Keyword-based multimedia data lookup in decentralized systems

EMANUELE FAZZINI, University of Bologna, Italy
MIRKO ZICHICHI, IOTA Foundation, Italy
STEFANO FERRETTI, University of Urbino Carlo Bo, Italy
GABRIELE D'ANGELO, University of Bologna, Italy

The use of decentralized systems, such as blockchains and decentralized file storage systems, has the potential to revolutionize various industries by enabling secure and efficient collaboration without intermediaries. However, searching for content in a decentralized system is still an issue. This paper proposes a novel keyword-based decentralized lookup scheme, based on the International Standard Content Code (ISCC), which enables the unique identification of digital content without the need for a centralized registry or authority. To this aim, our approach exploits a hypercube distributed hash table, a distributed peer-to-peer system for storing and retrieving shared resources, and compares different approaches to index contents within it. The results show that the use of ISCC as the basis for creating identifiers enables a more efficient content placement and retrieval, which could pave the way for novel decentralized solutions that surmount the presence of centralized servers for content lookup.

Additional Key Words and Phrases: Distributed Ledger Technology, Decentralized File Storage, Distributed Hash Table, Keyword Search

ACM Reference Format:
Emanuele Fazzini, Mirko Zichichi, Stefano Ferretti, and Gabriele D'Angelo. 2023. Keyword-based multimedia data lookup in decentralized systems. 1, 1 (December 2023), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Recent advances in decentralized systems provide novel ways to share and distribute data without relying on a central server. Decentralized systems distribute control and decision-making among all participants, which can increase transparency, security, and resilience [3]. A currently prominent example of a decentralized system is the blockchain, which is a distributed ledger that records transactions across a network of computers [14]. Other examples of decentralized systems include Peer-to-Peer (P2P) networks, where participants share resources directly with each other, and decentralized autonomous organizations (DAOs), which are organizations that operate through smart contracts on a blockchain and take decisions based on the consensus of their members. Decentralized systems have the potential to revolutionize industries such as finance, supply chain management, and social networking by enabling secure, transparent, and efficient collaboration without the need for intermediaries [7, 9].

Authors' addresses: Emanuele Fazzini, University of Bologna, Italy, emanuele.fazzini@studio.unibo.it; Mirko Zichichi, IOTA Foundation, Italy, mirko.zichichi@upm.es; Stefano Ferretti, University of Urbino Carlo Bo, Italy, stefano.ferretti@uniurb.it; Gabriele D'Angelo, University of Bologna, Italy, g.dangelo@unibo.it.

A relevant problem in decentralized systems arises when looking for content [5]. In a previous work, we already proposed a system for content lookup, based on a Distributed Hash Table (DHT) [13, 15]. A DHT is a distributed system that provides a lookup service similar to a hash table: any participating node can efficiently retrieve the value associated with a given key. DHTs are often used in P2P networks to store and retrieve shared resources. In particular, we resort to a Hypercube DHT, which is based on the geometry concept of the hypercube. In fact, P2P nodes are arranged in a hypercube topology, with each node having a unique identifier that is represented as a binary string. The length of the binary string determines the number of dimensions of the hypercube. The storing and lookup of a resource in the DHT is based on the navigation inside the different dimensions of the hypercube, in order to reach the appropriate node that maintains information about that resource. The main limitation of that solution was that it assumes a naive keyword mapping lookup scheme that can create scalability issues in large decentralized systems.

With this in view, in this paper we propose a novel keyword-based decentralized lookup scheme thought for multimedia content. The goal is to define a similarity content lookup scheme. The idea is to have a way to identify similar documents and present them as potential search results. With this in view, we employ the International Standard Content Code (ISCC). It is a standardized content identification system that enables the unique identification of digital content without the need for a centralized registry or authority. The ISCC system uses a combination of cryptographic hashing and content analysis to generate a unique identifier for any given piece of digital content. This identifier can be used to verify the authenticity and integrity of the content, as well as to help with content discovery and attribution. The key aspect is that through this approach, similar contents should have similar identifiers (ids), and this might ease the lookup process. Another key benefit of the ISCC system is that it is decentralized and open-source, meaning that anyone can use it to identify and verify digital content without the need for a centralized authority or proprietary software. This makes it ideal for use in decentralized systems such as blockchain networks or P2P content sharing platforms, where there is a need for a reliable and standardized content identification system that is not controlled by any single entity.

To assess the viability of the proposal, we set up a set of experiments using classes of images. We compare different approaches to index contents within the Hypercube DHT, with respect to the classic hashing scheme commonly used in DHTs. Results show that the use of ISCC, as the basis for creating identifiers, enables more efficient content placement and retrieval. This eases the discovery of multimedia content, which is distributed over decentralized file storage systems. We claim that these kinds of approaches can pave the way for novel decentralized solutions that surmount the presence of centralized servers for content lookup.

The remainder of this paper is organized as follows. Section 2 provides some background and related work. Section 3 presents the proposed system. Section 4 discusses the experimental evaluation we conducted and the obtained results. Finally, some concluding remarks are provided in Section 5.

2 BACKGROUND AND RELATED WORK

In this section, we provide the necessary background on the main components used to build the decentralized multimedia content discovery scheme, i.e., the ISCC and the Hypercube DHT, a specific type of structured P2P system [13]. Besides our previous work [15], which we extend in this paper, to the best of our knowledge no efforts have been done on content lookup in DLTs and novel decentralized systems.

2.1 Content Lookup in Decentralized Systems

Content discovery in distributed systems has a relevant state of the art [11, 13]. There are two main categories of approaches, i.e. those relying on a client/server system and those based on P2P solutions.

Searching content in a client/server system is quite simple. In this case, in fact, the server has complete knowledge about where contents are located. Thus, clients can ask the server in order to retrieve data.

Things get more complicated when a decentralized system (i.e., P2P) is employed. In this model, nodes are connected to each other without relying on a central server. Two main approaches to content lookup in P2P systems exist, i.e. unstructured and structured.

An unstructured content lookup is a simple approach where peers forward requests to their neighbors without any specific organization or structure [6]. Nodes are usually connected randomly and do not follow any specific topology. In unstructured content lookup, a node typically broadcasts a query message to its neighbors, and each of these neighbors further broadcast the message to their neighbors, until the message reaches the desired content [5]. Unstructured content lookup can be inefficient in large-scale P2P networks, where the number of nodes and the amount of data to be stored can be very large.

Structured content lookup, on the other hand, is a more organized approach that involves the use of Distributed Hash Tables (DHTs) to locate content [12]. DHTs are a type of data structure that distributes data evenly among nodes in the network. Each node is responsible for storing a small subset of the data, and the distribution is based on a predefined key space. In structured content lookup, when a node wants to retrieve content, it first hashes the content's unique identifier to obtain a key. The node then uses the key to locate the node that is responsible for storing the content in the DHT. The node can then retrieve the content from that node directly.

Structured content lookup is more efficient than unstructured content lookup in large-scale P2P networks because it allows for fast and efficient content retrieval without the need for extensive message flooding [10]. In our system, we resort to a specific DHT shaped as a hypercube, which we describe in the next subsection.

2.2 Hypercube

The Hypercube DHT is a structured P2P network that organizes nodes in a hypercube topology [8]. This topology allows for efficient lookup and retrieval of data among nodes [15]. In a hypercube DHT, nodes are assigned unique identifiers that are represented as binary strings of a fixed length. These nodes are responsible to maintain information about specific contents, which are identified through ids mapped in the same key-space of node ids. In essence, the key concept is how to define the way in which contents are assigned identifiers. The common solution is to compute the hash of the desired data object to obtain a binary string that corresponds to the id in the DHT. In this case, the content lookup corresponds to identifying the node whose id is closer to the hashed identifier. This solution works if the content to look for is known, i.e. the user does have the actual data, but he/she perfectly knows which data is needed.

When there is the need to make more complex queries to the system, some alternative solution is needed. For example, a user might want to locate images of a specific location (e.g. the Colosseum in Rome, Italy). In this case, he does not know which is the file to retrieve, but he knows a specific keyword associated with the type of the image he/she needs (e.g. `location:Rome`, `subject:Colosseum`). In this case, content indexing can be performed based on the keywords associated with the metadata of the content. This is the goal we pursue in our system. In the following of this section, we review the approach we already employed in [15].

ID type	Description
Content id, xID	It is the identifier for the content to be retrieved in the decentralized storage. It can be the CID in the case of IPFS, a DLT transaction ID, etc. In the DHT, it is the *value* of the <*key,value*> mapping.
Node id, v	It represents a vertex of the hypercube DHT. It is encoded as a r-bit string. This logical node is associated with a physical node in the distributed system that maintains pointers to where data with certain characteristics, i.e., set of keywords, are physically stored and retrievable.
Keyword set id, rID	It represents an identifier for a certain set of keywords K describing given contents. Thus this data aggregates in one single information different keywords. It is encoded as a r-bit string in the same space of node ids.

Table 1. Summary of the different identifiers employed in this work.

To sum up, we are looking for a method that, based on a certain set K of keywords exploited to perform a query, computes an identifier that describes a certain typology of contents (e.g. images of the Colosseum in Rome). This leads to the recognition of rID as the id for that query, which is used to locate the node v in the DHT that has knowledge about all the contents resolving the query and stored in the decentralized storage in use. Thus, the node will answer with all the references to these contents available in the decentralized storage. Clearly, these references are, in turn, other identifiers xID to locate contents (that have nothing to do with the DHT ids).

To avoid any possible confusion, Table 1 reports a description of the different identifiers which are involved in our system. Consider the xID as a generic identifier for a particular content in the decentralized storage. For instance, xID might be the CID for an IPFS Object or a specific DLT transaction identifier. xID is associated to a keyword set K describing the content resolved by xID. We assume that these keywords belong to a domain $K \subseteq W$.

Now, a uniform hash function $h : W \longrightarrow \{0, 1, \ldots, r-1\}$ is employed to map the keyword set K to a keyword set id rID. In particular, for each $k \in W$, $h(k)$ sets to 1 one specific bit of the r-bit string given by $mod_r(h(k))$. (In other words, each $k \in W$ has an assigned position in the r-bit string). Thus, the rID related to the keyword set K is thus generated as a r-bit string where the positions are "activated" (i.e., set to 1), by all the $k \in K$, i.e. $one(rID) = \{mod_r(h(k)) \mid k \in K\}$.

These r-bit strings not only represent a keyword set K, but they are used to identify logical nodes in the Hypercube DHT network. For example, if we fix the size of the r-bit string to $r = 4$, then node ids can take binary values in the range from 0000 to 1111. We can formally define a r-dimensional hypercube $H_r(V, E)$ as a set of vertices V and a set of edges E connecting them. Each of the 2^r vertices represents a logical DHT node, while edges are formed when two vertices ids differ by only one bit, e.g., 1011 and 1010 share an edge. The distance between two vertices u and v can be measured using the Hamming distance, i.e., $Hamming(u, v) = \sum_{i=0}^{r-1}(u_i \oplus v_i)$, where $\oplus$ is the XOR operation and u_i is the bit at the i-th position of the u string, e.g., for $u = 1011$ and $v = 1010$, we have $Hamming(u, v) = 1$.

In the Hypercube DHT, contents can be discovered through queries based on the lookup of keyword set id rID, which corresponds to a point in the hypercube. Contents are in fact stored on the node u with the identifier that is closest to rID (recall that node ids and resource ids are mapped into the same key-space). The query takes as input a keyword set id rID and, starting from a random node v, the request is propagated in the network until node u (that is responsible for that keyword set rID) is reached. This basic search will return all and only the xIDs exactly associated with a keyword set K, i.e., $\{xID \in D \mid K_{xID} = K\}$, maintained by the responsible node u (pin search). What might happen, however, is that one could be interested also in contents that are described by keyword sets that include K, i.e., $\{xID \in D \mid K_{xID} \supseteq K\}$ (superset search). Thus, it is also possible to ask for contents not only at the u node, but at its neighbours (in the hypercube) as well. Clearly enough, in this case a limit l is set to the number of returned results obtained by all the nodes with id associated to the keyword sets $K_{xID} \supseteq K$.

2.3 International Standard Content Code

The International Standard Content Code (ISCC) [2] is an ISO-approved standard that, given an input file, gives the ability to create a corresponding code that goes to identify the file itself. Generating the code consists of employing a set of content-driven, locality-sensitive, and similarity-preserving hash functions. Unlike cryptographic hash functions, these hash functions aim to preserve similarity between data, so that two similar contents do not have totally different codes. This process consists of four main components.

- Meta-Code: encodes metadata similarity, e.g., *AAA5C73C3GZDHDHD*[1];
- Content-Code: encodes perceptual or syntactic similarity of contents, e.g., *EEA2T2CQVF6ZR7BU*;
- Data-Code: encodes raw bitstream data similarity, e.g., *GAACDVmDpTfvWZfP*;
- Instance-Code: encodes checksum for data integrity, e.g., *IAACRtoh1WeiDvEi*.

These components can be considered separately, all together, or used to generate a code represented by a digest derived from the four components. This digest is composed of 52 characters, totaling 36 bytes (288 bits), and it is the result of a base32 function applied to the four components. (In this work, we will use Meta-Code and Content-Code, only.)

Each component consists of 72 bits, 8 bits for the header, and 64 bits for the body, and has as its return value a string encoded in base58-iscc [1]. The header is intended to recognize the type of code component and, in the case of Content Similarity Code, to indicate the file type from four main choices: text, audio, video, and image.

An example of a research work that uses the ISCC standard to build a similarity-based lookup scheme for multimedia content in decentralized systems is presented in [4]. That paper reports a comparison between the use of an ISCC based indexing and a classic hashing scheme commonly used in DHTs. The results show that the use of ISCC, as the basis for creating identifiers, enables more efficient content placement and retrieval, with respect to a traditional approach based on standard hash functions employed in other DHTs. This is basically what we are going to do, but coupling the indexing approach with the use of a Hypercube DHT for the retrieval of contents in decentralized storage.

3 MULTIMEDIA MULTIPLE KEYWORDS SEARCH

The deployment of a real decentralized file storage system, with multimedia contents to be indexed, leads to the need for designing a proper similarity content based indexing service, so as to promote an effective content lookup. What is needed, in essence, is an indexing scheme that is more sophisticated than a simple keyword-value based index, such as the one proposed in [15]. We thus envision a system that allows for multimedia content retrieval, not only thanks to one (or a set of) keyword, but based on metadata and data contents. With this goal in mind, in the following, we

[1]The example codes reported here are just reference examples that we will use in the rest of the paper.

describe and evaluate some alternative decentralized indexing approaches based on the hashing of the content itself, or on the use of the ISCC standard and the hashing of its metadata [2]. In this section, we first present the reference decentralized system model and secondly the indexing schemes we will evaluate in the rest of the paper.

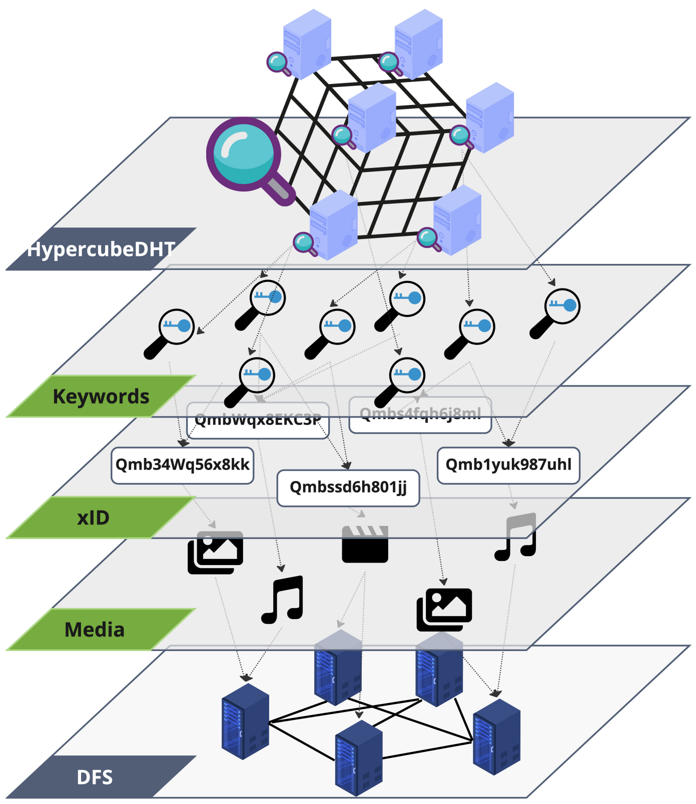


Fig. 1. Multi-level architecture of the system model.

3.1 System Model

The system architecture is based on two layers: the hypercube DHT network works on top of the DFS network, i.e., IPFS (see Figure 1). The underlying layer is the one where the contents are stored, while the upper layer is where contents are indexed. To properly work, there are three different kinds of information that are maintained by the system, i.e. the actual media contents, their identifiers, and the metadata (e.g., keywords) associated with them. Thus, the Hypercube DHT maintains an association between keywords K and the related content identifiers xIDs, i.e., a CID in IPFS. These identifiers are used to retrieve the actual data, i.e., multimedia content stored in the DFS.

3.2 Indexing scheme

The goal of the indexing scheme is to minimize the number of hops required to search contents through the Hypercube DHT, while automatizing the creation of keywords for multimedia content. In other words, we are considering the opportunity of having a decentralized lookup scheme that is able to cluster similar contents into near peers in the

Starting code	**Key set id generation scheme**	
	OR	concat
SHA-256	*SHA-OR*	*SHA-concat*
ISCC Meta	*ISCC-M-OR*	*ISCC-M-concat*
ISCC Content	*ISCC-C-OR*	*ISCC-C-concat*
ISCC Content OR Meta	*ISCC-CM-OR*	*ISCC-CM-concat*

Table 2. Different indexing schemes considered in this evaluation.

logical overlay. Our aim is to identify the best scheme that maps similar contents to similar keyword set ids, so that nearby nodes in the Hypercube DHT maintain pointers to similar contents located in a decentralized storage or ledger. In this work, we compare two different approaches, i.e., one based on the use of a traditional hash based function and some variants of an approach based on the use of ISCC standard.

3.2.1 The SHA-256 method. The first approach is the use of a traditional cryptographic hash function based scheme that, given content, computes the hash of the considered information and takes it to generate a keyword set id. Thus, given the content, this method produces a r-bit string that is then used in the approaches described in Section 3.2.3 to produce a final keyword set id. Hereinafter, we refer to this indexing scheme as *SHA-256*, which is the typical employed hashing function.

3.2.2 The ISCC methods. The alternative approach exploits the ISCC standard (hereinafter referred as the *ISCC* method). In particular, we consider different combinations of the ISCC Meta-Code and Content-Code associated to multimedia content:

- *ISCC-M* - only the Meta-Code associated with the metadata is employed to obtain the keyword set id used in the hypercube DHT.
- *ISCC-C* - only the Content-Code is used to obtain the keyword set id.
- *ISCC-CM* - both Meta and Content codes are computed and then their keyword set ids are combined through an OR operation (Meta-Code OR Content-Code).

3.2.3 Keyword Set Id generation method. Given a (hexadecimal base representation of a) r-bit string s, generated using one among the *SHA-256* or *ISCC* methods described above, we further manipulate s to generate the final keyword set id. Two (slightly) different strategies were adopted for the generation of this id.

- **OR-based indexing**. s is subdivided into a sequence of chunks of size g. For each chunk c_i, we compute its modulo $c_i \bmod r$, which identifies a position in the r-bit string that must be set to 1. By repeating this procedure for all chunks, we obtain a final bit string by OR-ing all the modulo operations for all the chunks.
- **Concatenation-based indexing** Given s expressed in hexadecimal format, every single character c_i is considered (i.e., with respect to the approach above, we set $g = 1$). Then, the final keyword set id is built through the concatenation of the result of $c_i \bmod r$, for all the c_i.

Table 2 shows the names we assigned to all the variants of the employed methods, based on the use of *SHA-256* or *ISCC*, and on the specific keyword set id generation method.

4 EXPERIMENTAL EVALUATION

This section is devoted to presenting how the different indexing schemes have been evaluated and the obtained results.

4.1 Metrics of Interest

The goal of the study is to assess which set of parameters and keyword set id generation technique is able to maximize the distance between classes of contents, while minimizing the internal distance between contents belonging to the same class. (As reported before, since we are dealing with bit-string ids, the distance we consider in this case is a classic Hamming distance.) Thus, for each class c, we measure a type of clustering index (*CI*) that is calculated as the average distance between the c-centroid and the centroids of all other classes, over the average distance of items in c with respect to the c-centroid. Finally, we use the average of these values (referred as different classes), to obtain a single measure for the considered scheme and parameters setting. The metric *CI* that is introduced can be considered as a variation of the Dunn index, which is the ratio between the minimum inter-cluster distance and the maximum intra-cluster distance. Clearly enough, the higher the value of *CI* the better it is.

4.2 Experimental Results

The test dataset was composed of 30 classes, each containing 15 photos of the same subject, like a monument or a painting.

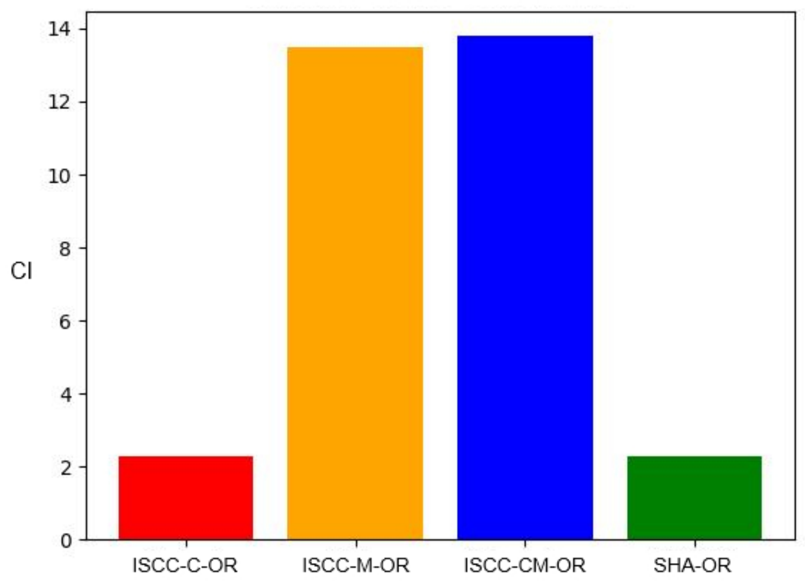
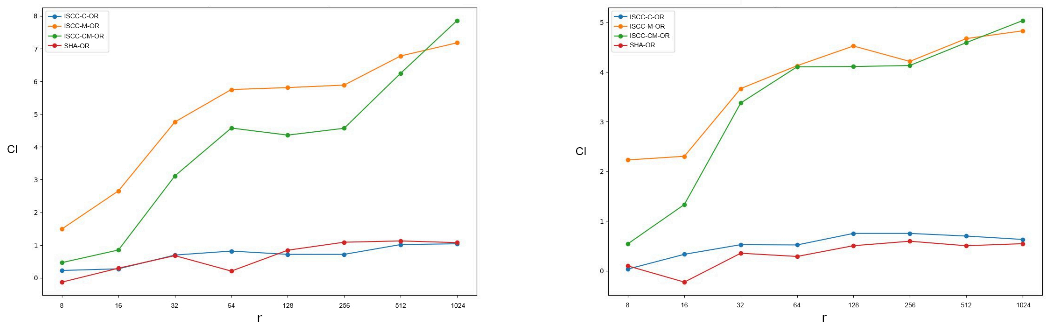


Fig. 2. Clustering Index (*CI*) for *OR-based* methods. The metrics refer to the ability of the method to cluster similar multimedia contents on the same nodes of the Hypercube DHT. Thus, the higher the better.

Figure 2 shows the average results when using the *OR-based* indexing method. It is possible to appreciate how the *ISCC-M-OR* and *ISCC-CM-OR* indexing schemes, based on the use of ISCC meta-codes, perform better than others. In particular, *ISCC-CM-OR* slightly outperforms *ISCC-M-OR*. This result might not be surprising, since *ISCC-CM-OR* uses more information than *ISCC-M-OR* to index contents. Indeed, adding more information should improve the ability to differentiate between different classes of multimedia content. However, this improvement seems only marginal in these experiments.



(a) g=2 (b) g=4

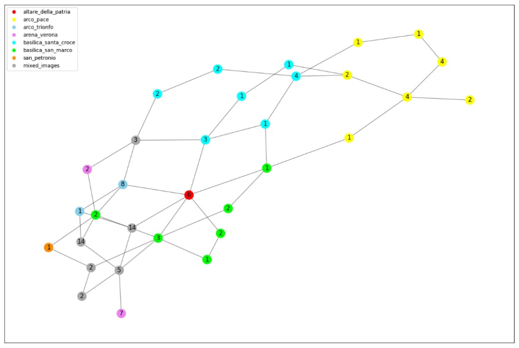
Fig. 3. Clustering Index (*CI*) for *concat-based* methods, with different g and r settings. The metrics refer to the ability of the method to cluster similar multimedia contents on the same nodes of the Hypercube DHT. Thus, the higher the better.

Results for the *concat-based* methods are shown in Figure 3. In particular, Figure 3a shows the results obtained from different values of the r parameter, with g kept fixed and equal to 2, while Figure 3b provides results when $g = 4$. In both plots, it is possible to notice that the use of Meta-Code, i.e. *ISCC-M-concat*, is the best choice for allocating images on the DHT. In fact, this is the methodology that guarantees better performance in terms of maximizing the distance between different classes and at the same time minimizing the distance between contents of the same class.

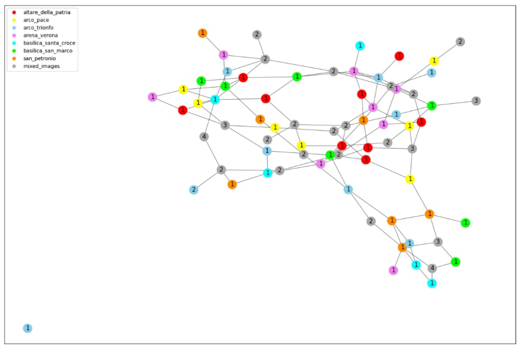
In this experiment, *ISCC-CM-concat* that employs both Content-Code and Meta-Code has a similar, but slightly worse, behaviour with respect to the *ISCC-M-concat*. This result suggests that the most important information that allows separating classes is the Meta-Code. From the results shown in the figure, it seems that the configuration with the parameter $g = 2$ provides slightly better performance, with respect to $g = 4$, but the improvement is actually limited.

An interesting thing to notice is that the best result obtained with the *OR-based* method, i.e. *ISCC-CM-OR*, is almost twice as much as the highest value obtained with the *concat-based* method, i.e., *ISCC-CM-concat*. This allows us to conclude that *ISCC-CM-OR* performs better in terms of the distribution of multimedia content on the DHT.

Figure 4 shows how the images were allocated over the hypercube nodes, in a configuration with a number of bits $r = 8$, respectively for *ISCC-CM-OR* (Figure 4a) and the typical *SHA-256* (Figure 4b). What each chart shows is a part of the Hypercube DHT. Each node of the graph corresponds to a node in the DHT. Each node is coloured based on the images it maintains. (To be more precise, it maintains pointers to where images are located.) Thus, for instance, a node coloured in orange means that it maintains images related to the “Altare della Patria” (a popular monument in Rome, Italy). Different colours pertain to different classes of images, with the exception of the grey colour, which represents the fact that the node contains (pointers to) images of different classes. Moreover, each node has a weight (label) associated with it, which represents the number of contents it stores. Simply put, what we would like to see here is a graph of nodes, with a high weight, which are highly connected when they have the same colour and with no nodes coloured in grey. It is possible to notice that *ISCC* creates more effective clustering for images of the same class. The nodes containing the images of the same class stay closer, and the ones containing images of different classes are



(a) ISCC allocation.



(b) SHA allocation.

Fig. 4. Subgraph of DHT nodes containing references to images of different classes. Links represent direct connections among nodes of the DHT. The more nodes with same the color are linked together, the better it is.

far apart. Instead, as expected, *SHA-256* causes an allocation of multimedia contents that is spread over the hypercube, causing a higher number of hops in content lookups over the DHT, and consequently performing worse in terms of returned multimedia contents and lookup delay. More in detail, it is worth mentioning that both schemes were applied to DHTs of the same size. The fact that *ISCC* subgraph has fewer nodes shown in the figure, w.r.t. *SHA-256*, means that contents were actually allocated in fewer nodes in the DHT, thus leaving other nodes not utilized. In our view, this is not a problem, but rather a benefit, since as mentioned the allocation results are more clustered. In a situation with a higher amount of multimedia content, our results suggest that a *ISCC Meta-Code* method guarantees that different classes of contents would be managed by nearby DHT nodes, with different classes going on different portions of the DHT. Conversely, *SHA-256* allocation does the job it is supposed to do, i.e. it distributes contents (even those of the same class) throughout the DHT. Few images are allocated in each DHT node. Indeed, several of these nodes have only one image associated, as reported in the node weights. This leads to a significant number of hops to lookup for a content type and a fewer number of returned results correlated with the query.

5 CONCLUSIONS

Decentralized systems have the potential to transform modern (yet, still centralized) services by enabling secure, transparent, and efficient interactions without the need for intermediaries. However, a major challenge in decentralized systems is content lookup. In this paper, we propose a novel keyword-based decentralized lookup scheme based on the International Standard Content Code (ISCC). Our experiments show that using the ISCC as the basis for creating identifiers enables a more efficient content placement and retrieval compared to the classic hashing scheme commonly used in DHTs. Overall, our proposal can pave the way for novel decentralized solutions that overcome the limitations of centralized servers for content lookup, making decentralized systems more scalable, efficient, and user-friendly.

REFERENCES

[1] [n.d.]. Base-ISCC. https://iscc.codes/specification/#base58-iscc.
[2] [n.d.]. ISCC. https://iscc.codes.
[3] Lijun Chen, Wei Yu, Jing Li, Shangguang Yang, and H Vincent Poor. 2021. A distributed deep learning approach for intelligent multimedia content analysis in the internet of things. *IEEE Internet of Things Journal* 8, 7 (2021), 5234–5245.

[4] M Couceiro, LR Nunes, and JS Silva. 2020. A similarity-based lookup scheme for multimedia content in decentralized systems. In *2020 International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 468–474.

[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] Stefano Ferretti. 2013. Gossiping for resource discovering: An analysis based on complex network theory. *Future Generation Computer Systems* 29, 6 (2013), 1631 – 1644. https://doi.org/10.1016/j.future.2012.06.002

[7] 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.

[8] Richard M Karp and Scott Shenker. 2000. A randomized algorithm for finding frequent elements in streams and bags. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. ACM, 163–174.

[9] Galia Kondova and Jörn Erbguth. 2020. Self-sovereign identity on public blockchains and the GDPR. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. 342–345.

[10] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. 2001. A scalable content-addressable network. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*. 161–172.

[11] Suman Sankar Roy and Sajal K Das. 2018. Content lookup in peer-to-peer networks: A survey. *Journal of Network and Computer Applications* 105 (2018), 35–56.

[12] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking* 11, 1 (2001), 17–32.

[13] Syed Wasiq, Syed Hasan Adil Bukhari, and M Asif Halepoto. 2020. Distributed image retrieval in a decentralized network using content-addressable networking. *Multimedia Systems* 26, 4 (2020), 445–461.

[14] M. Zichichi, S. Ferretti, and G. D'Angelo. 2020. A Distributed Ledger Based Infrastructure for Smart Transportation System and Social Good. In *2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*. 1–6.

[15] Mirko Zichichi, Luca Serena, Stefano Ferretti, and Gabriele D'Angelo. 2022. Complex queries over decentralised systems for geodata retrieval. *IET Networks* (2022). https://doi.org/10.1049/ntw2.12037