that is based on the dimension of the Hypercube DHT (that dictates the precision) to create a keyword set $K_o = split(OLC_o) = \{6P00000000, 00H5000000, 00007V0000, 000000P300, 00000000PR\}$. The final r-bit string is obtained by the ones in the position obtained by hashing each keyword in K_o, that is, for r-bit string u $one(u) = \{\mathrm{mod}_r(h(k)) | k \in K_o\}$ and $u = 101011$. As one can see, the precision depends on the length of the r-bit string and thus on the hypercube dimension. However, this measures only how many OLC based keywords a node would store that result in the same r-bit string. In fact, the more large the r, the fewer hash collisions there will be.

4 | DECENTRALISED AUTONOMOUS ORGANISATION FRAMEWORK

The contribution presented so far in this paper describes the use of keywords for data retrieval, and how these keywords are saved and how to use them to execute queries, all by using the Hypercube DHT. Layer two provides the technological means for implementing a keyword based search solution over a decentralised system. This can be enough for offering a complete solution in many cases.

The main focus of this section, on the other hand, is the layer on top of this one (see Figure 1). It consists of technologies and processes that form the governance of the Hypercube DHT network, that is, the DAO. In the P2P network that supports the Hypercube DHT, we envision the case in which the nodes are

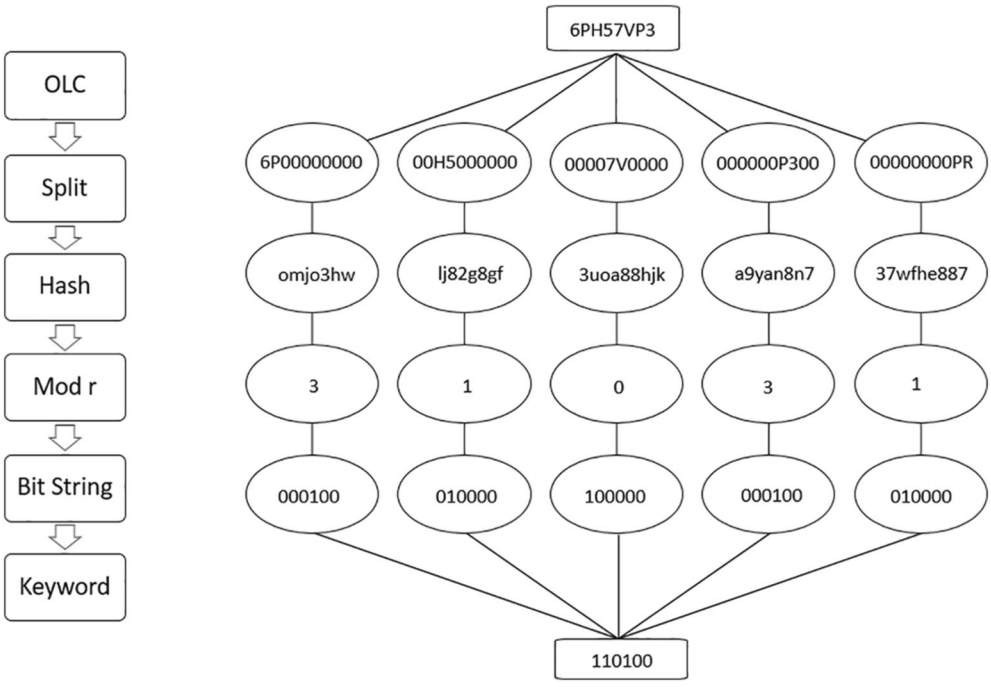

FIGURE 3 The graphical depiction of the process of encoding an OLC into a r-bit string, for $r = 6$

interested in keeping the network operational and healthy. The aim of the node operators should be to act in the context of a sharing economy, for example, as Wikipedia editors are encouraged to contribute to the free encyclopaedia [42]. P2P networks have this 'cooperative vein' intrinsically built into their structure. In this context, it has been argued that the use of DLTs can 'crystallise' the dynamics of a model of socio-economic production in which large numbers of people work cooperatively, that is, commons-based peer production [43]. We are interested, then, in describing the case where, in order to orchestrate the operational decisions and rewards, the DHT network nodes operators can form a DAO.

4.1 | Governance layer

The governance layer is mostly based on the use of smart contracts and the interfaces for interacting with those. Smart contracts, indeed, enable the creation of an organisation that takes advantage of a token-based economy and decentralised voting. In particular, we use Ethereum smart contracts to structure the DAO based on the following components [44, 45]:

Token economy: The DAO is built around the use of a unique token, that is, 'DAOToken', used for transferring value (e.g. users that pay node operators), or for stacking 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 [23].

Members Registry: A smart contract was developed as a members registry, to allow token holders to time-lock DAOTokens 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 ends.

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.

4.2 | Rewarding system

In this sub-section we describe one of the many rewarding systems implementable, namely the reward for DAO members passing through DeFi. For instance, in the case of the DAOToken, a Uniswap liquidity pool smart contract [24] can be created by locking into it x DAOTokens and an amount of y other ERC20 tokens to be exchanged with. The value of a single DAOToken with 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, for example, 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 burnt by the providers in order to redeem their share of liquidity (and accrued fees obtained when exchanges happen). This means that when a new 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, in our design, the DAO is based not on the time-lock of the DAOToken directly, but on the time-lock 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 interface. 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 behaviour, making it possible to have a strong incentive to behave correctly.

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 simulation in order to test a network populated by a higher number of nodes, and finally, we tested a prototype of the DAO smart contract framework.

In order to assess the viability of the solution proposed in this paper, we implemented the software in charge of building and maintaining the Hypercube DHT network and then we tested it over the Road Hazard Detection use case. The DHT software is implemented in Python and it exposes the four main node's actions by means of the Flask server framework [46], that is, Insert object, Remove object, Pin Search, Superset Search. Together with the core logic methods for a logical node, the implementation also includes an interface for communicating with an IOTA node, in order to operate with the DLT.

5.1 | Road hazard detection use case

The implemented use case is based on road hazard detection. The vehicular simulation and the tests are stored as Open Source code on Zenodo [47]. The first, fundamental step is the creation of the data to identify road anomalies. Generally speaking, it is possible to distinguish these techniques into two macro-categories (i) manual: the user himself signals the presence of pitfalls by means of photographs or descriptions indicating their geographical location; (ii) sensor-based: they use the gyroscope, accelerometer and GPS of smartphones to detect and measure the vibrations that occur in the presence of potholes and bumps.

The collected data are inserted into the Tangle through simple IOTA messages. For this specific task, the data of interest are the geodata related to the position where the hazard is recorded, but many other data points are also considered in our use case. For example, photographs, gyroscope data, vehicle status, etc. The message id, acquired after entering the data in the Tangle, is indexed in the Hypercube DHT by associating it with a set of keywords that allows for its subsequent retrieval. These keywords are determined by the encoding process (previously described in Section 3). In particular, the geoposition keywords follow what was described in sub-section 3.4.

In the data retrieval phase, given a certain location the system must report all the registered road faults within that specific area. For instance, giving in the input an OLC such as '6PH57VP3+PR' should not result in obtaining only the road hazards for that precise location, that is, a Pin Search, but should result in a set of hazards in a larger area, that is, a Superset Search. Once retrieved, the data, in geospatial format, are displayed in real-time on the maps on users' devices, providing navigation support. Thanks to Superset Search, the user will be alerted of the presence of an anomaly on the road long before being at the exact point where the alert is located. Thus, the driver (or the autonomous vehicle) will have a sufficient amount of time to slow down in the immediate vicinity of the anomaly, to avoid it or to take an alternative route.

5.1.1 | Vehicular simulation

In order to test the issuing and retrieving of data based on geolocation, we simulated a vehicular scenario in which road hazard detection was performed through the Hypercube DHT and IOTA. We stress the fact that in this evaluation, while the vehicular environment was simulated, in order to generate data and queries, the rest of the system architecture (i.e., DHT, smart contract execution and DLTs) was deployed and executed in a real system (the test setup details are provided in the next sub-section).

The two main elements for the creation of realistic scenarios are the presence of vehicles and the roads they drive on. Furthermore, a third necessary element is the presence of potholes/obstacles along the routes. It is also necessary to consider an aspect of complexity, that is, the presence of traffic that allows us to stress the system and to consequently evaluate its ability to adapt to real-world situations. As far as the scenario simulation is concerned, the following steps were followed:

- Creation of a simulated geographical area: in our tests 10 routes were generated, represented by lists of geographical coordinates, each of which has at least one point (latitude, longitude) in common with another;
- Definition of the generic vehicle to be placed in the simulated environment: each vehicle has a starting point, a destination and a path to follow;
- Generation of road hazards (potholes/obstacles): randomly generated in the simulation environment and stored in the (real) Hypercube DHT by the vehicles passing nearby;
- Vehicle system simulation: once created, vehicles movements were simulated; during its path each vehicle was in charge of periodically (every 3 min) querying the system to query for road hazards.

For each route, 10 vehicles were instantiated and started simultaneously. In order to stress the DHT, for each vehicle, the delay time between sending one query and the next was of 3 min. In a real context, on the other hand, the use of Superset Search would allow the user to obtain aggregated data related to vast areas, therefore, the frequency of the queries could be quite low for each vehicle. Once the OLC has been determined the query is directed to a randomly chosen node of the Hypercube DHT. From here, through the routing mechanism, the node i.e. responsible for the specified keyword set is reached, which, in the case of Pin Search, returns all objects associated with it. In the case of Superset Search, on the other hand, a maximum threshold of 10 objects is set, which are returned to the user.

5.1.2 | Test setup

In our implementation, the same physical node can host more than one logical hypercube node and that is what we made use of, in order to perform the tests. We tested our implementation running the software on a dedicated host (i.e. a quad core Intel Core i7 CPU, 16 GB RAM). Each logical node was executed by a dedicated Flask server and after a bootstrap phase, and each node was connected to its neighbours following 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 objects generated following a vehicular simulation (as described below).

With regard to IOTA, two different ways of issuing data on the Tangle were tested, that is, through a local PoW or by delegating it to an external IOTA node. In our tests, the IOTA Mainnet was used. In addition, two types of full nodes were used:

- **Private node**: a node that we set up and connected to the main IOTA DLT network, dedicated to our requests only (1 core CPU, 2 GB RAM, 50 GB storage, Virtual Private Server);
- **Public node**: a public IOTA DLT network node that accepts requests to issue messages to and read from the ledger (https://chrysalis-nodes.iota.org).

5.1.3 | Insert tests - signalling a hazard

The process of signalling a hazard to the system involves two operations that are performed in sequence. The first one consists in generating the IOTA message. This operation involves the creation of a message to be submitted to the DLT network. Upon generation, the device operating for the vehicle stores the data related to the hazard into a message, that is, the geolocation and other structured data representing the hazard information. Then, it performs the steps necessary for issuing the message to the DLT, that is, tips selection and PoW. For the tips selection, na IOTA node maintaining the ledger is needed. For the PoW, two kinds of situations can happen, that is, the PoW is locally executed by the user device or the PoW execution is delegated to an IOTA node. PoW and tips selection processes are described in Section 2. The second operation consists in inserting the keywords on the Hypercube DHT. When a message id is obtained from the previous operation, the data is inserted into the Hypercube DHT based on the keywords extracted from the geographical information.

- *IOTA Insert Results*: The latency delays in the IOTA Mainnet are variable due to the time required for tip selection and the difficulty of the PoW. The time required for inserting a message is, on average, ∼3.6 s for the private node (see Figure 4, left) and ∼1.5 s for the public one (see Figure 4, right). One relevant aspect concerns the difference in terms of the way the PoW is carried out. The insertion times obtained by performing the PoW locally and remotely are shown in Figure 4. First of all, in both cases it can be seen that although the insertion of messages with remote PoW seems a bit faster, the difference between local and remote is not so marked. It is necessary to keep in mind that, however, nodes with suitable hardware (for this task) are required to achieve results of this kind; without them, delegating the PoW burden to the IOTA node would cause a relevant slowdown [37].
- *DHT Insert Results*: In the DHT, the latency times depend on the efficiency of the DHT routing mechanism, as well as on the size of the network. Figure 5 (left) shows how the average time for inserting the object in the DHT grows as a function of the parameter r that determines the size of the hypercube. Varying from a little more than one second, with a $r = 3$, to a little more than two with a $r = 6$, the average insertion time is to be considered as low. A second aspect, visible in Figure 5 (left), concerns the presence of outliers in all the simulations carried out; in fact, in some cases, the insertion in the DHT requires longer times than on the average. The presence of outliers, whose frequency increases as the size of the network increases, is, in part, due to the randomness factor in the choice of the node that receives the request. In fact, if the requested node might be very distant in terms of the Hamming distance from the node responsible for the set of keywords K and it may take many forwards before reaching it. This clearly translates into an increase of the latency.

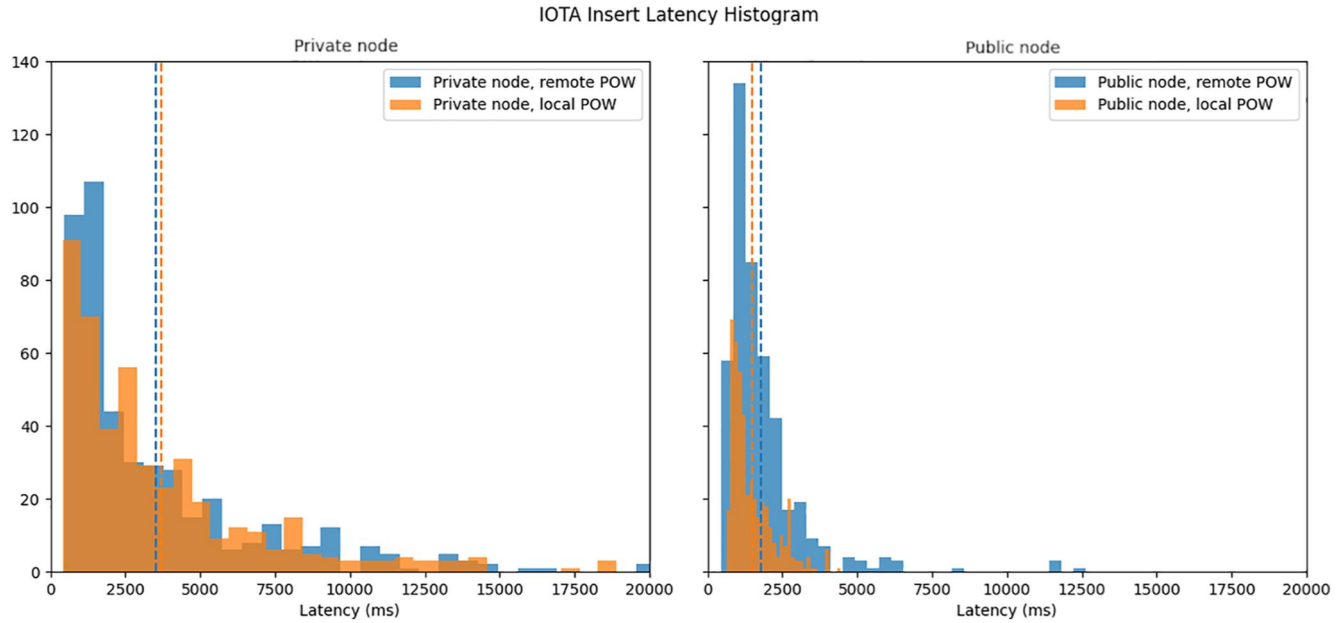

FIGURE 4 IOTA insert operation latency, comparing remote and local POW for the private node. The y-axis shows the number of observations within certain ranges (bins) of latency values. Vertical dotted lines indicate the average values

5.1.4 | Retrieve tests - searching for hazards

The research process is the one that should maintain a high grade of decentralisation while offering an efficient performance and that is what we evaluate here. The process of retrieving the hazards from the system involves two operations performed in sequence, that is, searching the geolocalised data from the Hypercube DHT and then fetching the actual hazard information from the IOTA DLT. As far as the search phase is concerned, a first evaluation criterion is the efficiency of the DHT routing mechanism underlying the keyword search process. In the following, the Pin Search and Superset Search results are reported. Furthermore, we also analyse the fetch latency from the IOTA DLT, once the message id has been obtained from the Hypercube DHT.

- *Number of Hops Results*: As a measure of efficiency, we consider the hops. A hop occurs when a query message is passed from one DHT node to the next. The results for the Pin Search (Figure 6 left) show a similar average hops number 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, that is, $\frac{\log(n)}{2}$ or $\frac{r}{2}$. For instance, with 128 nodes, the experienced average number of hops was around $\frac{\log(128)}{2} = 3.5$. In the case of Superset Search results are different from the previous case. As Figure 6 (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

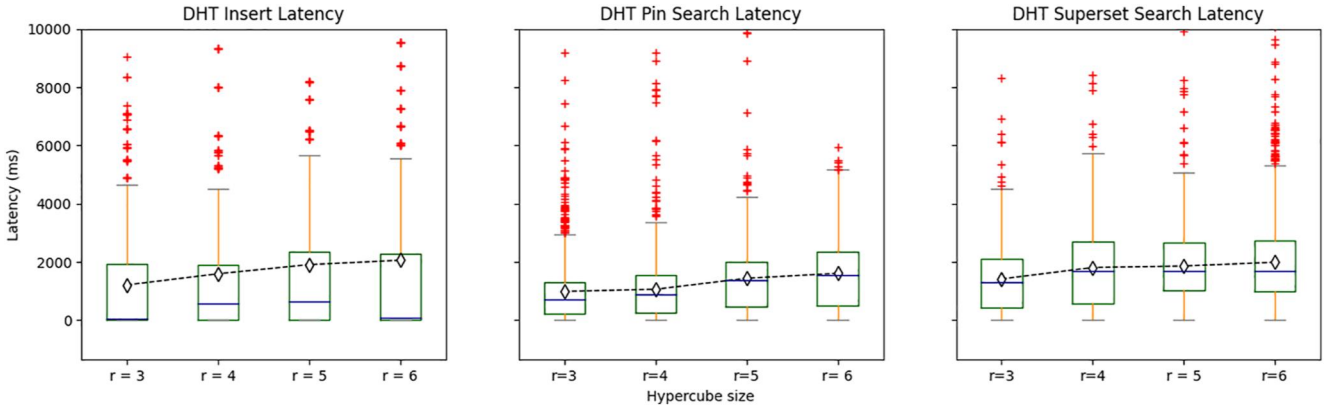

FIGURE 5 DHT Insert, Pin and Superset Search operations latency. Results are reported as box plots and with the plot of a line where the diamond represents the mean value of the overall latency. The rectangle identifies the Inter-Quartile Range (IQR), that is, values from the 25th to the 75th percentile, representing the middle 50% of values. Hence, the lower part of the box (let denote it Q1) is the first quartile (25th percentile), the highest (denote it Q3) is the third quartile (75th percentile). The blue line inside the box is the median value. The lower and upper values identified by the vertical line are the whiskers. In box plots, the whiskers are defined as 1.5 times the IQR. Thus, the lower whisker is Q1 - 1.5*IQR, while the upper whisker is Q3 + 1.5*IQR; they represent a common way to describe the dispersion of the data. Finally, the red '×' symbols outside the whiskers are the outliers

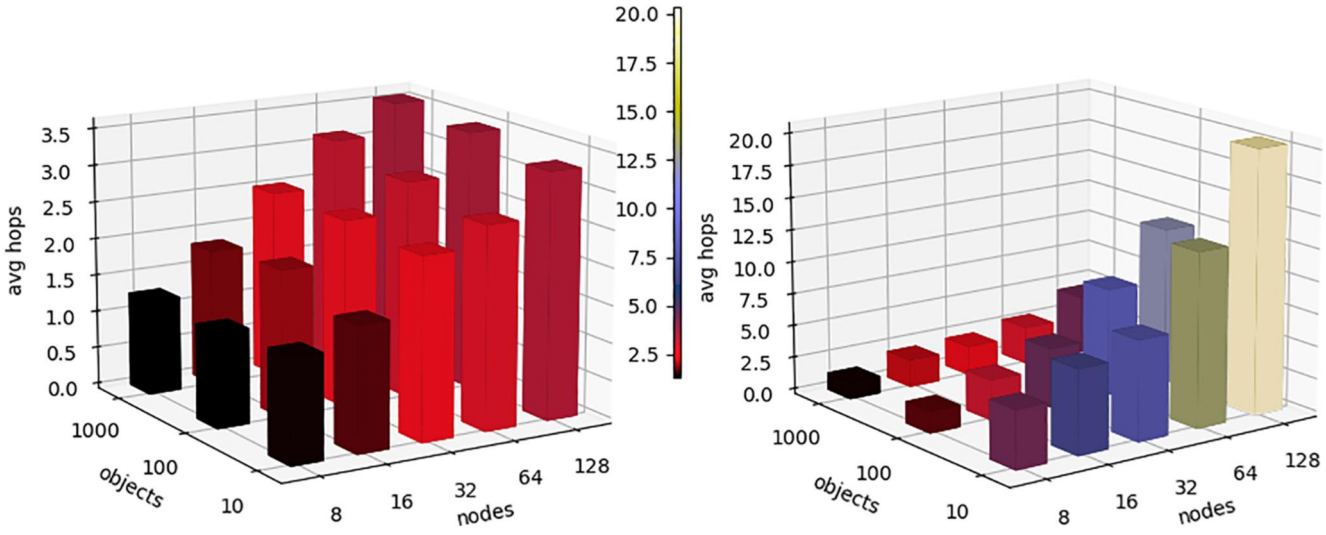

FIGURE 6 Number of hops, that is, when a query message is passed from one Distributed Hash Table (DHT) node to the next. Pin Search results are on the left, Superset Search ones are on the right

maximum is 20.36 for 10 objects and 128 nodes. Theoretically, the average number of hops should be equal to the average hops number required to get to the node responsible for query keywords set K, that is, Pin Search $\frac{\log(n)}{2}$, plus the average hops number to get from that node to all the nodes that include K, until the limit of objects l (or nodes including K) is reached.

- *DHT Retrieve Results*: In Figure 5, it can be seen that, when retrieving objects from the DHT, the increase in the size of the hypercube has a relatively low impact on the overall latency. In fact, in both types of search, despite the number of nodes, the average search time, in a relatively complex situation such as the one assumed in the creation of the scenarios, is low. The Pin Search has a minimum latency value of 1 s for $r = 3$ with an increase to only 1, 6 s when r is doubled (Figure 5, centre). At the same time, however, the results present much more outliers when compared to the Superset Search (Figure 5, right). The Superset Search latencies increase by ∼400 milliseconds with respect to the Pin Search ones.
- *IOTA Retrieve Results*: Similarly to what has been done in the data insert evaluation, that is, assessing the latency of IOTA, the time taken to retrieve data on the Tangle must be taken into account in this search phase. In the retrieve case, the results are much more variable. Moreover, comparing the results of the public node with those obtained using the private node, there is a negligible difference in terms of average latency and variability at the expense of the public one. First of all, the average latency for the private node is smaller than the public node one, that is, ∼3.4 opposed to ∼7.1 respectively. Moreover, the frequency distribution shown in the histogram in Figure 7 presents three spikes with different distances between the private and public nodes cases. For the private node there are recorded frequency spikes for latencies at ∼200 ms, ∼3 s and ∼6.5 s. Public nodes results show spikes at ∼500 ms, ∼3 s and ∼13 s.

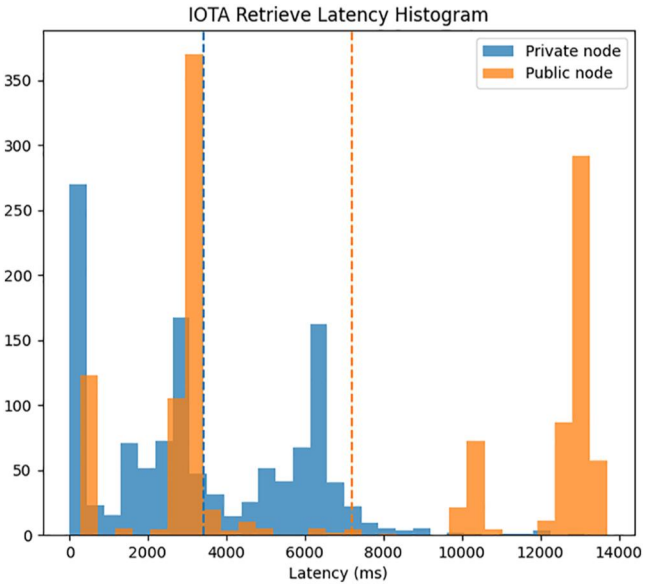

FIGURE 7 IOTA retrieve operation latency, comparing private and public node. The y-axis shows the number of observations within certain ranges (bins) of latency values. Vertical lines indicate the average values

5.2 | DAO smart contracts

For this part of the system evaluation, we implemented the governance layer and the rewarding system described in Section 4. The implemented smart contracts have been developed in Solidity and then stored as Open Source code on Zenodo [48]. The experimentation has been carried out in a local deployed blockchain following the Ethereum protocol.

We developed four smart contracts, namely:

- *DAOToken*: the ERC20 token used to exchange value within the DAO. Anyone holding a certain amount of tokens can invoke the *transfer()* method to transfer part of it.
- *TokenTimelock*: a smart contract for timelocking tokens. It receives an amount of tokens to hold until the *release()* method is invoked. The invocation can happen only when the current time comes after the previously set '*releaseTime*'.
- *TokenTimelockProxy*: a smart contract implementing the EIP-1167 Minimal Proxy pattern [49], that clones an already deployed timelock contract (i.e. the *TokenTimelock*) functionalities by delegating all methods invocations to it. Basically, instead of creating new contracts, it creates a series of proxy objects that refer to the original timelock contract for executing methods.
- *Voting*: a smart contract used for submitting proposals and for voting for those within the DAO. Any DAO member with a certain amount of tokens locked can invoke *submitProposal()* to submit a new proposal. The same or other members can include suggestions in the proposal by invoking *submitSuggestion()*. Then, any member can vote for a suggestion before the execution deadline using the *vote()* method and the vote weight is proportioned to the amount of tokens the member locked. In any of the previous cases, having an amount of tokens locked means also that the lock's *releaseTime* comes before the execution deadline. Finally, the most voted suggestion is picked by invoking *executeProposal()* after the deadline.

In Table 1, we provide the execution cost in terms of gas [9] for the main operations. Gas is a unit that measures the amount of computational effort that takes to execute operations in Ethereum smart contracts. The most expensive operation is the *lockTokens()* function, which 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 [50], each lock request creates a new smart contract that locks tokens for a unique account. However, normally the creation and deployment of such a

TABLE 1 DAO smart contracts operations cost in terms of gas

Smart contract	Operation	Gas usage
DAOToken	Transfer()	51 167
TokenTimelockProxy	lockTokens()	232 024
TokenTimelock	Release()	25 626
Voting	submitProposal()	133 501
Voting	submitSuggestion()	114 523
Voting	Vote()	142 848
Voting	executeProposal()	56 991

smart contract through the Factory pattern would require at least 501818 gas units. By using the EIP-1167 Minimal Proxy pattern [49], that, instead of deploying a new contract each time such as in the Factory pattern, creates proxies that invoke methods of an already deployed contract, we managed to halve the gas usage of *lockTokens()*. Moreover, most of the *Voting* smart contract methods have a quite high cost in terms of gas. This depends on the verifications that these operations must execute, that is, checking that the tokens are locked. This verification, indeed, is costly in terms of gas because it performs a search operation in another smart contract that is, the *TokenTimelockProxy* contract.

6 | DISCUSSION

In this Section, we discuss the results obtained in the tests, starting with the Hypercube DHT performances, then continuing with IOTA and the DAO smart contracts.

- In the case of the DHT, the results shown in the previous section confirm what was expected due to the hypercube structure of the network: (i) that the queries' number of hops is of the order of the logarithm of the hypercube logical nodes number, that is, r; and that (ii) the latency for insertion does not increase linearly with r. In particular, on average Pin Search number of hops is equal to $\frac{\log(n)}{2} = \frac{r}{2}$. For what concerns the Superset Search number of hops, on average, it is equal to $\frac{\log(n)}{2} + l$, where l is the limit of the number of nodes in the sub-hypercube to reach. Those apparently anomalous values stand out, corresponding to a high number of hops between nodes, which decreases with the referenced objects number. This phenomenon can be explained by the fact that the Superset Search traverses the spanning binomial tree of the sub-hypercube induced by the node responsible for the keyword set, until it finds the number of objects indicated by the limit, that is, $l = 10$. Hence, in a network with many nodes and few objects, the query might take longer to reach that limit, because many nodes are 'empty', that is, they do not refer any object. Moreover, it should be noted that the two types of search have a little difference in terms of average latency that is due to the Superset Search limit parameter l, which in this use case indicates the geographical area related to the hazards and to the geoposition of the user executing the query. In general, the Hypercube DHT results show the goodness of the solution in the trade-off between the memory space and response time. In traditional DLTs, such as Ethereum and IOTA, indeed, searching for a datum in a message means traversing all the 'message sea' in the ledger and for this reason, the current solution is to use centralised 'DLT explorers' [51].
- For what does concern IOTA, we need two distinct discussions for insertion and retrieval. As far as the comparison between private and public nodes is concerned in the data insertion, taking into consideration the lowest average latency values, that is, those obtained by opting for remote PoW, it can be seen that the difference between the two is almost negligible. In fact, if on the one hand the public nodes, receiving more messages, are able to provide the tips faster, on the other hand, they may have fewer computational resources to provide to the clients for performing the PoW (because, being public, this node receives more requests to handle). This clearly translates into an increase in latency of the message insertion process. Although the two nodes present a similar average insertion time, in the case of the private node, the variance is much larger. This might be due to the smaller amount of computational resources compared to the public node and maybe to the neighbours connections in the DLT network. In fact, with the same workload, the results of the private node are much more erratic. For what does concern the IOTA data retrieval, our experiments show an unstable latency, probably due to the load suffered by the DLT nodes. Indeed, the results have a frequency distribution with three different spikes (Figure 7), both for the private node and the public one, even if the private one performs better. Generally, latencies for the data retrieval are also higher with respect to insertion. All of this suggests that, since the DLT never stops and nodes always work for inserting new messages, when in presence of a high workload, the IOTA nodes give priority to the message insertion. Indeed, with high workload nodes seem to reply to a retrieve request with a latency of 13–14 s for public and 6–7 for private nodes, that is, latency near the right-most spike in Figure 7. Best case scenario, the response of nodes for messages retrieval are between 200 and 500 milliseconds, that is, latency near the spikes in the middle.
- Finally, regarding the DAO smart contracts implementation, the evaluation has shown interesting results in terms of gas usage. Most of the operations are feasible to execute in a local or permissioned environment, and follow the standard gas usage in respect to the already existing DAO implementations in Ethereum. The only limit seems to be the *lockToken()* method in the *TokenTimelockProxy* contract, but we already halved its gas usage in our implementation. We have not tested the performances in terms of latency for the interactions with the Ethereum public blockchain, since these can greatly vary depending on the transaction fees [52] and/or on the levels of supply and demand in the network [53], but generally we can expect a latency that ranges from 2 to 60 s [52]. Multiplying the

gas usage for its price, we obtain an indication of the actual monetary cost for operating with Ethereum. At the time of writing, executing *lockToken()* would cost ∼120 dollars, and obviously, it does not represent a feasible option in most scenarios. However, this represents a further incentive for the rise of technologies that operate using the same protocol for executing Ethereum smart contracts, but with fewer latencies and a reduced gas price. For instance in Polygon [54], at the time of writing, the *lockToken()* method would cost ∼0.05 dollars. Another possibility is to set up a dedicated permissioned Ethereum blockchain, that would further reduce costs.

7 | CONCLUSIONS

In this paper, we proposed a system based on a Hypercube DHT that provides an efficient content lookup through multiple keyword-based queries and that can be applied to different kinds (or combinations) of DLTs and DFS. In our proposal, we address the problems of DLTs and DFS regarding efficient data lookup and the possibility of implementing complex queries but without reinstating an element of centrality. Our first contribution is the design of such a decentralised system and its implementation using as the underlying data storage a specific DLT, that is, IOTA. Our second contribution is the development of a DAO for the layer-two architecture management and its economic sustainability. With our third contribution, we have shown a case study in which the proposed architecture is used for geodata storing and retrieval based on a road hazards detection scenario.

All the layers of the developed decentralised architecture, that is, (i) data layer, (ii) lookup layer, (iii) DAO organisation, have been validated through an experimental evaluation. First, as concerns data retrieval over IOTA, results show that this technology can be viably exploited to store multiple and variegated data, which can be profitably discovered through the proposed decentralised system. Second, our results show that, as expected, the Hypercube DHT on top of a DLT approach allows for a fast identification of data that satisfy a given keywords-based query. In fact, being r the hypercube dimension (a value in the order of the logarithm on the number of DHT nodes), on average $\frac{r}{2}$ number of hops (i.e. when a query message is passed from one DHT node to the next) are required for a Pin Search, that is, a punctual search based on a specific set of keywords. As concerns the Superset Search, that is, a broader search based on a specific set of keywords and on its supersets, the number of hops depends on the ratio between the limit l assigned to the query and the distribution of objects between nodes. Moreover, in our implementation, the latency for executing the insert and retrieve operations in the Hypercube DHT, runs in sublinear time with respect to r. Third, the use of Ethereum smart contracts enables the possibility of voting for making organisational decisions inside the DAO. Furthermore, the ability to create ERC20 tokens allows for rewarding nodes that have actively contributed to the operation of the P2P system.

A possible extension of our system is to add a 'pay-per-query' model, where node operators within the DAO are rewarded at the level of granularity of the query. Another improvement is to adopt a clever load balancing system to avoid that some nodes might suffer higher workloads due to the popularity of the contents/keywords that they are storing.

ACKNOWLEDGEMENTS

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie International Training Network European Joint Doctorate grant agreement No 814177 Law, Science and Technology Joint Doctorate - Rights of Internet of Everything. This research was also funded in part by the University of Urbino through the 'Bit4Food' research project. We are indebted to Federica La Piana, Alessio Leurini and Davide Tropea for their contribution on a preliminary implementation of the system.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Zenodo at https://doi.org/10.5281/zenodo.5810396 and https://doi.org/10.5281/zenodo.4767755.

ORCID

Mirko Zichichi https://orcid.org/0000-0002-4159-4269
Luca Serena https://orcid.org/0000-0002-7951-4682
Stefano Ferretti https://orcid.org/0000-0002-1911-4708
Gabriele D'Angelo https://orcid.org/0000-0002-3690-6651

REFERENCES

1. Zichichi, M., et al.: Governing decentralized complex queries through a DAO. In: Proc. of the Conference on Information Technology for Social Good (GoodIT), pp. 1–6. ACM (2021)
2. Ramachandran, G.S., Radhakrishnan, R., Krishnamachari, B.: Towards a decentralized data marketplace for smart cities. In: 2018 IEEE International Smart Cities Conference, pp. 1–8. ISC2 (2018). https://doi.org/10.1109/ISC2.2018.8656952
3. Zichichi, M., Ferretti, S., D'Angelo, G.: A Framework Based on Distributed Ledger Technologies for Data Management and Services in Intelligent Transportation Systems. IEEE Access (2020)
4. Park, J.-S., et al.: Smart contract-based review system for an iot data marketplace. Sensors. 18(10), 3577 (2018)
5. Özyilmaz, K.R., Doğan, M., Arda, Y.: IDMoB: IoT data marketplace on blockchain. In: Proc. of the Crypto Valley Conference on Blockchain Technology (CVCBT). IEEE (2018)
6. Ferretti, S., D'Angelo, G.: On the ethereum blockchain structure: a complex networks theory perspective. Concurrency Comput. Pract. Ex. 32(12), e5493
7. Belotti, M., et al.: A vademecum on blockchain technologies: when, which, and how. IEEE Commun Sur & Tutorials. 21(4), 3796–3838 (2019)
8. Santos, J., Santos, N., Dias, D.: Dclaims: A Censorship Resistant Web Annotations System Using Ipfs and Ethereum (2019). *arXiv preprint arXiv:1912.03388*
9. Buterin, V., et al.: Ethereum White Paper (2013). https://github.com/ethereum/wiki/wiki/White-Paper
10. Popov, S.: The Tangle (2016). https://iota.org/IOTA_Whitepaper.pdf
11. Benet, J. Ipfs-content addressed, versioned, p2p file system (2014) *arXiv preprint arXiv:1407.3561*

12. Joung, Y.-J., Yang, L.-W., Fang, C.-T.: Keyword search in dht-based peer-to-peer networks. IEEE J. Sel. Area. Commun. 25(1), 46–61 (2007)
13. Jentzsch, C.: Decentralized Autonomous Organization to Automate Governance. White paper (November, 2016). https://lawofthelevel.lexblogplatformthree.com/wp-content/uploads/sites/187/2017/07/WhitePaper-1.pdf
14. ISO/TC 211. Geographic information - reference model. International Standard. Technical report, vol. 19101. ISO/IEC (2002)
15. Bargiotti, L., et al.: Guidelines for Public Administrations on Location Privacy: European Union Location Framework. Technical report. Joint Research Centre (Seville site) (2016)
16. D'Angelo, G., Ferretti, S.: Highly intensive data dissemination in complex networks. J. Parallel Distr. Comput. 99, 28–50 (2017)
17. Ratnasamy, S., et al.: A scalable content-addressable network. In: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, vol. 31, pp. 161–172 (2001)
18. Aiello, M., et al.: Ippo: A Privacy-Aware Architecture for Decentralized Data-Sharing (2020). *arXiv preprint arXiv:2001.06420*
19. Mehedi Hassan Onik, Md., et al.: Privacy-aware blockchain for personal data sharing and tracking. Open. Com. Sci. 9(1), 80–91 (2019)
20. Bez, M., Fornari, G., Vardanega, T.: The scalability challenge of ethereum: an initial quantitative analysis. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), pp. 167–176. IEEE (2019)
21. Lewis, G., et al.: Sok: layer-two blockchain protocols. In: International Conference on Financial Cryptography and Data Security. Springer (2020)
22. James, B., Baskaran, I., Ramachandran, N.: Authenticating health activity data using distributed ledger technologies. Comput. Struct. Biotechnol. J. 16 (2018)
23. Buterin, V., Vogelsteller, F.: Eip-20: Erc-20 Token Standard (2015). https://eips.ethereum.org/EIPS/eip-20
24. Werner, S.M., et al.: Sok: Decentralized Finance (Defi) (2021). *arXiv preprint arXiv:2101.08778*
25. Guidi, B., Michienzi, A., Ricci, L.: Data persistence in decentralized social applications: the ipfs approach. In: 2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC), pp. 1–4. IEEE (2021)
26. Crabtree, A., et al.: Building accountability into the internet of things: the iot databox model. J. Reliable Intelligent Environ. 4(1), 39–55 (2018)
27. European Commission: European Data Governance (Data Governance Act) (2020)
28. de la Vega, F., et al.: A peer-to-peer architecture for distributed data monetization in fog computing scenarios. Wireless Commun. Mobile Comput. 2018 (2018)
29. Zhu, L., Xiao, C., Gong, X.: Keyword search in decentralized storage systems. Electronics. 9(12), 2041 (2020)
30. Serena, L., Ferretti, S., D'Angelo, G.: Cryptocurrencies activity as a complex network: analysis of transactions graphs. Peer-to-Peer Netw. Appl. 15, 839-853 (2022)
31. The Graph: The Graph Protocol (2020). https://thegraph.com/
32. Community, I.P.F.S.: Search Engine for the Interplanetary File System (2021). https://github.com/ipfs-search/ipfs-search
33. Khudhur, N., Fujita, S.: Siva-the ipfs search engine. In: 2019 Seventh International Symposium on Computing and Networking (CANDAR), pp. 150–156. IEEE (2019)
34. Jiang, P., et al.: Searchain: blockchain-based private keyword search in decentralized storage. Future Generat. Comput. Syst. 107, 781–792 (2020). ISSN 0167-739X. https://doi.org/10.1016/j.future.2017.08.036
35. Zhu, L., Xiao, C., Gong, X.: Keyword search in decentralized storage systems. Electronics. 9(12), 2020. ISSN 2079-9292. https://doi.org/10.3390/electronics9122041
36. Zichichi, M., Ferretti, S., D'Angelo, G.: On the efficiency of decentralized file storage for personal information management systems. In: Proc. of the 2nd International Workshop on Social (Media) Sensing, Co-located with 25th IEEE Symposium on Computers and Communications 2020 (ISCC2020), pp. 1–6. IEEE (2020)
37. Zichichi, M., Ferretti, S., D'Angelo, G.: Are distributed ledger technologies ready for intelligent transportation systems? 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), pp. 1–6. ACM (2020)
38. IPLD Team: Interplanetary Linked Data (Ipld) (2016). https://specs.ipld.io/
39. Guidi, B.: An overview of blockchain online social media from the technical point of view. Appl. Sci. 11(21), 9880 (2021)
40. Guidi, B., Michienzi, A., Ricci, L.: Steem blockchain: mining the inner structure of the graph. IEEE Access. 8, 210251–210266 (2020)
41. Rinckes, D., Bunge, P.: Open Location Code: An Open Source Standard for Addresses Independent of Building Numbers and Street Names (2015). https://github.com/google/open-locationcode/blob/master/docs/olc definition.adoc (accessed Dic 2021)
42. Hamari, J., Sjöklint, M., Ukkonen, A.: The sharing economy: why people participate in collaborative consumption. J. Assoc. Inform Sci & Technol. 67(9), 2047–2059 (2016)
43. Pazaitis, A., De Filippi, P., Kostakis, V.: Blockchain and value systems in the sharing economy: the illustrative case of backfeed. Technol. Forecast. Soc. Change. 125, 105–115 (2017)
44. Zichichi, M., et al.: Likestarter: a Smart-contract based social DAO for crowdfunding. In: Proc. of the 2st Workshop on Cryptocurrencies and Blockchains for Distributed Systems (2019)
45. Distefano, B., Pocher, N., Zichichi, M.: 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), pp. 1–6. ACM (2020)
46. Grinberg, M.: Flask Web Development: Developing Web Applications with python. O'Reilly Media, Inc. (2018)
47. felap96, Zichichi, M.: miker83z/hypercube_vehicles (December 2021). https://doi.org/10.5281/zenodo.5810396
48. Zichichi, M.: HypercubeDAOContracts (May 2021). https://doi.org/10.5281/zenodo.4767755
49. Murray, J.M.P., Welch, N.: Eip-1167: Minimal Proxy Contract (2018). https://eips.ethereum.org/EIPS/eip-1167
50. OpenZeppelin. Openzeppelin Website. https://openzeppelin.com/, 2021
51. Blockchain.com. Blockchain Explorer, 2020. https://www.blockchain.com/explorer
52. Zhang, L., et al.: Evaluation of ethereum end-to-end transaction latency. In: 2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS). IEEE (2021)
53. Spain, M., Foley, S., Gramoli, V.: The impact of ethereum throughput and fees on transaction latency during icos. In: International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
54. Polygon - Ethereum's Internet of Blockchains, 2021. https://polygon.technology/papers/

How to cite this article: Zichichi, M., et al. Complex queries over decentralised systems for geodata retrieval. IET Netw. 1–16 (2022). https://doi.org/10.1049/ntw2.12037