Computer Networks 234 (2023) 109911

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Survey paper

How to decentralize the internet: A focus on data consolidation and user privacy

Ted "Taekyoung" Kwon [a,*], Junghwan Song [a], Heeyoung Jung [b], Selin Chun [a], Hyunwoo Lee [c], Minhyeok Kang [a], Minkyung Park [a], Eunsang Cho [a,1]

[a] *School of Computer Science and Engineering, Seoul National University, 1, Gwanak-ro, Gwanak-Gu, Seoul, Republic of Korea*
[b] *Electronics and Telecommunications Research Institute, 218, Gajeong-ro, Yuseong-gu, Daejeon, Republic of Korea*
[c] *Department of Computer Science, Purdue University, Lawson 2142U, West Lafayette, 47906 IN, United States*

ARTICLE INFO

Keywords:
Privacy
Internet
Network architecture and design
Distributed networks

ABSTRACT

Over the years, the Internet has become a field in which a small number of large Internet companies dominate most of the Internet services. As users get used to using their services, the users' generated content and the data about their online behaviors are concentrated in such companies. This phenomenon, called "*data consolidation*", has become a serious problem, which makes the Internet society seek to *decentralize* the current Internet. The decentralized Internet aims to (i) prevent the concentration of user data in a few giant companies like Google and Facebook, and (ii) give users full ownership and control of their data. Various technical solutions that address the data consolidation problem have been proposed; however, those solutions focus on somewhat different scopes of the problem often from their limited viewpoints. The main contributions in this paper are the following. First, we survey the solutions relevant to Internet decentralization based on the following criteria: *data consolidation*, *data ownership*, and *the privacy of user data*. Second, we suggest a holistic reference framework from a functional viewpoint, while the prior proposals in the literature handle a limited set of requirements. Last, we seek to identify remaining research issues, considering additional requirements that have not been addressed in the existing solutions.

1. Introduction

The Internet has been increasingly pervasive in our daily lives, which becomes the most critical infrastructure globally. The number of Internet-connected devices was estimated to exceed 20B in 2019 [1]. The number of the Internet users exceeds 60% of the total population on earth in 2020 [2].

While increasingly more people and devices are connected to the Internet, the Internet ecosystem has shown the symptoms of the mature markets, such as market concentration and fewer opportunities for market entry. For instance, Facebook is dominant in online social networks; Amazon takes almost 50% of the e-commerce market in the US. Overall, a few giant Internet companies monopolize the Internet ecosystem [3]; this phenomenon of concentration is called ***consolidation*** (e.g., [4]). Such a trend of power concentration in the Internet economy is not desirable since it obstructs fair competition and degrades the soundness of the Internet ecosystem.

What would be the ramifications of the Internet consolidation? In the case of search engines, Google is estimated to account for more than 90% of the global search engine market [5]. The Chrome web browser developed by Google is used by the majority of Internet users [6]. Thus, Google can collect users' search queries from its search engine and users' browsing histories from Chrome. Therefore, online activities of billions of Internet users can be analyzed by Google. Facebook, including Facebook Messenger, WhatsApp, and Instagram, dominates online social networks, whose users voluntarily post and share their personal data [7]. Facebook can thus collect users' posts, social relations, and chats. Many things can be done by analyzing such user data. For instance, Google and Facebook can each find out individual users' preferences and tastes, and hence profile each user. User profiling is widely used in many business models, such as targeted advertising. Indeed, the two companies account for more than 80% of the online advertising market [8]. Such a monopoly of the Internet services leads

* Corresponding author.
E-mail addresses: tkkwon@snu.ac.kr (T.T. Kwon), jhsong@mmlab.snu.ac.kr (J. Song), hyjung@etri.re.kr (H. Jung), slchun@mmlab.snu.ac.kr (S. Chun), hwlee2014@mmlab.snu.ac.kr (H. Lee), mhkang@mmlab.snu.ac.kr (M. Kang), mkpark@mmlab.snu.ac.kr (M. Park), escho@mmlab.snu.ac.kr (E. Cho).
[1] Currently affiliated with sPresto, 10th floor, 174, Donggyo-ro, Mapo-gu, Seoul, Republic of Korea.

https://doi.org/10.1016/j.comnet.2023.109911
Received 13 September 2021; Received in revised form 3 June 2023; Accepted 26 June 2023
Available online 5 July 2023

to the concentration of user data, referred to as ***data consolidation***, which is the focus of this paper.

The data consolidation problem is recognized by many Internet communities and civil organizations. In particular, the Internet Society (ISOC) has recently published a report [4], which points out the trend of consolidating user data as one of the most pressing issues to be addressed. As our society heavily relies on the Internet and its services, data consolidation leads to crucial social and economic problems.

First, the user data, one of the most valuable Internet elements, is concentrated on a few big Internet companies. The user data is fundamental to develop technical platforms and applications in the era of the fourth industrial revolution [9,10]. As the oil is vital in a large swath of the current industries, digital data will be the prime resource for numerous innovative and disruptive online services. For instance, data are used to feed the algorithms for machine learning and other AI services. As the amount of the user data increases, the performance of machine learning-based services is improved substantially. Thus, the giant Internet companies monopolizing the user data are expected to be the "data barons", like the oil barons. The imbalance of the amount of data under control also results in unfair competition among Internet companies [4]. If most of the user data is concentrated in a few giant companies who keep the data exclusively, other companies (in particular, start-ups) cannot devise new or better services exploiting the user data. It may lead to a vicious cycle that worsens data consolidation further.

Second, the user data "belongs to" service providers as well as users who create the data. The user data produced by a user should be fully owned and controlled only by the user, even if the data is used for a particular service to which she subscribes. However, for most Internet services, the user data is (i) typically stored in the service provider's systems, (ii) unrestrictedly processed by the service provider, and (iii) often transferred to third parties, while the owners usually being unaware of such activities. Thus, Internet companies utilize user data for user profiling, targeted advertising, training machine learning algorithms, and so on. The problems are (i) users cannot get rewards for their data, and (ii) users do not know how and where their data is processed and transferred, respectively.

Third, the privacy of the online user is not protected. Every activity or content generated by the user is handled by the above giant companies in exchange for their services provided usually for free. However, most users are unaware that their data is under the complete control of the giant Internet companies. Besides, the service providers may collect not only data directly uploaded by users but also user behavior patterns (e.g., search keywords, browsed websites) collected by their websites and applications. This means that users' privacy is not protected at all if there are no countermeasures.

Recently, to address the adverse effects of data consolidation and the concerns of privacy breaches, some legal regulations have been enacted. EU establishes the General Data Protection Regulation (GDPR) for data protection and privacy for all individual citizens in the EU, which became effective in 2018 [11]. The GDPR aims primarily to give control to individuals over their personal data and hence contains provisions and requirements related to the processing of personal data. Other countries also seek to protect user data through legislations like data localization [12], which forces Internet companies to locate their servers storing user data in the same country of users.

Aside from the regulation, the Internet technical community has proposed many technical approaches to address data consolidation. For instance, InterPlanetary File System (IPFS) [13] is designed to store data in a decentralized fashion leveraging peer-to-peer networking. Decentralized Identifier (DID) [14] seeks to remove a central authority in issuing identifiers; in a DID document, an ID created by an entity as well as her public key is published using distributed ledger technologies (e.g., blockchain).

The decentralized operations on the Internet are not new at all. Note that one of the main objectives in creating the Internet was to avoid the problem of centralized networking: a single point of failure [15,16]. Thus, we believe the current trend of consolidation goes against the design philosophy of the Internet. In this sense, it is worthwhile to review what kinds of decentralized networking approaches have been proposed and are on the horizon.

In summary, data consolidation is widely recognized as a critical problem in the current Internet, which both legal regulations and technical solutions have tried to address. However, there are limitations to both approaches. Legal regulations have focused on how to prevent large companies from misusing or exporting user data; however, they are reactive in nature and usually come into effect after a violation has occurred. Thus, legal regulations may not be sufficient to address the whole consolidation problem. Also, existing technical solutions have not provided a whole picture for decentralizing the Internet, as individual solutions have been developed to address specific issues rather than the entire framework. Consequently, there is currently no comprehensive framework that encompasses existing technical solutions.

In this paper, we propose a reference framework for the decentralized Internet that considers networking functions as well as user and system requirements such as privacy protection and incentivization. Our framework aims to address the limitations of existing solutions by providing a holistic perspective on decentralization. Through this approach, we try to contribute to the development of a decentralized Internet that is more secure, available, and user-centric.

We make the following contributions in this paper:

- We investigate the problem of data consolidation in the current Internet.
- We survey representative technical proposals that seek to mitigate the problems of centralized Internet. Then, we analyze which proposal provides which functionality. We also identify which functionality should be added or improved to move towards the decentralized Internet.
- We suggest the reference framework in which the above functions are classified into modules from a holistic perspective.
- Lastly, we present open issues and areas that need further research. To this end, we analyze the functions of the proposed reference framework that are not met by existing technologies.

This paper has the following structure. In Section 2, we briefly introduce the most representative proposals in the literature on the decentralized Internet. In Section 3, based on the analysis in Section 2, we provide a taxonomy to analyze the decentralized applications and systems holistically. In Section 4, we raise further research issues to be addressed. Finally, in Section 5, we conclude with our observations and findings.

2. Existing technical solutions for decentralization

Currently, Internet-based content services are mostly centralized, and hence user data is concentrated on the servers of the service operators. On the other hand, in the recently proposed decentralization solutions, data consolidation problems are mitigated by storing user data in a decentralized fashion. In this section, we review recent representative solutions for the decentralized Internet.

2.1. InterPlanetary file system (IPFS)

IPFS [13] is a renowned open source decentralized storage solution by Protocol Labs. Since its initial version in 2015, IPFS developers claim that the current content services have the following problems: hyper-centralization of services, inefficient content delivery, low resiliency, etc. To mitigate these problems, IPFS is developed as a global-scale decentralized storage system leveraging peer-to-peer (P2P) networking.

In building the global-scale P2P file distribution system, there are at least two points to address. First, how to find the requested file

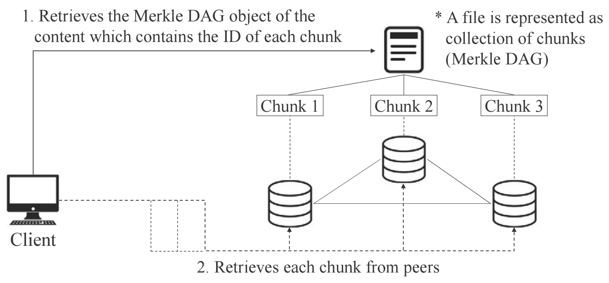

1. Retrieves the Merkle DAG object of the content which contains the ID of each chunk
* A file is represented as collection of chunks (Merkle DAG)
Chunk 1
Chunk 2
Chunk 3
Client
2. Retrieves each chunk from peers

Fig. 1. In IPFS, a file is divided into smaller chunks, and these chunks are aggregated to the original file using the Merkle DAG. The user first requests the file then receives its chunk IDs. Finally, the user requests each chunk by its chunk ID.

or the peer holding the file? Second, how to deal with peer churning in P2P networking? To solve such issues, IPFS has combined some of the successful prior mechanisms, including Distributed Hash Table (DHT) [17–20], BitTorrent [21], and Merkle Directed Acyclic Graph (DAG) [22].

In IPFS, a P2P network is built based on a Kademlia DHT [23], which has been widely used in P2P distributed systems. The DHT is a key–value store, where a key is a file identifier and a value is a peer identifier (ID) that holds the file. As the entries of the DHT are distributed among the participating peers, a request to locate a file will be routed among the peers. More precisely, a file consists of one or more chunks in IPFS, and each chunk corresponds to the entry in the DHT: the key field is the chunk ID (i.e., a cryptographic hash of the chunk itself), and the locator field contains the peer ID who holds the chunk. Actually, the DHT has another kind of a record, which maps a peer ID to its IP address.

To allow other peers to locate a chunk, a peer first has to advertise which chunks she has. Thus, the peer has to inform its neighbors of the corresponding DHT entries, each of which consists of a chunk ID as a key and its own peer ID as a value. Then, these neighbors will store the information in their own DHTs and forward the information to their own neighbors, thus spreading the chunk information throughout the P2P network.

When a peer first sends the query to its neighbors using the chunk ID as the key, its neighbors respond that (i) if the neighbor has the corresponding entry, it will return the value for the given key, (ii) if the neighbor does not have the corresponding entry, then it will return the information of its neighbors whose IDs are closer to the given key. In this way, the peer can reach the destination (peer) that holds the entry of the given key by iterating the above process since every peer is responsible for managing the locations of the chunks whose IDs are close to its own peer ID. Note that the above routing process delivers not the chunk itself but the ID of the peer holding the chunk. With the delivered peer ID, the peer can query the DHT again to find out its locator.

After finding out the peers who hold the chunks of interest, data exchanges occur between a requesting peer and the chunk holding peers. IPFS proposes the data exchange protocol, BitSwap [24], to successfully deliver the chunks between peers and further incentivize the participation of peers. Upon the peer connection, peers exchange the want-list of chunk IDs that they want. If a peer has a chunk in the want-list, it will notify the other peer what chunk she has. Then, the chunk transfer occurs. To promote the data transfer between the peers, BitSwap takes the tit-for-tat strategy; each peer keeps the ledger that records the numbers of exchanged bytes with others, called the debt ratio. That is, when sending chunks, a peer may prioritize a counterpart whose debt ratio is high (who sent more data than received), and deprioritize one with a low debt ratio (who received more than sent) in order to penalize free-loaders.

In addition, IPFS provides a method to represent the relations among the chunks of a file: Merkle DAG, which is a hierarchical tree structure where each leaf node is a chunk. In Fig. 1, the Merkle DAG is the root hash of the tree that consists of the three chunks. In IPFS, a chunk ID is the cryptographic hash of the data of the chunk. Thus the file ID is the Merkle DAG of the file, and is used to request the file. Using these primitives, IPFS provides useful properties: (i) every chunk is uniquely identified, (ii) a tampered chunk can be detected, and (iii) the same chunks can be deduplicated.

Let us analyze IPFS in terms of data consolidation, ownership, and privacy. IPFS solves the data consolidation problem by distributing data across peers. However, compared with other schemes like Storj, IPFS has the following shortcomings. As any user can request and replicate any data chunk in her repository, IPFS cannot guarantee the ownership of the data. There is no encryption and access control, which means user privacy is not preserved systematically.

2.2. *Storj*

Storj [25], developed by Storj Labs, is a notable project for a decentralized cloud storage system that aims to provide robust, reliable, and scalable operations. However, Storj is slightly different from IPFS in setting its goals. Storj focuses on security and performance, such as latency, and introduces specialized nodes called the Satellite.

Whereas all participating peers are treated equally in IPFS, three kinds of nodes exist in a Storj system: (1) **A client** handles user requests (Upload/Download). (2) **A storage node** provides its residual disk space and responds to user requests. (3) **The Satellite**[2] mediates the clients and storage nodes by discovering storage nodes that hold the requested file, managing the reputation of storage nodes, performing data auditing, and so on. As similar to IPFS, Storj builds a global-scale P2P network of nodes using a Kademlia DHT. Since the DHT usually requires multiple round trips in querying the DHT, Storj uses a caching mechanism to reduce the latency for the (storage) node lookup process: the Satellite caches the pair of a (storage) node id and its locator. Upon a routing request, the Satellite will propagate the query through the DHT in case of cache misses. In this way, Storj can reduce the overall latency in finding nodes.

Regarding the data availability, while IPFS relies on a simple replication strategy, Storj relies on more sophisticated techniques, namely the Reed–Solomon [26] erasure coding and a reputation mechanism. Erasure coding [27] is a coding technique used for resilience to failures: when a file is erasure-coded, it outputs n chunks, where only k out of n chunks are required to restore the original file. In Storj, each of the n chunks may be hosted in different storage nodes. Thus, the original file may be recovered in the presence of node failures since it does not require all of the n chunks. Also, it can enhance the overall latency since the Satellite can choose better storage nodes (e.g., lower round trip times). Note that when splitting a file into chunks for the erasure coding, the chunks are encrypted as well.

Uploading/downloading operations of Storj are illustrated in Fig. 2. The client first requests uploading/downloading a file to the Satellite. In the case of uploading, the Satellite finds and returns the set of storage nodes based on the requirements of her requests, such as the residual disk space and latency. Similarly, it returns the set of nodes holding the requested chunks to the client in the case of data retrieval. The Satellite chooses not a single but a number of storage nodes so that the chunks of the same file can be stored across different nodes.

While IPFS relies on BitSwap to incentivize nodes, Storj has a network-wide reputation system since the Satellite keeps track of the reputation of each storage node. A storage node's reputation will be degraded if it behaves badly: failing data audit, [3] failing to return the requested data, or failing uptime checks. If a node continuously fails

[2] A group of multiple control nodes is collectively called *the Satellite*.

[3] In Storj, the Satellite periodically audits storage nodes to check whether they keep the chunks in its storage.

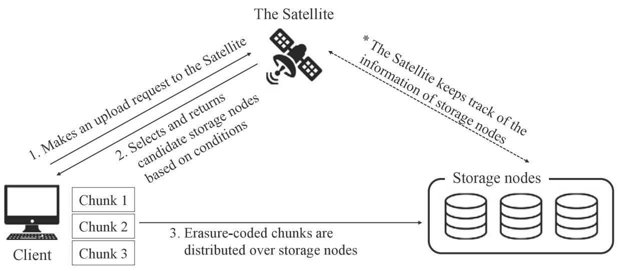

The Satellite
1. Makes an upload request to the Satellite
2. Selects and returns candidate storage nodes based on conditions
* The Satellite keeps track of the information of storage nodes
Storage nodes
Chunk 1
Chunk 2
Chunk 3
Client
3. Erasure-coded chunks are distributed over storage nodes

Fig. 2. Data uploading operations in Storj are illustrated. Thanks to the Satellite and the erasure coding, Storj can enhance its delay performance and availability compared to IPFS.

to comply, it will no longer be selected to host the data chunks, and stored chunks will be moved to other storage nodes, thus providing fault-tolerance and increasing data availability.

In summary, Storj solves data consolidation like IPFS. Storj achieves better performance than IPFS by introducing the Satellite; however, it compromises the decentralization more or less. Also, Storj guarantees the data ownership through access control by encryption. On the other hand, Storj does not fully support user privacy since the Satellite can monitor user requests, while user-generated data can be protected through encryption.

2.3. Solid

Solid [28] separates data from applications on the web, aiming to empower users to have data ownership. One of the rising concerns of current online social networking (OSN) services is data consolidation. Every data created by a user is stored in the service providers' private storage, leading to the service provider lock-in [29]. To mitigate such data consolidation, Solid logically separates user data from service providers by introducing the concept of a Personal Online Datastore (POD). Thus, in Solid, only users own their data, while OSN service providers cannot store user data. A Solid application receives user data from other users' PODs when user data is needed.

The Solid ecosystem consists of three kinds of entities: users, PODs, and Solid applications. A user is a client who owns her data and uses OSN services. A POD is a logical storage for each user, in which each user's data is stored. A Solid application is an OSN application that runs on the application user's POD that reads or writes data from the PODs of data owners.

More specifically, a user's POD stores every object belonging to the user. Note that a POD is an abstraction of logical storage; any physical method of storing data (e.g., cloud storage, decentralized storage, users device storage, etc.) can be used. In the existing OSN service structure, user data is stored in the service provider's storage. Thus, when a user requests data from another user, the application authorizes the user and returns the requested data if authorized. However, in Solid, since user data belongs to each user's POD, the operation of fetching the user data differs significantly. When a user requests data of another user (or her POD) through the Solid application, the application can access the requested POD only if the requested POD authenticates the requester and authorizes the request. Thus, Solid employs an access control method for PODs.

Solid authenticates a user's identity using a WebID [30], a globally unique decentralized identifier in the form of a URI. A WebID indicates the location of a public WebID profile that contains the user's information, e.g., her public key, name, and nickname. Solid uses the WebID-TLS protocol as its primary authentication mechanism by which a user presents her signature to another user's POD through a TLS connection. In this way, the POD verifies a requester's identity by matching the signature with the public key stored in her WebID profile.

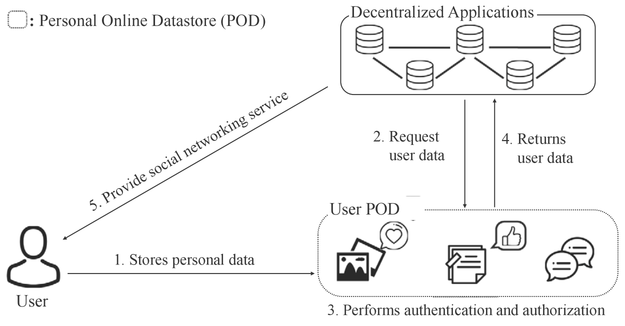

: Personal Online Datastore (POD)
Decentralized Applications
2. Request user data
4. Returns user data
5. Provide social networking service
User POD
User
1. Stores personal data
3. Performs authentication and authorization

Fig. 3. Solid operations are illustrated with a single POD. A user stores her personal data in her POD, which can be accessed by decentralized social applications depending on the corresponding ACL.

A user's POD controls the accesses for her objects in her POD based on the authorization policy specified by herself. More specifically, users can control accesses to objects in their PODs based on the Web Access Control (WAC) mechanism. WAC is a decentralized access control mechanism allowing users to access the objects with different scopes based on the access control lists (ACLs). The ACL is a set of authorization statements describing who can access what. Note that the ACL is also an object in a POD.

Fig. 3 illustrates the Solid operations with a single POD. Decentralized social applications on the Solid platform offer services by accessing users' data stored in PODs only with their owners' grants. To bridge distributed pieces of data, Solid represents data as Linked Data [31]. In Linked Data, every data object has its identifier, e.g., URL, and the object can be connected to other objects by referring to their identifiers. For example, assume that *Alice'* photo identified by https://alicepod.solid/photo/summer is stored in her POD and *Bob's* comment, with the identifier https://bobpod.solid/comments/12345, to *Alice'* photo is stored in *Bob's* POD. The relation between the photo and the comment can be represented by setting the URL of *Bob's* comment as one of the attributes of her photo.

To sum up, Solid addresses data consolidation by introducing PODs. Solid also solves the data ownership problem based on the WAC, which allows the data owners to control who can access what. User privacy is not fully supported. While the privacy of User-Generated Content (UGC) is preserved by giving full control of data to its owner, users' online behaviors can be revealed since the users send their requests to Solid applications.

2.4. Mastodon

Mastodon [32–34] is an open source project for decentralized OSN services with microblogging features. Over the last decade, popular OSN services began to provide advertising services based on user data to meet their business model. This caused the customer's trust problem with the OSN services. The main focus of Mastodon is to provide a user with the control of her content free from sponsored content or manipulation of content locations by service administrators [33].

To provide decentralized OSN services, Mastodon constructs two layers: (i) client-to-server and (ii) server-to-server. For communications within each layer, the ActivityPub protocol [35] is used, which is an open protocol for decentralized OSN services. First, a user can create and run her own Mastodon server with her own set of rules. Here, the server created is called an instance, and the user who created the instance is referred to as an instance manager. In order to receive the Mastodon service, users register their accounts with the instance. Users who register themselves with the instance are referred to as clients. In an instance, multiple clients register their accounts, and they can follow one another to see other's messages. In this way, client registration,

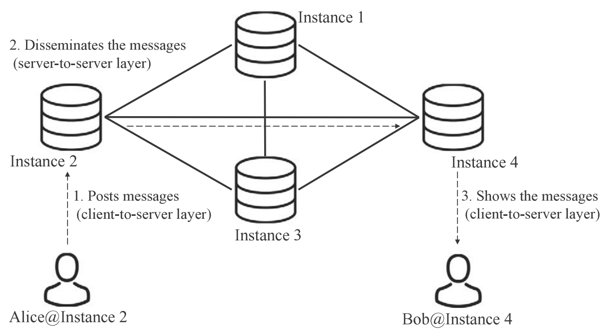

Instance 1
2. Disseminates the messages (server-to-server layer)
Instance 2
Instance 4
1. Posts messages (client-to-server layer)
Instance 3
3. Shows the messages (client-to-server layer)
Alice@Instance 2
Bob@Instance 4

Fig. 4. The operation of Mastodon is illustrated. A user registers her account with the Mastodon instance, where the combination of the username and the instance is globally unique. Therefore, each user is identified by her name, by which message exchanges between users occur.

follow-up between clients, and exchange of messages within a single instance occur within (i) the client-to-server layer by using ActivityPub's client-to-server protocol. With the above operations, clients can exchange messages from one another only within the same instance. For global federated interworking, communications between instances are required. Mastodon guarantees the global reach by relaying messages between clients registered in different instances at (ii) the server-to-server layer by using ActivityPub's server-to-server protocol. Due to the above architecture and its operations, user data in Mastodon is not consolidated on a single server operated by a single administrator. User data is decentralized to multiple instances that anyone can operate.

More specific operations are illustrated in Fig. 4. An instance has a globally unique name, and a client has a locally unique name within the instance. Thus, by combining the username and the instance name (e.g., username@instance-name), the client has a globally unique name. Each client has an inbox for incoming messages from other clients and an outbox that posts its outgoing messages. Note that a client, her inbox and her outbox each have URL addresses (e.g., https://social.example/alice/outbox is for *Alice'* outbox). These addresses comply with ActivityPub, which defines the format of serialized linked data based on JSON [36,37]. In Fig. 4, there are *Alice@instance 2* and *Bob@instance 4*, who wish to send and receive messages. *Alice@instance 2* posts her message (to Bob) through her outbox. This message is called a "note". The note contains the inbox URL of the receiving client. In this example, the *Bob's* inbox URL is written in the note. The note in the *Alice'* outbox is sent to *instance 2* with which *Alice* has registered, and this operation occurs at the client-to-server layer. Next, *instance 2* receives the note from *Alice*, and then assigns the note an id in the form of a URL (e.g.,https://social.example/alice/posts/*{post* id}), then checks the inbox URL of the client who will receive the note. In this example, the note contains the inbox URL of *Bob*, and thus it is delivered to *instance 4* through the server-to-server layer. Finally, *instance 4* delivers it to *Bob's* inbox through the client-to-server layer.

In summary, in Mastodon, anyone can run an instance, and instances connect to one another through the ActivityPub protocol to provide a global OSN service. Hence, user data is decentralized to instance managers running Mastodon instances. Note that user data is exchanged and managed by instance managers, and thus users do not have complete data ownership. User privacy is not specifically considered in Mastodon.

2.5. Dat

Dat [38] is a file sharing protocol among peers over a distributed P2P network. Existing solutions for sharing data among multiple users have some constraints. For example, centralized cloud storage services such as Dropbox and Google Drive require users to store their data in the service providers' infrastructures, which leads to the issues of vendor lock-in and user privacy leakage. In addition, delivering data over an HTTP or FTP connection is not suited for collaborative applications since those protocols lack the built-in support for version control and synchronization. Therefore, the Dat protocol is designed to support sharing files with synchronization and version control for distributed and collaborative applications.

The main design goals of Dat are collaborative synchronization and version control. Thus, DAT seeks to achieve (i) content integrity, and (ii) decentralized mirroring. Content integrity means being able to verify that an object has not been tampered nor modified from the original one. When multiple users work on the same object, content integrity must be ensured because it can be forged or unintentionally altered. Decentralized mirroring means that users who want to share the same object can automatically discover each other and exchange the object in a swarm. Decentralized mirroring is necessary since users who work on the object must view the same version simultaneously.

Every object in Dat has its own URL that consists of a protocol identifier, a public key, and an optional suffix. The protocol identifier of Dat is determined as *dat://*, which makes Dat URLs easily recognizable. A public key is unique to the corresponding object and generated by its publisher. A user discovers other peers who have the object and verifies its integrity by using the publisher's public key. The optional suffix is used to identify additional data within the URL (e.g., file path).

To retrieve an object, a user submits its URL that includes its public key; Dat calculates a discovery key by applying the BLAKE2b hashing function [39] to its public key. Then, the discovery key is broadcast via a network to find any peers who hold the corresponding object or are interested in the object. Peers having the data respond to the user with their IP addresses and port numbers. After finding peers, the user selects a sender among peers and establishes a TCP connection. The sender first sends a feed message, which contains the discovery key and a random value of the nonce. The nonce is used to encrypt subsequent messages with the XSalsa20 stream cipher [40]. XSalsa20 takes the public key in the URL and the nonce to generate a keystream, which is XORed with plaintext data. After the initial feed message, the sender and the user exchange handshake messages to determine exchange rules (e.g., whether to send ACK for each chunk). Then, they exchange chunks by exchanging *want* and *have* messages.

Dat provides content integrity by using cryptographic hashes of objects. An object in Dat consists of multiple variable-sized chunks, and each chunk of the object has a corresponding hash. There is a parent hash that verify the integrity of two child chunks, which iteratively forms a tree structure. The data publisher signs the root hash with the secret key corresponding to the public key in the URL. Therefore, anyone who knows the public key can verify the integrity of the object.

Dat also achieves decentralized mirroring by broadcasting a discovery key. When a user broadcasts a discovery key of the object she wants, other peers in the network can report that they are also interested in the same object. Thus, a peer holding the object sends it to not only the user who broadcasts the discovery key, but also other peers who express interests in it.

As an example, Fig. 5 shows how peers share an object of interest using Dat. First, *Peer 1* requests an object by its URL that contains its public key. Dat calculates the discovery key by hashing the public key and broadcasts it via the network. If there are other peers interested in the same object, they also report/broadcast to the network with their IP addresses. Then, *Peer 4*, who has the object, responds to *Peer 1* and the other peers who reported their interest. Finally, peers in the network share the object after exchanging data.

From the perspective of the decentralized Internet, Dat can mitigate the data consolidation problem since it distributes user data among peers in the network. However, the data ownership cannot be supported since every peer can access an object without the permission of its owner. Also, user privacy is not systematically protected since an object is not encrypted. As a user identity is anonymized by using only her public key, anonymization is supported in Dat.

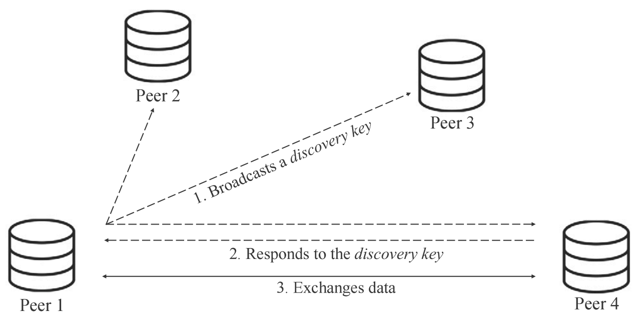

Peer 1
Peer 2
Peer 3
Peer 4
1. Broadcasts a discovery key
2. Responds to the discovery key
3. Exchanges data

Fig. 5. A flow of the Dat protocol is illustrated. A user discovers peers that hold the object of interest using a discovery key. Then, it exchanges data after negotiating with the peers.

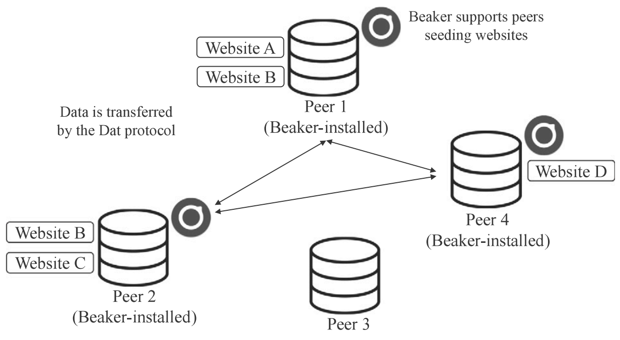

Beaker supports peers seeding websites
Website A
Website B
Peer 1 (Beaker-installed)
Data is transferred by the Dat protocol
Website D
Peer 4 (Beaker-installed)
Website B
Website C
Peer 2 (Beaker-installed)
Peer 3

Fig. 6. The operations of Beaker are illustrated. The Beaker browsers running on peers treat a website as a group of files. Thus, websites can be seeded by Beaker-installed peers. The Beaker browser leverages Dat as its underlying exchange protocol.

2.6. Beaker

Beaker [41] is an open source web browser for the decentralized Internet, developed by Blue Link Labs. They develop a decentralized peer-to-peer web browser, leveraging Dat [38] as a base protocol. Beaker allows a user to self-publish a website without setting up any web server and uses Dat to replicate an entire website.[4] That is, an original server publishes a website, and then any peer can seed the website (like mirroring) if she wishes to do so. Thus, a website can be accessed as long as at least one peer can provide its files even if the original server is not available.

Fig. 6 illustrates how Beaker works. First, peers build a distributed P2P network by installing Dat. As a website is a set of files in Beaker, peers can seed websites if they wish to do so. In Fig. 6, *Peers 1, Peer 2*, and *Peer 4* use Beaker (Notice the Beaker icon). *Peer 1* seeds websites A and B, *Peer 2* seeds websites B and C, and *Peer 4* seeds website D. Thus, if *Peer 1* or *Peer 2* tries to access website D, she can browse website D by getting the corresponding files from *Peer 4* using Dat.

In summary, the Beaker browser operates on top of Dat and basically has the same drawbacks as Dat. Websites (or their files) can be distributed across multiple peers, which helps to mitigate the concentration issue. However, it cannot address the ownership and privacy issues.

2.7. YaCy

The current search engine market is monopolized by Google, which leads to the concentration of user data (e.g., search keywords). There is also a possibility that companies monopolizing the search engine market might manipulate the search results (e.g., promoting sponsor sites). A decentralized search engine, YaCy [42,43], is developed as a countermeasure of centralized search engines. To prevent search engines from dominating the indexes of web pages and providing users with biased results (e.g., advertised or censored links), YaCy relies on decentralized crawling and indexing.

To help you understand Yacy, let us give the background of search engines. A search engine has three main functions; (i) crawling, (ii) indexing, and (iii) ranking [44]. Search engines crawl web pages through a crawler. These crawled web pages are pre-processed (e.g., natural language processing) and indexed for easy retrieval (e.g., inverted indexes). When a user's search query comes in, the query and the relevant indexes are put into a ranking algorithm to score the web pages. The user receives a list of web pages ranked by this score.

In the existing search engine, a single service operator performs crawling and indexing. Therefore, web pages and index information are consolidated to a specific service operator, and user data (e.g., search terms) are also concentrated. In addition, ranking results are also provided through the consolidated indexes; thus, search service operators might inexplicably or intentionally manipulate them.

To solve the consolidation problem in search engines, YaCy performs decentralized crawling and indexing. Fig. 7 illustrates the operations of YaCy. First, YaCy configures a P2P network of YaCy peers, who collectively maintain a search engine in a decentralized fashion. Each peer crawls web pages independently (step (1) in Fig. 7), and then preprocesses and indexes the crawled web pages. Thus, there is no single search engine service operator. Crawled web pages and their indexes are stored and managed by each peer. To globally share the information of crawling and indexing of each peer, each YaCy peer shares its index information to the YaCy peer network through a DHT. For example, if an inverted index is constructed based on a word, each peer indexes and shares (words, web pages) pairs, and the globally shared DHT is populated using each word as a key. Thus, in the DHT, words serve as the keywords to the relevant web pages (i.e., their URLs) as shown in step (2) in Fig. 7. After configuring the DHT for words, the YaCy P2P network can answer search queries.

When a user sends a query as shown in step (3) in Fig. 7, The query is sent to the P2P network through the YaCy search engine. Then the P2P network finds the peers' IP addresses holding the relevant URLs from the DHT (step (4)). Finally, the user connects to the peers to retrieve the URLs (step (5)).

In summary, the indexes and web pages are decentralized among YaCy peers, which addresses the problem of data consolidation. YaCy can also protect user privacy by preventing users' search histories from being exposed since search keywords are forwarded among peers in the form of hash values. However, keywords may be inferred from the URLs in the reply. Also, YaCy does not support the data ownership.

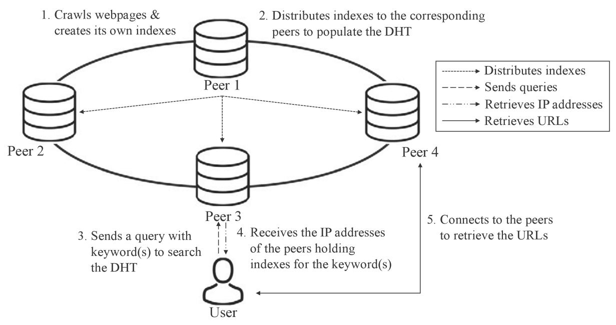

1. Crawls webpages & creates its own indexes
2. Distributes indexes to the corresponding peers to populate the DHT
Peer 1
Peer 2
Peer 3
Peer 4
Distributes indexes
Sends queries
Retrieves IP addresses
Retrieves URLs
3. Sends a query with keyword(s) to search the DHT
4. Receives the IP addresses of the peers holding indexes for the keyword(s)
5. Connects to the peers to retrieve the URLs
User

Fig. 7. The operations of YaCy are illustrated. Each peer crawls websites and makes its own indexes, which populate the DHT for keywords. When a user sends a query, YaCy looks up its DHT to find out the IP addresses of the peers (that hold the relevant URLs in their indexes). Finally, the user connects to the peers to retrieve the URLs.

[4] Note that a website is deemed a set of individual files here.

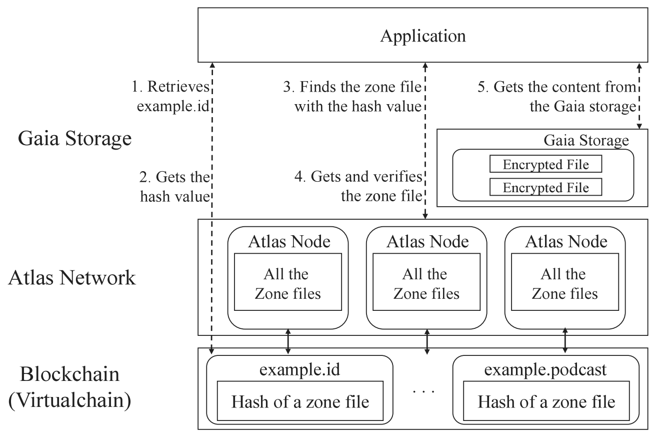
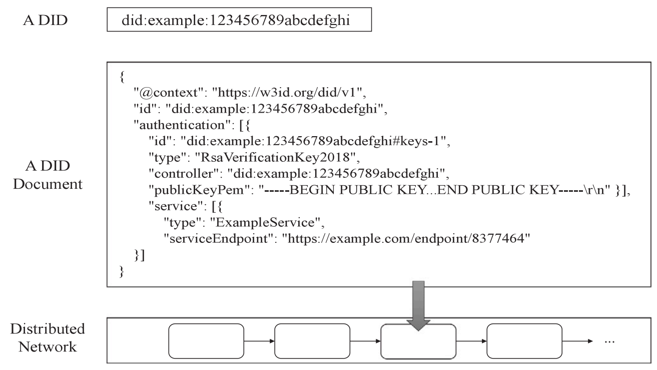


Fig. 8. The operations of Blockstack are illustrated. The blockchain maintains the hash of the zone file of every user (under her name), and the Atlas network contains the zone file, which includes the location of the user's data locker in Gaia.



```
did:example:123456789abcdefghi
```

```
{
  "@context": "https://w3id.org/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{
    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n" }],
    "service": [{
      "type": "ExampleService",
      "serviceEndpoint": "https://example.com/endpoint/8377464"
  }]
}
```

Fig. 9. The examples of a DID and its DID document are shown. The DID consists of a URL scheme identifier, the name of the DID method, and the method-specific identifier. The DID is used to fetch its DID document that has information about the DID subject, such as its public key and service point.

2.8. Blockstack

Blockstack [45,46] aims to develop a decentralized computing architecture that provides a full-stack alternative to traditional cloud computing. Blockstack redesigns the application layer of the current Internet and provides a new infrastructure for decentralized applications; applications built on Blockstack enable users to own and control their data. The Blockstack architecture provides a framework of decentralized computing without any centralized entity, allowing users to have sovereignty on their names and storage spaces.

To provide the full stack for decentralized computing, Blockstack consists of four parts – (i) Stacks Blockchain, (ii) Gaia, (iii) Authentication/authorization, and (iv) Libraries & SDKs. When a Blockstack application tries to retrieve an object by using its name and the username of its owner, the application first accesses the Stacks Blockchain. The Stacks Blockchain enables a user to register and control her username linked with pointers to her private storage, which is called data locker(s). More specifically, the Stacks Blockchain has two sublayers: a blockchain [47] and the Atlas network. The underlying blockchain stores the pairing information between the username and the pointer (hash) of her zone file, which includes the locations of her data lockers. The zone file of every user is stored in the Atlas network, and the Atlas network returns the corresponding zone file when the hash of a zone file comes in as a query. Note that to find a location of a data locker, Blockstack uses only the corresponding username. The name of an object (in the data locker) is used when reading/writing the object in the data locker in Gaia. Gaia is a storage system that allows a user to deploy her data locker[5] and control her data. The user can select any storage system (e.g., a cloud provider, local storage, or remote storage) for her data locker. User data is encrypted and signed with user-controlled keys; thus, although a cloud provider manages the data locker, it cannot read the data. Thus, authentication/authorization is required to read/write data to the user's data locker. The Authentication protocol in Blockstack is based on public key cryptography. A user authorizes an application by logging into the application with her username to give the application the credential to read/write her data (with decryption/encryption, respectively) under her username. For this, the user uses her private master key to generate application-specific keys, which are sent to allow the application to encrypt and sign her data. Also, the user's credential can be used to assure other users that the application requests their data on behalf of the user. Blockstack also provides Libraries and SDKs to help to develop the Blockstack applications.

The operations of a user's application that wishes to retrieve an object of another user (say, her id is *example.id*) are illustrated in Fig. 8. Initially, the application retrieves the hash of the zone file from the blockchain with the object owner's name (*example.id*). The application then requests the zone file to the Atlas network with the hash of the zone file. On receiving the zone file, the application verifies its integrity by comparing the hash of the received zone file with the one from the blockchain. The application then finds the Gaia URL (of the data locker) from the zone file and finally retrieves the object from the data locker in Gaia.

In summary, Blockstack addresses the consolidation problem by storing user data in a user-controlled data locker, not in the storage of a service provider. Blockstack also solves the data ownership problem since applications cannot access user data without its owner's permission. However, user privacy is not perfect. While the user data is privacy-preserved since the Gaia storage system does not allow unauthorized access, users' online behaviors, such as requests to obtain zone files, can be observed by the peers in the Atlas network.

2.9. Decentralized identifiers (DIDs)

Decentralized Identifiers (DIDs) [14] defines a new type of identifier to provide verifiable, decentralized digital identities. It is being standardized by W3C and Decentralized Identity Foundation (DIF) with goals such as security and discoverability. DIDs is motivated to address the drawback of the centralized authority like DNS, i.e., a single point of dependency/failure; instead, DIDs relies on distributed ledger technologies like a blockchain for authenticating identifiers. DIDs provides a specification by which every entity can generate her own identifier or verify other entities' identifiers in a decentralized fashion. Note that "DIDs" refers to either a specification or a plural of a DID, which is an instance of an identifier.

As shown in Fig. 9, a DID consists of the URL scheme identifier (*did*), the method (e.g., *example*), and the method specific identifier (e.g., *123456789abcdefghi*). Thus, a DID indicates the corresponding DID document that describes a DID subject. Examples of the attributes in the DID document are the subject's public key and service point. Note that the DID is also contained in the DID document as one of the attributes. Based on the relations among a DID, its DID document, and its DID subject, any system that wants to use DIDs should specify how to create, read, update, and deactivate DIDs in the system, which is called a DID method.

With DIDs, any entity can be authenticated by using cryptographic proofs, which is illustrated in Fig. 10. For instance, a DID subject (say,

[5] A data locker is similar to a POD in Solid.

Table 1
Summary of the prior solutions.

Name	Objective	Key features	Problem addressed		
			Data consolidation	Data ownership	User privacy
IPFS [13]	Distributes files over multiple nodes	– DHT: Node info. (Node ID, address) and the chunk info. (CID, chunk location) are stored – Linked Data: Data chunks are linked based on Merkle DAG – Bitswap: Measures the trustworthiness of peers	O	X	X
Storj [25]	Distributes files over multiple storage nodes securely	– The Satellite: Provides functions of locating, access control, and so on – Erasure coding: Increases availability – Reputation system: Evaluates the trustworthiness of nodes	O	O	△[a]
Solid [28]	Separates user data from applications and gives the full control of data to its owner	– POD: A logically personalized and distributed storage which is fully controlled by the owner – Linked Data: Every resource in Solid is represented by Linked Data principles	O	O	△[b]
Mastodon [32]	Provides users with control of the content distribution channels free from sponsored contents	– Server-to-server layer: Global federation between (server) instances by relaying messages between clients registered with different instances – Client-to-server layer: Client registration, relationships, and exchange of messages within a single instance	O	X[c]	X
Dat [38]	Shares files on a P2P network with functions of synchronization and versioning	– Content integrity: Any entity can perform the verification of a content object by the Merkle tree – Decentralized mirroring: Peers can discover each other and exchange data in a swarm	O	X	△[d]
Beaker [41]	Shows users the decentralized Internet with supporting decentralized functions	– Dat support: Beaker can use Dat, a decentralized file sharing protocol – Seeding web pages: A user can seed web pages as a peer	O	X	X
YaCy [42]	Distributes crawled web pages (and their indexes) over multiple peers	– Decentralized crawling: Each peer crawls web pages – Decentralized indexing: Each peer creates indexes for its own crawled web pages	O	X	O
Blockstack [45]	Provides a full-stack framework for decentralized computing	– Stacks blockchain: Stores usernames which is linked with pointers to their data lockers – Gaia: A storage system that allows individual users to make their private data lockers and control their own data – Authentication/authorization: allows to read/write data from data lockers for authenticated users only	O	O	△[e]
DIDs [14]	Supports individuals to create their own identifiers without central authorities	– DID document: Describes the attributes of a DID – DID method: System-specific way of creation, retrieval, update and deletion of DIDs	O[f]	O[g]	X[h]
Sovrin [48]	Provides an ID system based on DIDs	– Attributes: Information about the subject – Claims: A digitally signed assertion about particular attributes – Disclosure: A tailored claim that does not disclose unnecessary information	O	O	O[i]

[a]The Satellite can observe user's data upload/download requests; thus Storj cannot fully guarantee user privacy.
[b]Solid assures that UGCs are protected by any unauthorized access; however, user's online behaviors are revealed since users send their requests directly to social applications.
[c]User machines may store user data; however, the main storage of user data is the (server) instance with which she is registered.
[d]Simple anonymization is supported in Dat since a user uses the hash value of her public key as her pseudonym.
[e]Users' requests to the blockchain can be observed by the peers of the Atlas network.
[f]In case of DIDs, we considered IDs as a kind of data. DID documents are stored in distributed networks; thus, they are not consolidated.
[g]A user who wants to create her own DID can generate her DID document on her own.
[h]A DID just represents the identity of the user, and user privacy depends on an external solution that utilizes DIDs.
[i]Sovrin provides a way to hide particular attributes in a claim, and private attributes are not recorded in the public ledger.

a student) that wants to prove her identifier sends her DID with a proof of her identifier (say, a signature on a challenge message) to a relying party (say, a university authority). On receipt of the identifier proof, the relying party asks a DID resolver [49] to fetch a DID document from the blockchain. Then, the relying party verifies the proof by using the public key in the DID document, which finishes the authentication process.

Note that DIDs addresses the issues related to IDs rather than user data. DIDs addresses the "ID consolidation" issue since there is no central entity in a DID system, and the DID documents are stored in distributed networks such as the blockchain. DIDs also solves the ID ownership problem; anyone can generate and register her DID document. However, whether DIDs can preserve ID privacy or not depends on the method in DIDs. If the method allows an ID to be generated in an anonymized fashion (e.g., public keys), there is no ID privacy issue.

2.10. Sovrin

Sovrin [48] is a self-sovereign identity system with its catchphrase — identity for all. The Sovrin system is primarily motivated by the two following problems: (1) the lack of a standardized format to describe identities and (2) the standardized way to verify the identities.

To address the above issues, the Sovrin system leverages (1) the DIDs to describe identities and (2) a public blockchain to authenticate identities. For each DID, there is a pair of a private key and a public key. Thus, an entity can generate a verifiable claim by generating her digital signature, and anyone can verify the claim with her public key in the corresponding DID document.

In Sovrin, there are several properties for privacy by design. First, Sovrin aims to mitigate correlation attacks for a particular entity. To

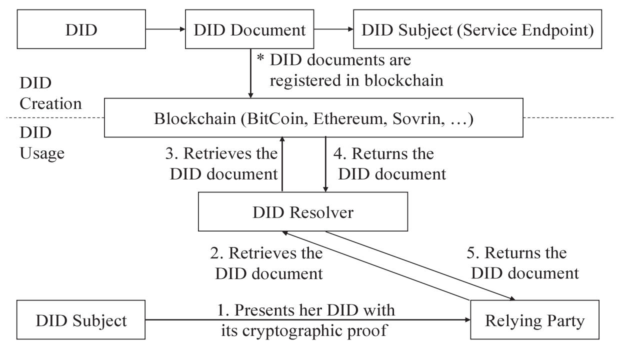

DID
DID Document
DID Subject (Service Endpoint)
* DID documents are registered in blockchain
DID Creation
DID Usage
Blockchain (BitCoin, Ethereum, Sovrin, …)
3. Retrieves the DID document
4. Returns the DID document
DID Resolver
2. Retrieves the DID document
5. Returns the DID document
DID Subject
1. Presents her DID with its cryptographic proof
Relying Party

Fig. 10. The operations of how a subject authenticates herself in DIDs are illustrated. First, the subject who wants to prove her identifier sends her DID with a cryptographic proof. Then, the relying party that receives the DID sends a query to the DID resolver and verifies the proof with the DID document in the response.

this end, an entity generates different public keys for different entities (say communication counterparts). For example, *Alice* has one public key with her bank and another with her company. In this way, an adversary cannot infer easily with whom *Alice* has relations. Second, Sovrin provides a method to specify an entity's private attributes in the ledger. This is supported by allowing the entity to store the hash values of her private attributes in the ledger. With this property, she can prove the integrity of her private attributes without revealing them to the public. Third, an entity can control how much data is shared in a particular context, called selective disclosure. Selective disclosure uses the zero-knowledge proof with attribute-based credentials for trust [50]. For instance, a local authority can generate a claim about only the gender of *Alice* with its signature while hiding the other information of *Alice* in the claim.

In Sovrin, the consolidation issue of IDs is addressed since IDs are stored in a blockchain. Sovrin also guarantees the ownership of IDs, attributes, and claims since they can be generated in a decentralized fashion by individual users. Sovrin enhances user privacy by the three mechanisms: using different public keys, storing only the hashes of the private attributes in the ledger, and supporting hidden attributes.

2.11. Blockchain-based approaches

As blockchain technologies emerge as the solution to the problem of Internet consolidation, they have attracted much attention of academia and industry. Blockchain has been widely adopted as the foundation for decentralization techniques, as it offers security, automation, and availability [51].

In [52], the authors proposed a privacy-preserving pricing system for the smart grid. They achieve this by designing a lightweight distributed cloud storage architecture that uses a dual blockchain. This storage system stores encrypted data and employs the dual blockchain, which consists of private and public. This way, they ensure user privacy, reduce costs, and provide security.

[53] proposed Blockchained on-device Federated Learning (FL), which addresses two key limitations of traditional FL: (i) the dependence on a central server and (ii) the lack of incentives for local devices to contribute their computing resources. Instead of the central server, the proposed approach utilizes a blockchain, by which miners verify and distribute the local model updates from participating devices. Furthermore, the devices are incentivized to participate in the FL process through proportional rewards based on the sizes of their training samples.

[54] presents a parking-space recommendation platform that preserves user privacy. It achieves the privacy of users while providing the benefits of other parking-space sharing services by employing

Table 2
Categorization of prior solutions.

Category	Name	Differences & limitation
Distribute storage	IPFS	• Fully distributed with DHT • Long latency due to DHT
	Storj	• Decentralized, but Satellite intermediates • Single point of failure
Social network	Solid	• Full ownership of user data • Limited to social network
	Mastodon	• Block unwanted content • No ownership, no access control
Application	Dat	• File transfer protocol • No ownership
	Beaker	• Distributed browser • No ownership
	Yacy	• Decentralized search engine • Long latency due to DHT
Identifier	DIDs	• ID scheme without central authorities • Limited to ID representation
	Sovrin	• ID system based on DID • Deals with IDs only

anonymous authentication, anonymous payment, and reputation ratings. The authors achieved this goal by utilizing blockchain and existing privacy-enhanced cryptographic tools.

In [55], the authors proposed an advanced zero-knowledge ledger. They replaced their previous range-proof techniques with improved inner product-based zero-knowledge proofs, which are the most efficient range-proof techniques available. This technique makes the zero-knowledge ledger perform faster than the existing ones since multiple range-proofs can be integrated into a single range-proof.

2.12. Summary: Existing solutions

So far, we have introduced representative studies proposed for the decentralized Internet. Since there have been already several surveys on blockchain [56–61], we focus on solutions that are not blockchain-centric in this paper, which are summarized in Table 1.

We also categorized the existing solutions in Table 2 for the purposes of comparison and analysis. While the solutions had been developed independently by different developers, they can be classified by the problem that each technology aims to address. Both IPFS and Storj propose a method of storing files in a distributed fashion, but they differ in their approaches. IPFS uses the DHT for locating and delivering files in networks, while Storj uses the DHT with special nodes called the Satellite. Thus, in case of Storj, the Satellite has the issue of a single point of failure. By contrast, IPFS does not face this issue, but it has the DHT-inherent performance problem [62]. Solid and Mastodon both offer distributed social network services, but they differ in their main purposes. Solid provides a social network platform that allows users to own and manage their data. Solid seeks to fundamentally address data consolidation, ownership, and user privacy, but it provides social networking services only. On the other hand, Mastodon's main purpose is to allow users to block unwanted content like advertisements. Note that there is no access control (and hence no ownership) in Mastodon. Dat is an application protocol for distributed file sharing, and Beaker is a browser that utilizes Dat to provide distributed web browsing. However, both Dat and Beaker do not offer access control or ownership. YaCy is a decentralized search engine that uses a DHT, which may result in longer response times compared to centralized search engines. DIDs and Sovrin primarily focus on the technical aspects of distributed digital identifiers. While the above proposals address some particular aspects of Internet decentralization, Blockstack aims to address a broader range of problems by changing

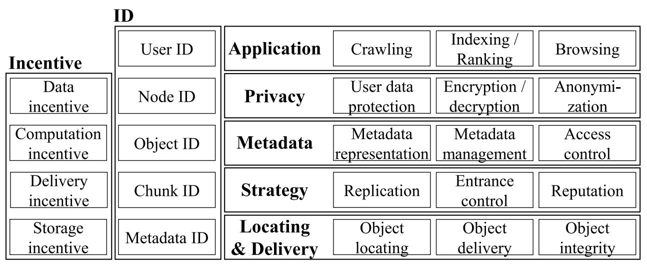


Fig. 11. A reference framework of the decentralized Internet is illustrated. The five modules on the right are related to data plane functions, such as data exchange between nodes. The two modules on the left are responsible for the incentivization and ID management.

the current cloud-based environments to distributed environments. To be brief, Blockstack offers data ownership and solves data consolidation by leveraging blockchain technologies.

3. A reference framework for the decentralized internet

In the previous section, we have reviewed various technical solutions towards the decentralized Internet. However, each solution focuses on different problems, usually from a limited point of view. That is, none of the above solutions consider the decentralized Internet from a holistic viewpoint.

In this paper, we seek to draw a reference framework for the decentralized Internet to address the data consolidation problem. To this end, we first investigate which functions are needed for the decentralized Internet by analyzing the designs of the existing solutions. Then we classify the functions needed for the decentralized Internet into seven modules; "**Locating & Delivery**", "**Strategy**", "**Metadata**", "**Privacy**", "**Application**", "**ID**", and "**Incentivization**" as shown in Fig. 11. The seven modules will be detailed one by one. After that, we identify functions not covered by the prior solutions; we believe they should be added in the reference framework for the full-fledged decentralized Internet. We also discuss the functions that can be technically improved from the prior solutions in Section 4. Finally, in Table 3, we summarize the functions, the research issues, and the existing solutions for each module.

3.1. *Locating & delivery*

We refer to a resource as the storage space (of a participating node). Thus, a decentralized application requires a method to locate such resources. Since participating nodes are not managed in a centralized manner, we need the "Locating & Delivery" module that performs locating resources to store and to retrieve objects (or their chunks). Henceforth, we assume that the unit of storing or transporting an object is a chunk. When a client wishes to upload or download an object, the role of the "Locating & Delivery" module is to perform the storing and retrieval of its chunks. To support the chunk distribution and retrieval, "Locating & Delivery" should support the following functions.

- **Object locating:** In a distributed P2P system, there should be a mechanism to find the node(s) holding a requested chunk; furthermore, the system needs to support the method to find the node's location based on the node's identifier.
- **Object delivery:** When a client has retrieved a set of nodes for the chunk distribution or retrieval request, connections between the client and the nodes (selected by the system or client) need to be established, followed by data transmissions.
- **Object integrity:** In a current centralized web, there are mechanisms to authenticate servers such as HTTPS, and the authenticated server is believed not to act maliciously. However, in distributed systems, any node can join the network; thus, there is no guarantee that chunks from nodes are correct or not modified. A system hence requires mechanisms to guarantee the object/chunk integrity.

The functional requirements of "Locating & Delivery" are satisfied by many systems deployed in the field, such as IPFS and Storj. For the object delivery, most of the existing solutions rely on TCP/IP. In the case of the data integrity, IPFS utilizes the Merkle DAG, as described in Section 2.1.

3.2. *Strategy*

Nodes participating in the P2P network may join or leave at will [63], which is called "churning". Thus, we cannot assume that individual nodes are always up and running. In addition, some nodes may be malfunctioning or malicious. Thus, it is crucial that a client can retrieve an object of interest even if a certain number/portion of nodes fail. The "Strategy" module is responsible for resilient operations.

To improve the availability in "Strategy", we identify the following functions: object/chunk replication, entrance control, and reputation management.

- **Replication:** The first approach is to make multiple copies (of a chunk/object), which are distributed across different nodes. Even though some of the copies are unavailable, the others can be accessed.
- **Entrance control:** Another direction is to require a node to make some commitment before joining the system. In this way, it is not easy for impulsive and unreliable nodes to join the system, or for attackers to launch Sybil attacks [64]. However, if a node that passes the entrance control fails or misbehaves, the following function will handle such nodes.
- **Reputation management:** It is important to run a reputation mechanism [65,66] that continuously evaluates the reliability of nodes in the P2P network. The possible factors in evaluating a node's reputation are its uptime, the success probability of chunk delivery, delivery throughput, protocol conformance, and so on. When the client selects a node to store/retrieve a chunk, she (or the system) will assess the reputation levels of the node.

One of the popular replication approaches is the erasure coding [67], which Storj adopts. For the entrance control, in Storj, a new node is required to find an input for a particular hash output. In SAFE Network [68], a newcomer can carry out only simple jobs until being upgraded to do advanced jobs. The rationale behind such entry barriers is that a node that has spent its computing powers or similar resources is less likely to deviate from the normal system operations. For reputation management, in IPFS, every node's reputation is evaluated in a distributed manner based on the tit-for-tat strategy; each node records the numbers of exchanged bytes with other nodes, which is called the debt ratio. Nodes that have sent more data than have received have a higher debt ratio, and a client preferentially selects such nodes when she receives a data chunk. On the other hand, nodes that receive more data have a lower debt ratio, which will get fewer incentives from the system (The incentive issues will be elaborated later on).

3.3. *Metadata*

The role of the "Metadata" module is to represent and manage the information about objects, i.e., metadata. The metadata of an object may include the information to verify the integrity of the object (e.g., the hash value of the object), which chunks constitute the object (e.g., chunk sizes, indexes), other attributes of the object (e.g., the owner, size, access control), and the relations with other objects.

To this end, "Metadata" should provide the following functions.

- **Metadata representation:** Metadata should be represented in a compatible and extensible fashion. Representation of metadata should be devised for reusability of metadata and agnostic to decentralized applications or development tools. Extensibility is essential to allow for developers to make new types of metadata for their application needs.
- **Metadata management:** The metadata of an object is also an object. As metadata has different characteristics from user data, there can be other design issues when managing metadata — whether to segment metadata into chunks like objects, how to ensure the integrity of metadata, and so on.

In most of the proposals we surveyed, the representation of metadata or the attributes of an object is specialized to their systems [13,25,38]. They typically use a fixed structure of metadata or a predefined set of attributes. For example, Storj represents metadata as a set of key–value pairs, and it only contains a fixed set of information such as chunks and encryption. Dat uses metadata feeds to represent metadata of an object, which contains a set of simple file attributes similar to those of UNIX file systems. On the other hand, some proposals support the extensible representation of metadata. Solid, for example, represents every object, including metadata based on Linked Data principles, which can express any kind of attributes such as a relation between objects.

However, existing solutions usually do not specify how to manage metadata clearly. What we have found in metadata management in the solutions are (1) Dat manages a single metadata for each file object, and stores a metadata feed containing whole metadata in the root directory of a Dat repository, and (2) Solid treats metadata (such as object relation) equally as any other objects.

3.4. Privacy

The “Privacy” module is responsible for protecting user privacy. Depending on the sensitive nature of user data, the owner of the data may wish to control what entities (i.e., other users and apps) can access what data. If the owner posts an opinion that can be shared with everyone, there may be no need for access control. However, personal pictures may have different requirements; some pictures can be shared with friends, and other pictures should be shared only with family members. Thus, the “Privacy” module should support the flexible control on which objects can be accessed by which entities.

In addition, a user may wish to hide her real-world identity when publishing her objects. Suppose that a user has published a review of products with her real-world identity (e.g., her name). Depending on the products, it may leak her preferences, religion, political stance, or income level. Therefore, her identity may have to be replaced with a random-looking identifier, such as a hash of her public key. Even the identifier itself is random-looking, a series of activities may leak a user’s real-world identity [69,70]. For example, if a user publishes a series of restaurant reviews along with visiting times, her identity might be revealed. Therefore, it is vital to support a user to easily change her identifier whenever needed.

The “Privacy” module provides the following functions.

- **User data access control:** This function controls which users can access which objects. Access control is done based on access control lists (ACLs) stored in the metadata of the objects.
- **User online behavior protection:** User data includes not only User-Generated Content (UGC) but also the records of users’ online behaviors: e.g., a history of which websites a user has visited and what the user is searching for. The user’s online behaviors should be protected according to the user’s privacy policy.
- **Object encryption/decryption:** When a decentralized application uses a decentralized storage system such as IPFS, an object (or its chunks) is stored on any participating nodes in the network. To prevent the nodes from accessing the object without authorization, it is necessary to encrypt the object. Also, the authorized user/app should be able to decrypt the object.
- **Anonymization:** When an object owner wants to hide her real-world identity, “Privacy” should support anonymization by providing the pseudonymity and unlinkability. Pseudonyms can be used to hide the real identity of the object owners. Moreover, the unlinkability prevents an attacker from tracking a series of activities of the same user; usually, multiple pseudonyms are used for the same user to provide the unlinkability.

Existing solutions have some functions related to the “Privacy” module as follows. Solid supports access control through the Web Access Control (WAC) based on the ACL. YaCy mitigates the privacy leakage issue of online user behaviors by hashing the user’s search keywords in forwarding user requests in the P2P network. Storj deals with object encryption/decryption systematically. Every object in Storj is split into segments, and encryption is performed at the segment level. A user uses different keys for different objects in Storj. For the anonymization, existing solutions (including DID, IPFS, Storj, and Dat) use public keys (or their hashes) for user identifiers. They have a basic anonymization mechanism since they use a public key as the key owner’s pseudonym, but do not provide the unlinkability in the system.

3.5. Application

From the “Locating & Delivery” module to the “Privacy” module, our reference framework covers from ‘how to store/retrieve data objects’ to ‘how to represent metadata’ to ‘how to protect objects from unauthorized accesses.’ The remaining part provides an abstraction to application users, such as how to browse the decentralized Internet and how to find the object distributed on the decentralized Internet, which is called the “Application” module.

The “Application” module can include various functions as needed by end-user application software. Among such functions, to search for objects and to browse the Internet serve as primitive application programming interfaces (APIs).

- **Browsing:** A browsing function shows web pages on the decentralized Internet to users. Browsing on the conventional Internet means accessing a particular server (specified in the URL) using existing end-to-end protocols such as HTTP. However, web pages may be stored across multiple nodes in the decentralized Internet. Thus, browsers for the decentralized Internet should be able to understand protocols that retrieve web pages from multiple points in a decentralized fashion. Furthermore, browsers may optionally cache and serve web pages. In this way, the availability of web pages can be increased.
- **Searching:** A searching function is to find a user’s object of interest among the crawled-and-indexed objects. Conventional search engines collect web pages through crawling, parse web pages to generate indexes, and rank web pages through ranking algorithms. The problem is that the crawled web pages and indexes are consolidated, and the search engine might return biased results by sponsorship or censorship. Thus, a decentralized searching function must allow nodes to crawl, index, and rank the web pages in a decentralized fashion. Note that the indexed results might be ranked by the same or different ranking algorithms.

The existing solutions related to the “Application” module are Beaker and YaCy. The Beaker web browser operates based on the decentralized protocol Dat and provides the caching function for users to seed web pages. YaCy is a decentralized search engine operating on a P2P network, where each node crawls and indexes web pages. As individual nodes run their ranking algorithm, there is no global coordination yet.

3.6. ID

An identifier (ID) is necessary for identifying the following entities in the proposed framework: users (i.e., object owners or object consumers), nodes, objects, chunks, and metadata. To represent the data ownership, we need to specify who (i.e., the user) owns what (i.e., the object). Also, to retrieve the chunks of an object, we need to figure out from where (i.e., the participating node) we can retrieve what (i.e., the object, chunk, or metadata).

- **User ID:** A user ID indicates a user in the decentralized Internet such as an object owner or an object consumer. A representative ID scheme in the decentralized Internet is DIDs [14]. Users of the decentralized Internet need public key pairs, for example, to sign their objects. In DIDs, the public key of a user is included in her DID document. For instance, we can use the hash value of the user's public key as the user's DID. Then, we can validate (i) the user's DID by comparing with the hashed value of the public key, and (ii) the authenticity of objects (signed by the user's private key) with the public key.
- **Node ID:** A node ID is needed to identify a participating node. In the decentralized Internet, nodes contribute their resources (e.g., storage, bandwidth) and get rewards. Thus, nodes need a public key pair since they should authenticate themselves when they make contributions and get rewards. As similar to user IDs, DIDs can be adopted for node IDs. Furthermore, a node can describe its own properties, such as its link bandwidth, storage space, and unit cost of storage in its DID document.
- **Object ID:** An object ID represents an object such as a web page or an image file. An object ID can be generated as its hash value (e.g., content identifier (CID) [71]). A user that retrieves a particular object can verify its integrity with its object ID. In the CID scheme, the ID of an object contains the hash algorithm, the encoding scheme, and the type of the object (e.g., image, text) as well as its hash value.
- **Chunk ID:** A chunk ID identifies a chunk that constitutes an object. The CID scheme can also be used to indicate a chunk.
- **Metadata ID:** There are two main approaches for constructing a metadata ID: (1) the hash value of the metadata itself and (2) the derived form from the corresponding object ID. The former is similar to the case of an object ID and a chunk ID, which helps to prevent tampering. The latter may be constructed by adding a prefix or suffix to its object ID. Thus, we can easily verify whether the metadata is related to the corresponding object.

Recall that the existing solutions support only a subset of the above IDs. The DIDs scheme is appropriate for user IDs as well as node IDs, while the CID scheme is more relevant to object IDs or chunk IDs.

3.7. Incentivization

For operating applications and systems in the decentralized Internet, resources from individuals and organizations are needed, which highlights the importance of the Incentivization module. Another kind of contribution from users for decentralized applications is user data: UGC or online behaviors. The use of the user data may also require compensation. One of the notable incentive examples is Bitcoin [75].

Bitcoin issues its token as a reward to operate its blockchain system by introducing a mining mechanism. The token incentivizes a miner to contribute by finding a nonce to construct a valid block in the ledger. As a token is issued for each block, Bitcoin is working as an economic system by circulating the tokens.

The following are the functional requirements of incentive systems for the framework.

- **Incentives for resources:** Rewards for storage and delivery of data are required for operating decentralized Internet services. The reward for computing data may also be given when computational tasks are delegated to nodes that cannot benefit from the tasks. A decentralized application service provider should devise how to pay for the resources based on the resource usage.
- **Incentives for user data:** In the current Internet, data (UGC) creators are increasingly compensated, usually depending on the popularity of their content. Youtube may be a notable example as its platform shares the advertisement profit with its content uploaders. However, data about online user behaviors are not normally compensated, which should be addressed. The Incentivization module in the reference framework should keep track of the data usage (of both UGC and online behaviors) based on data ownership and access control, and an application service provider should establish a reward system based on the usage information.

Some of the decentralized Internet solutions include incentive mechanisms. For instance, Storj has an incentive mechanism that rewards storing and delivering data, whereas IPFS does not have any incentives. Instead, Filecoin [74] based on the IPFS architecture incentivizes storing data continuously for persistent data storage [79].

Users get rewarded for user-created data on some decentralized data platforms. Steemit [76], a blockchain-based blogging and social media platform, rewards users with its own tokens for creating and curating content based on voting and their token shares (i.e., capital contributions). DTube [77], a decentralized platform for uploading and sharing video, rewards users with its tokens based on voting from other users.

3.8. Summary of the reference framework

The reference framework is intended to encompass the comprehensive issues of the decentralized Internet. Thus, the existing solutions are deemed to cover a subset of functions in the reference framework. A summary of each module of the reference framework is presented in Table 3.

"**Locating & Delivery**" is responsible for finding and delivering data in the decentralized Internet, requiring three functions: locating objects, delivering objects, and checking the integrity of objects. They are partially covered in IPFS, Storj, Dat, and Blockstack. "**Strategy**" improves the availability of data through object replication, node entrance control, and node reputation management. Some of these functions are performed in IPFS and Storj. "**Metadata**" defines how metadata is represented and managed. Some of metadata-related functions are implemented in IPFS, Dat, and Solid. "**Privacy**" preserves user privacy and maintains the confidentiality of data. This module requires access control, privacy protection of online users, object encryption/decryption, and anonymization functions. Solid and Storj have access control and encryption/decryption functions, and YaCy provides protection of online user behaviors. "**Application**" provides the necessary functions for end-user application requirements. Browsing and search engines are introduced as representative functions. Currently, web browsing is provided by Beaker and searching is provided by YaCy. "**ID**" deals with how to assign IDs to users, nodes, objects, chunks, and metadata in a decentralized fashion. Individual ID schemes have their own structures for the entities of their interest. For instance, DID focuses on how to define identifiers of users and nodes and how to verify their public keys. "**Incentive**" aims to promote peers' participation in a given system for higher availability. Users contribution to the system is typically evaluated in terms of the storage capacity, network bandwidth, or computing power. Storage incentives are considered in IPFS; computation incentives are main drivers in cryptocurrency systems such as Bitcoin; and user data forms the basis of incentivization in Steemit, DTube, and Brave.

Table 3
Summary of the reference framework.

Module	Description	Required functions	Related work	Research issues
Locating & delivery	Finds where data (and a node) is located & deliveries the data	– Object locating – Object delivery – Object integrity	IPFS [13], Storj [25], Dat [38], Blockstack [46]	– Inefficiency of DHT
Strategy	Improves availability of data	– Replication – Entrance control – Reputation management	IPFS [13], Storj [25], SAFE Network [68]	– Efficiency of replication – Reputation management overhead
Metadata	Describes how to represent metadata to specify the properties of data and how to manage it	– Metadata representation – Metadata management	IPFS [13], Dat [38], Solid [28]	– Efficiency of access control – Metadata ID construction – Version control
Privacy	Preserves the user privacy and the data confidentiality	– User data access control – User online behavior protection – Object encryption/decryption – Anonymization	Solid [28], Storj [25], YaCy [42]	– Key management – Unlinkability
Application	Provides various functions as needed by end-user application software	– Browsing – Searching	Beaker [41], YaCy [42]	– Coordinated crawling & indexing – Coordination of ranking – High search latency
ID	Specifies to whom/which ID is assigned & what we can do with an ID	– User IDs – Node IDs – Object IDs – Chunk IDs – Metadata IDs	DID [14], Sovrin [48], lifeID [72], Veres One [73]	– Compatibility – Trust model
Incentivization	Motivates peers to participate in the system operations, who contribute resources or data	– Storage incentive – Delivery incentive – Computation incentive – Data incentive	IPFS (Filecoin) [74], Bitcoin [75], Steemit [76], DTube [77], Brave rewards [78]	– Amounts of reward – Methods of reward – Sources of reward

4. Challenging issues on the decentralized internet

In this section, we highlight challenging issues that are not covered or poorly performed by the existing technologies for each module of the reference framework in Section 3.

4.1. Issues on locating & delivery

The goal of "Locating & Delivery" is storing and retrieving objects in a decentralized fashion. DHT is the most popular mechanism for this goal; however, the real-world scalability in a pure DHT is still in question [80,81].

- **Inefficiency of DHT:** The inefficiency of DHT due to overlay routing is still a drawback. Since DHT does not consider geographical locality, a query might go back or forth across the globe [80]. While some recent schemes suggest DHT modifications to alleviate such problems, its inherent inefficiency results in substantial delays when locating objects.

4.2. Issues on strategy

"Strategy" is responsible for increasing the availability of objects. The higher availability can be achieved by managing the reputation of nodes, and storing multiple replications of objects. In order to enhance the availability of system, there are the following issues:

- **Efficiency of replication:** As the number of copies increases, the availability improves. However, at the same time, it wastes storage resources. A system should be designed by considering a trade-off between the storage efficiency and the availability level. It is worth considering the reputation (or trustworthiness) of nodes in deciding the redundancy level of replication.
- **Reputation management overhead:** The system should keep track of the reputation of nodes, which should be accessible by all participating nodes. A simple way to manage the reputation is to have a central node to keep track of all nodes' reputation, which is however another type of data consolidation. In order to manage reputation in a fully decentralized fashion, each node needs to maintain/manage the reputation of other nodes in the network. The trade-off between the management overhead and the concentration may also have to be balanced.

4.3. Issues on metadata

"Metadata" specifies how objects are represented and managed. We have identified the following issues that require further research:

- **Efficiency of access control:** One straightforward way to implement access control is to maintain an access control list for each object. The access control list of a given object determines who is authorized to access (e.g., read or write) the object. Solid takes a similar approach to the access control list. However, Solid's management overhead of access control lists is potentially in proportion to the number of objects and the number of users. To reduce the overhead, one alternative is to group users depending on their relations to the owner (say, family, friend, or colleague) in the access control list. Note that there is a trade-off between the granularity and the overhead of access control.
- **Metadata ID construction:** As the metadata has different characteristics from user data, they may have to be handled differently from other objects. We believe that there are two main approaches on how to construct metadata IDs. First, like data objects (in most solutions), a metadata ID is the hash of its content. In this way, tampering with the attributes of the metadata is not easy. However, in such cases, the relation between the data object and its corresponding metadata object must be managed and retrieved by an additional mapping mechanism. Second, the metadata ID can be derived from the corresponding object ID (e.g., prepending a prefix or appending a suffix to the object ID). With such constructions, we need a mechanism to avoid tampering with the metadata. For instance, the metadata may have to include the owner's signature, or the owner of the metadata (and its object) encrypts the metadata.
- **Version control:** When multiple users edit an object, and its multiple versions are derived, it should be determined how to manage their corresponding metadata. When multiple versions of an object are created, there are two main approaches to manage the metadata. The first is to give different IDs to the updated

objects and generate metadata for each object. In other words, a new version of the object is managed as if a new object is created. In this case, the relations among the different versions may have to be managed by a separate mechanism depending on application requirements. For example, the version information can be indicated in the object relation inside the metadata. Integrity is easily guaranteed since different versions can use their hash values as object IDs. The second is to keep the same object ID for all the updated versions like Dat. In this case, the metadata is generated for each new version, and the metadata of each version describes and points to the corresponding version. Thus, the relations between different versions can be maintained through the same object ID.

4.4. Issues on privacy

"Privacy" is essential for protecting users' private information. Since the level of allowed exposure may vary based on the sensitivity of the data, in some cases, complete concealment of user information may be necessary. While current solutions offer privacy-related functions, recent studies such as [69,70] indicate that users' real-world identities can still be exposed even with pseudonyms. Hence, several challenges remain unaddressed by the existing solutions, which are outlined below.

- **Unlinkability:** Using a public key (or its hash) as an identifier provides only basic anonymization; that is, an attacker might be able to reveal a user's real-world identity by tracking the behaviors of the same identifier. Currently, there is no work to provide the unlinkability in the decentralized solutions in the literature. It is desirable to support a user to use different pseudonyms systematically in such a way that she need not be aware of using different pseudonyms.
- **Key management:** Object encryption/decryption should be supported to provide object confidentiality. It is essential to use different keys for different objects. One of the well-known problems in this situation is for the data owner to securely distribute keys only to authorized users/apps. For instance, Storj stores the cipher information of encryption/decryption in a trusted database in the Satellite. However, Storj does not specify how to manage the keys.

4.5. Issues on application

"Application" is responsible for providing end-user services with the necessary functions to browse and search for objects in the decentralized Internet. However, there are still the following challenges.

- **Coordinated crawling & indexing:** The decentralized crawling and indexing functions are performed by nodes in a P2P network, where each node crawls and indexes web pages independently, as YaCy does. However, this approach results in crawling the same web page multiple times from a network-wide view. For reducing such inefficient crawling and indexing, it is necessary to coordinate the nodes so that the target ranges of crawling are not overlapping.
- **Coordination of ranking algorithms:** In the case of YaCy, each node crawls web pages and generates the indexes of the web pages. Also, each node runs a ranking algorithm to rank the indexed web pages on its own. However, the comprehensiveness[6] of crawled web pages of a single node is limited. Thus, the coverage of search results will be improved if the indexed results of participating nodes are merged in a coordinated manner [82]. If the same ranking algorithm is used across the nodes, abusing/biasing search results might be easier. If different ranking algorithms are used, the coordination among the search results will be crucial, not yet proposed in the literature.
- **High search latency:** YaCy maintains a DHT for keywords. As mentioned in the "Locating & delivery" module, a DHT lookup can result in a substantial delay. By contrast, the result of a DHT lookup for a search query is the list of peers who have the indexes for the keywords in the query. Thus, to get the final result consisting of relevant URLs, the client should set up connections to the peers, which incurs additional delays.

4.6. Issues on ID

"ID" of the decentralized Internet encompasses not only users but also nodes, objects, chunks, and metadata. The following aspects should be further explored.

- **Compatibility:** In order to make decentralized systems using the above IDs interwork with the current Internet applications and systems, at least some of the IDs need to be compatible with the well-known protocols currently in use. For example, if user IDs or node IDs based on DIDs are used for secure applications, they might need data structures that are compatible with X.509 certificates used in Transport Layer Security (TLS) [83]. To this end, we may upgrade TLS to be aware of DIDs, or create a new DID-enabled certificate that can be used in TLS.
- **Trust model:** The trust model for online identities on the conventional Internet is built around certificate authorities (CAs) and the public key infrastructure (PKI). Since this trust model is a centralized model, we need a new one for decentralized applications and networking. However, existing ID-related solutions, such as DIDs, do not provide a trustworthy framework in which anyone can verify whether online identities match real-world ones. The web-of-trust model [84] is well known for no centralized authority; however, several challenges should be addressed [85,86]. For example, a new certificate typically cannot have enough endorsements.

4.7. Issues on incentivization

"Incentivization" improves the availability by motivating participants to contribute their resources more into the system. Although Steemit and DTube use coin or token systems to incentivize users, they cover only a small area of Internet applications like social networking or video platforms. Thus, there are several design issues for incentivization that need to be addressed:

- **Amount of rewards:** Designing an incentive system is not simple. It is sophisticated to design an economic system, and it is important to strike a balance between payments and rewards for resources. While users of a decentralized application usually do not make commitments, a given system should make critical resources for system operations be supplied from its service provider or from the resources of users in a reliable fashion (say, a smart contract). For the data incentives, the amount of payout can be decided by auctioning or a per-service policy.
- **Method of rewards:** For distributed networking systems, the method of rewards can be categorized into two types: implicit and explicit. The implicit rewards have been proposed for decentralized systems, e.g., peer-selection [21,87] and reputation mechanisms [88] in P2P systems. In the case of peer-selection (e.g., tit-for-tat schemes), contributors to the system are rewarded with more chances to be selected, resulting in better performance. As to the reputation mechanism, a reputation score is calculated based on the history of a peer's activities. Then, the peer will be incentivized to increase her score. The explicit reward typically

[6] It indicates, for a given keyword, how much of the relevant web pages are indexed.

takes a form of a token system based on a cryptocurrency like Ethereum. By designing issuing and allocating mechanisms of tokens, an explicit reward system for a decentralized application can be established without a central entity.

- **Sources of rewards:** A survey on blockchain incentive mechanisms [89] shows that the existing blockchains (or their applications) normally have issuing and allocating mechanisms of tokens. That is, decentralized applications can have internal sources of rewards. Otherwise, monetary rewards should be provisioned from external sources, e.g., application service providers, subscription fees (periodical fee or pay-per-view), or advertisement/marketing.

5. Conclusions

This paper makes the following contributions:

1. Investigating the data consolidation problem: We investigated the data consolidation problem on the current Internet. While legal regulations have been made to mitigate the problem, we emphasized that they cannot be fundamental solutions.
2. Surveying the existing solutions for decentralization: We surveyed the existing technical solutions for decentralization, which are classified by their similarity. We also explained their differences and limitations. We found that each solution addresses its own scope of problems, usually from a limited point of view.
3. Suggesting the reference framework: To address the limitations of the existing solutions, we designed the holistic reference framework for the decentralized Internet. The reference framework considers not only networking functions (e.g., locating/delivering a particular object, reliable data transmissions), but also user/system requirements such as user privacy and incentivization. In the decentralized Internet, the owner of user data should have sovereignty; thus, functions such as access control and privacy protection are required. Additionally, ID systems should be decentralized to prevent the single point of dependency and power concentration.
4. Presenting future research issues: We analyzed which solutions cover which functions in each module of the reference framework for the decentralized Internet. Base on this, we presented challenging issues and directions for further research by identifying functions that cannot be met by the related work in Section 4.

While we introduced the technical approaches to decentralization, the data consolidation problem is also a socio-economic issue [4]. Indeed, governments of many countries are enacting legislations to counter data consolidation. Despite these efforts, the standardization efforts of legal and technical solutions to the decentralized Internet are still insufficient and often not cooperative across the world. We expect that the survey of technical solutions for addressing the data consolidation problem and the suggestion of the reference framework for general decentralized applications in this paper will facilitate further studies towards the international standardization of the decentralized Internet.

CRediT authorship contribution statement

Ted "Taekyoung" Kwon: Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision, Project administration. **Junghwan Song:** Investigation, Methodology, Writing – original draft, Writing – review & editing, Supervision. **Heeyoung Jung:** Conceptualization, Methodology, Supervision, Funding acquisition. **Selin Chun:** Investigation, Writing – original draft. **Hyunwoo Lee:** Investigation, Writing – original draft. **Minhyeok Kang:** Investigation, Writing – original draft. **Minkyung Park:** Investigation, Writing – original draft. **Eunsang Cho:** Investigation, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the ICT R&D program of MSIT/IITP (Ministry of Science and ICT/Institue of Information & Communications Technology Planning & Evaluation), Republic of Korea [2017-0-00045, Hyper-connected Intelligent Infrastructure Technology Development], MSIT under the ITRC (Information Technology Research Center), Republic of Korea support program [IITP-2023-2021-0-02048] supervised by the IITP, National Research Foundation of Korea(NRF) grant funded by the Korea government (MSIT) [NRF-2022R1A2C2011221, No. RS-2023-00220985].

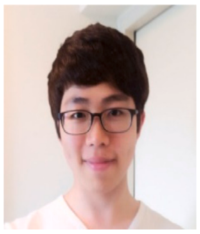
References

[1] State of the IoT 2020, 2023, https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/ (Accessed: 09 April 2023).
[2] World population and internet users, 2023, https://www.worldometers.info/ (Accessed: 09 April 2023).
[3] Internet health report 2018, 2023, https://internethealthreport.org/2018/ (Accessed: 09 April 2023).
[4] ISOC global internet report 2019, 2023, https://future.internetsociety.org/2019/ (Accessed: 09 April 2023).
[5] Search engine market share worldwide 2019, 2023, https://gs.statcounter.com/search-engine-market-share/ (Accessed: 09 April 2023).
[6] Browser market share worldwide 2019, 2023, https://gs.statcounter.com/browser-market-share/ (Accessed: 09 April 2023).
[7] Social media market share worldwide 2019, 2023, https://gs.statcounter.com/social-media-stats/ (Accessed: 09 April 2023).
[8] Major global digital ad sellers 2019 net digital ad revenues, 2023, https://www.emarketer.com/content/global-digital-ad-spending-2019/ (Accessed: 09 April 2023).
[9] The world's most valuable resource, 2017, pp. 14–17, The Economist, May 2017.
[10] H. Jung, N. Koh, Survey on the data consolidation issue in the internet and corresponding researches, in: Electronics and Telecommunications Trends, ETRI, 2019, http://dx.doi.org/10.22648/ETRI.2019.J.340611.
[11] EU gdpr(general data protection regulation), 2023, https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN (Accessed: 09 April 2023).
[12] J. Selby, Data localization laws: Trade barriers or legitimate responses to cybersecurity risks, or both? Int. J. Law Inform. Technol. 25 (3) (2017) 213–232, http://dx.doi.org/10.1093/ijlit/eax010.
[13] ProtocolLabs, IPFS documentation, 2023, https://docs.ipfs.io/ (Accessed: 09 April 2023).
[14] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, Decentralized identifiers (DIDs) v1.0, 2023, https://w3.org/TR/did-core/ (Accessed: 09 April 2023).
[15] D. Clark, The design philosophy of the DARPA internet protocols, ACM SIGCOMM Comput. Commun. Rev. 18 (4) (1988) 106–114, http://dx.doi.org/10.1145/52324.52336.
[16] L.-D. Ibáñez, E. Simperl, F. Gandon, H. Story, Redecentralizing the web with distributed ledgers, IEEE Intell. Syst. 32 (1) (2017) 92–95, http://dx.doi.org/10.1109/MIS.2017.18.
[17] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, ACM SIGCOMM Comput. Commun. Rev. 31 (4) (2001) 149–160, http://dx.doi.org/10.1145/964723.383071.
[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content-addressable network, in: Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, 2001, pp. 161–172, http://dx.doi.org/10.1145/383059.383072.
[19] A. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, in: IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer, 2001, pp. 329–350, http://dx.doi.org/10.1007/3-540-45518-3_18.

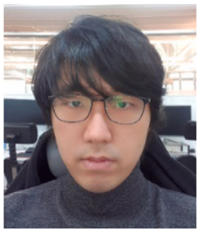
[20] B.Y. Zhao, J. Kubiatowicz, A.D. Joseph, et al., Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing, Tech. Report No. UCB/CSD-01-1141, Computer Science Division, University of California Berkeley, 2001.
[21] B. Cohen, Incentives build robustness in BitTorrent, in: Workshop on Economics of Peer-To-Peer Systems, Vol. 6, 2003, pp. 68–72.
[22] D. Bleichenbacher, U.M. Maurer, Directed acyclic graphs, one-way functions and digital signatures, in: Annual International Cryptology Conference, Springer, 1994, pp. 75–82, http://dx.doi.org/10.1007/3-540-48658-5_9.
[23] P. Maymounkov, D. Eres, Kademlia: A peer-to-peer information system based on the XOR metric, vol. 2429, 2002, http://dx.doi.org/10.1007/3-540-45748-8_5.
[24] ProtocolLabs, Bitswap documentation, 2023, https://docs.ipfs.io/concepts/bitswap/ (Accessed: 09 April 2023).
[25] S. Wilkinson, T. Boshevski, J. Brandoff, V. Buterin, Storj a Peer-to-Peer Cloud Storage Network, Citeseer, Princeton, NJ, USA, 2014.
[26] I.S. Reed, G. Solomon, Polynomial codes over certain finite fields, 8, (2) SIAM, 1960, pp. 300–304,
[27] L. Rizzo, Effective erasure codes for reliable computer communication protocols, ACM SIGCOMM Comput. Commun. Rev. 27 (2) (1997) 24–36, http://dx.doi.org/10.1145/263876.263881.
[28] E. Mansour, A.V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga, T. Berners-Lee, A demonstration of the solid platform for social web applications, in: Proceedings of the 25th International Conference Companion on World Wide Web, International World Wide Web Conferences Steering Committee, 2016, pp. 223–226, http://dx.doi.org/10.1145/2872518.2890529.
[29] M. Zignani, C. Quadri, S. Gaito, H. Cherifi, G.P. Rossi, The footprints of a "mastodon": How a decentralized architecture influences online social relationships, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, IEEE, 2019, pp. 472–477, http://dx.doi.org/10.1109/INFCOMW.2019.8845221.
[30] M. Sporny, T. Inkster, H. Story, B. Harbulot, R. Bachmann-Gmür, Webid 1.0: Web Identification and Discovery, W3C Editors Draft, 2011.
[31] C. Bizer, T. Heath, T. Berners-Lee, Linked data: The story so far, in: Semantic Services, Interoperability and Web Applications: Emerging Concepts, IGI Global, 2011, pp. 205–227, http://dx.doi.org/10.4018/978-1-60960-593-3.ch008.
[32] Mastodon, 2023, https://joinmastodon.org/ (Accessed: 09 April 2023).
[33] M. Zignani, S. Gaito, G.P. Rossi, Follow the "mastodon": Structure and evolution of a decentralized online social network, in: Twelfth International AAAI Conference on Web and Social Media, 2018, http://dx.doi.org/10.1609/icwsm.v12i1.14988.
[34] A. Raman, S. Joglekar, E.D. Cristofaro, N. Sastry, G. Tyson, Challenges in the decentralised web: The mastodon case, in: Proceedings of the Internet Measurement Conference, 2019, pp. 217–229, http://dx.doi.org/10.1145/3355369.3355572.
[35] C.L. Webber, J. Tallon, E. Shepherd, A. Guy, E. Prodromou, ActivityPub, W3C, 2018.
[36] J. Snell, E. Prodromou, Activity Streams 2.0, W3C Rec, 2017.
[37] M. Sporny, D. Longley, G. Kellogg, M. Lanthaler, P.-A. Champin, N. Lindström, JSON-LD 1.1–a JSON-based serialization for Linked Data (Ph.D. thesis), W3C, 2019.
[38] C. for Science, Dat whitepaper, 2023, https://github.com/datprotocol/whitepaper/raw/master/dat-paper.pdf (Accessed: 09 April 2023).
[39] J.-P. Aumasson, S. Neves, Z. Wilcox-O'Hearn, C. Winnerlein, BLAKE2: Simpler, smaller, fast as MD5, in: International Conference on Applied Cryptography and Network Security, Springer, 2013, pp. 119–135, http://dx.doi.org/10.1007/978-3-642-38980-1_8.
[40] D.J. Bernstein, Extending the Salsa20 nonce, in: Workshop Record of Symmetric Key Encryption Workshop, Vol. 2011, 2011.
[41] Beaker browser, 2023, https://beakerbrowser.com/ (Accessed: 09 April 2023).
[42] YaCy - the peer to peer search engine, 2023, https://yacy.net/ (Accessed: 09 April 2023).
[43] M. Herrmann, K.-C. Ning, C. Diaz, B. Preneel, Description of the Yacy Distributed Web Search Engine, Technical Report, KU Leuven ESAT/COSIC, IMinds, 2014.
[44] W.B. Croft, D. Metzler, T. Strohman, Search Engines: Information Retrieval in Practice, Vol. 520, Addison-Wesley Reading, 2010.
[45] M. Ali, J. Nelson, R. Shea, M.J. Freedman, Blockstack: A global naming and storage system secured by blockchains, in: 2016 USENIX Annual Technical Conference, USENIX ATC 16, 2016, pp. 181–194.
[46] M. Ali, J. Nelson, A. Blankstein, Blockstack technical whitepaper v 2.0, 2023, https://silverway.io/whitepaper.pdf (Accessed: 09 April 2023).
[47] M. Swan, Blockchain: Blueprint for a New Economy, "O'Reilly Media, Inc.", 2015.
[48] T.S. Foundation, Sovrin: A protocol and token for self-sovereign identity and decentralized trust, 2023, https://sovrin.org/wp-content/uploads/Sovrin-Protocol-and-Token-White-Paper.pdf (Accessed: 09 April 2023).
[49] M. Sabadello, A universal resolver for self-sovereign identifiers, 2023, https://medium.com/decentralized-identity/a-universal-resolver-for-self-sovereign-identifiers-48e6b4a5cc3c (Accessed: 09 April 2023).
[50] K. Rannenberg, J. Camenisch, A. Sabouri, Attribute-based credentials for trust, in: Identity in the Information Society, Springer, 2015, doi:10.1007%2F978-3-319-14439-9.
[51] K. Yue, Y. Zhang, Y. Chen, Y. Li, L. Zhao, C. Rong, L. Chen, A survey of decentralizing applications via blockchain: The 5g and beyond perspective, IEEE Commun. Surv. Tutor. 23 (4) (2021) 2191–2217, http://dx.doi.org/10.1109/COMST.2021.3115797.
[52] K. Li, Y. Yang, S. Wang, R. Shi, J. Li, A lightweight privacy-preserving and sharing scheme with dual-blockchain for intelligent pricing system of smart grid, Comput. Secur. 103 (2021) 102189, http://dx.doi.org/10.1016/j.cose.2021.102189.
[53] H. Kim, J. Park, M. Bennis, S.-L. Kim, Blockchained on-device federated learning, IEEE Commun. Lett. 24 (6) (2019) 1279–1283, http://dx.doi.org/10.1109/LCOMM.2019.2921755.
[54] Z. Li, M. Alazab, S. Garg, M.S. Hossain, PriParkRec: Privacy-preserving decentralized parking recommendation service, IEEE Trans. Veh. Technol. 70 (5) (2021) 4037–4050, http://dx.doi.org/10.1109/TVT.2021.3074820.
[55] R. Singh, A.D. Dwivedi, R.R. Mukkamala, W.S. Alnumay, Privacy-preserving ledger for blockchain and Internet of Things-enabled cyber-physical systems, Comput. Electr. Eng. 103 (2022) 108290, http://dx.doi.org/10.1016/j.compeleceng.2022.108290.
[56] M.S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli, M.H. Rehmani, Applications of blockchains in the Internet of Things: A comprehensive survey, IEEE Commun. Surv. Tutor. 21 (2) (2018) 1676–1717, http://dx.doi.org/10.1109/COMST.2018.2886932.
[57] N.Z. Benisi, M. Aminian, B. Javadi, Blockchain-based decentralized storage networks: A survey, J. Netw. Comput. Appl. 162 (2020) 102656, http://dx.doi.org/10.1016/j.jnca.2020.102656.
[58] J. Zhang, S. Zhong, T. Wang, H.-C. Chao, J. Wang, Blockchain-based systems and applications: A survey, J. Internet Technol. 21 (1) (2020) 1–14, http://dx.doi.org/10.3966/160792642020012101001.
[59] K. Yue, Y. Zhang, Y. Chen, Y. Li, L. Zhao, C. Rong, L. Chen, A survey of decentralizing applications via blockchain: The 5g and beyond perspective, IEEE Commun. Surv. Tutor. 23 (4) (2021) 2191–2217, http://dx.doi.org/10.1109/COMST.2021.3115797.
[60] C. Antal, T. Cioara, I. Anghel, M. Antal, I. Salomie, Distributed ledger technology review and decentralized applications development guidelines, Future Internet 13 (3) (2021) 62, http://dx.doi.org/10.3390/fi13030062.
[61] N. Six, N. Herbaut, C. Salinesi, Blockchain software patterns for the design of decentralized applications: A systematic literature review, Blockchain: Res. Appl. (2022) 100061, http://dx.doi.org/10.1016/j.bcra.2022.100061.
[62] O. Abdullah Lajam, T. Ahmed Helmy, Performance evaluation of IPFS in private networks, in: 2021 4th International Conference on Data Storage and Data Engineering, DSDE '21, Association for Computing Machinery, New York, NY, USA, 2021, pp. 77–84, http://dx.doi.org/10.1145/3456146.3456159.
[63] D. Stutzbach, R. Rejaie, Understanding churn in peer-to-peer networks, in: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, 2006, pp. 189–202, http://dx.doi.org/10.1145/1177080.1177105.
[64] J.R. Douceur, The sybil attack, in: International Workshop on Peer-to-Peer Systems, Springer, 2002, pp. 251–260, http://dx.doi.org/10.1007/3-540-45748-8_24.
[65] P. Dewan, P. Dasgupta, Pride: peer-to-peer reputation infrastructure for decentralized environments, in: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters, 2004, pp. 480–481, http://dx.doi.org/10.1145/1013367.1013535.
[66] S.Y. Lee, O.-H. Kwon, J. Kim, S.J. Hong, A reputation management system in structured peer-to-peer networks, in: 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise, WETICE'05, IEEE, 2005, pp. 362–367, http://dx.doi.org/10.1109/WETICE.2005.9.
[67] I.S. Reed, G. Solomon, Polynomial codes over certain finite fields, J. Soc. Ind. Appl. Math. 8 (2) (1960) 300–304, http://dx.doi.org/10.1137/0108018.
[68] SAFE network, 2023, https://safenetwork.tech/ (Accessed: 09 April 2023).
[69] R. Lambiotte, M. Kosinski, Tracking the digital footprints of personality, Proc. IEEE 102 (12) (2014) 1934–1939, http://dx.doi.org/10.1109/JPROC.2014.2359054.
[70] P. Reynolds, A.S. Irwin, Tracking digital footprints: Anonymity within the bitcoin system, J. Money Laund. Control (2017) http://dx.doi.org/10.1108/JMLC-07-2016-0027.
[71] ProtocolLabs, CID (content identifier), 2023, https://github.com/multiformats/cid (Accessed: 09 April 2023).
[72] lifeID, lifeID: Digital identify simple & secure, 2023, https://lifeid.io/ (Accessed: 09 April 2023).
[73] V. One, Veres one: A globally interoperable blockchain for identity, 2023, https://veres.one/ (Accessed: 09 April 2023).
[74] ProtocolLabs, Filecoin: A decentralized storage network, 2023, https://filecoin.io/filecoin.pdf (Accessed: 09 April 2023).
[75] S. Nakamoto, et al., Bitcoin: A peer-to-peer electronic cash system, 2008.
[76] Steemit, 2023, https://steemit.com/ (Accessed: 09 April 2023).
[77] Dtube whitepaper, 2023, https://token.d.tube/whitepaper.pdf (Accessed: 09 April 2023).
[78] Brave rewards, 2023, https://brave.com/brave-rewards/ (Accessed: 09 April 2023).
[79] IPFS and filecoin, 2023, https://docs.filecoin.io/basics/how-storage-works/filecoin-and-ipfs/ (Accessed: 09 April 2023).

[80] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, I. Stoica, The impact of DHT routing geometry on resilience and proximity, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '03, ACM, New York, NY, USA, 2003, pp. 381–394, http://dx.doi.org/10.1145/863955.863998.
[81] J.S. Kong, J.S. Bridgewater, V.P. Roychowdhury, A general framework for scalability and performance analysis of DHT routing systems, in: International Conference on Dependable Systems and Networks, DSN'06, IEEE, 2006, pp. 343–354, http://dx.doi.org/10.1109/DSN.2006.4.
[82] A. Broder, M. Fontura, V. Josifovski, R. Kumar, R. Motwani, S. Nabar, R. Panigrahy, A. Tomkins, Y. Xu, Estimating corpus size via queries, in: Proceedings of the 15th ACM International Conference on Information and Knowledge Management, ACM, 2006, pp. 594–603, http://dx.doi.org/10.1145/1183614.1183699.
[83] E. Rescorla, The transport layer security (TLS) protocol version 1.3, 2018.
[84] P.R. Zimmermann, P.R. Zimmermann, The Official PGP User's Guide, Vol. 5, MIT press Cambridge, 1995.
[85] J. Golbeck, Weaving a web of trust, Science 321 (5896) (2008) 1640–1641, http://dx.doi.org/10.1126/science.1163357.
[86] A. Ulrich, R. Holz, P. Hauck, G. Carle, Investigating the openpgp web of trust, in: European Symposium on Research in Computer Security, Springer, 2011, pp. 489–507, http://dx.doi.org/10.1007/978-3-642-23822-2_27.
[87] A. Habib, J. Chuang, Incentive mechanism for peer-to-peer media streaming, in: Twelfth IEEE International Workshop on Quality of Service, 2004 IWQOS 2004, 2004, pp. 171–180, http://dx.doi.org/10.1109/IWQOS.2004.1309377.
[88] S. Marti, H. Garcia-Molina, Taxonomy of trust: Categorizing P2P reputation systems, Comput. Netw. 50 (4) (2006) 472–484, http://dx.doi.org/10.1016/j.comnet.2005.07.011.
[89] J. Huang, K. Lei, M. Du, H. Zhao, H. Liu, J. Liu, Z. Qi, Survey on blockchain incentive mechanism, in: Data Science: 5th International Conference of Pioneering Computer Scientists, Engineers and Educators, ICPCSEE 2019, Guilin, China, September 20–23, 2019, Proceedings, Part I 5, Springer, 2019, pp. 386–395, http://dx.doi.org/10.1007/978-981-15-0118-0_30.

Ted "Taekyoung" Kwon received the B.S., M.S., and Ph.D. degrees from Seoul National University (SNU) in 1993, 1995, and 2000, respectively. He is a professor with the Department of Computer Science and Engineering, Seoul National University. Before joining SNU, he was a postdoctoral research associate at the University of California Los Angeles and City University New York. During his graduate program, he was a visiting student at the IBM T.J. Watson Research Center and at the University of North Texas. He was a visiting professor at Rutgers University in 2010. His research interest lies in future Internet, network security, and wireless networks.


Junghwan Song received B.S. in the college of Information & Communication from Korea University in Republic of Korea at 2012. He is currently pursuing an integrated M.S. and Ph.D degree at Seoul National University in Republic of Korea. He has researched on network architecture, especially on future Internet (Content-Centric Networking, Software-Defined Networking, Decentralized Internet, etc.).

Heeyoung Jung joined Electronics and Telecommunications Research Institutes (ETRI) in Korea in 1991 and is a principal research member. He received his Ph.D. degree in information and communications engineering from the Chungnam National University in Korea in 2004. His major research areas have included the Internet/Future Internet and mobile network technologies and been closely related to standardization activities in ITU-T, IETF/IRTF, etc. His current research topic is data sovereignty issue in data-driven future society.


Selin Chun received B.S. degree in the computer science from Seoul National University in 2015. He is currently pursuing an integrated M.S and Ph.D degree at in the computer science at Seoul National University. His research interest lies in analyzing user behaviors or user-generated texts (e.g. posts, comments) in online environment, measuring political bias in the online news articles, auditing the algorithm of online platforms such as YouTube.

Hyunwoo Lee received B.S. and M.S./Ph.D degrees from Seoul National University in 2011, 2020 respectively. He is currently a postdoc research associate at Purdue university. His research interests lie in applied cryptography, network security, and distributed systems.

Minhyeok Kang received B.S. in the college of Computer Science & Engineering from Seoul National University in Republic of Korea at 2017. He is currently pursuing an integrated M.S. and Ph.D degree at Seoul National University in Republic of Korea. He has researched on network security, especially on Internet Backbone Routing Security (Border Gateway Protocol, Resource Public Key Infrastructure, etc.).

Minkyung Park received her B.S. degree in computer science from Korea Aerospace University, Korea, in 2014, and M.S/Ph.D degree in computer science from Seoul National University, Korea, in 2022. She is currently a postdoc researcher at the School of Computer Science and Engineering, Seoul National University, Korea. Her research interests include privacy, user tracking, secure hardware, network security, and anonymity.

Eunsang Cho received his B.S. and Ph.D. in computer science and engineering from Seoul National University. After researching as a postdoctoral researcher at Seoul National University from 2019 to 2021, he has been affiliated with sPresto, Seoul, Korea. His research interests include network security, Internet-of-Things, blockchain, content-centric, and peer-to-peer networking.