IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks

Erik Daniel and Florian Tschorsch

***Abstract*—Decentralized, distributed storage offers a way to reduce the impact of data silos as often fostered by centralized cloud storage. While the intentions of this trend are not new, the topic gained traction due to technological advancements, most notably blockchain networks. As a consequence, we observe that a new generation of peer-to-peer data networks emerges. In this survey paper, we therefore provide a technical overview of the next generation data networks. We use select data networks to introduce general concepts and to emphasize new developments. Specifically, we provide a deeper outline of the Interplanetary File System and a general overview of Swarm, the Hypercore Protocol, SAFE, Storj, and Arweave. We identify common building blocks and provide a qualitative comparison. From the overview, we derive future challenges and research goals concerning data networks.**

***Index Terms*—Data Networks, Blockchain Networks, Peer-to-Peer Networks, Overlay Networks**

I. Introduction

Nowadays, users store and share data by using cloud storage providers in one way or another. Cloud storages are organized centrally, where the storage infrastructure is typically owned and managed by a single logical entity. Such cloud storage providers are responsible for storing, locating, providing, and securing data. While cloud storage can have many economical and technical advantages, it also raises a series of concerns. The centralized control and governance leads to data silos that may affect accessibility, availability, and confidentiality. Data access might, for example, be subject to censorship. At the same time, data silos pose a valuable target for breaches and acquiring data for sale, which risk security and privacy. In general, users lose their self-determined control and delegate it to a cloud provider.

One direction to break free from data silos and to reduce trust assumptions are *peer-to-peer data networks*. The term summarizes a family of data storage approaches that build upon a peer-to-peer (P2P) network and include aspects of data storage, replication, distribution, and exchange. As typical for P2P networks, peers interact directly, build an overlay network, share resources, and can make autonomous local decisions. Consequentially, P2P data networks strive to jointly manage and share storage.

P2P data networks are not a new technology, though. There are many different older P2P networks that can be classified as data networks as well. The popularity of P2P technologies emerged in 1999 with the audio file sharing network Napster, closely followed by Gnutella for sharing all types of files [1]. Napster and Gnutella marked the beginning and were followed by many other P2P networks focusing on specialized application areas or novel network structures. For example, Freenet [2] realizes anonymous storage and retrieval. Chord [3], CAN [4], and Pastry [5] provide protocols to maintain a structured overlay network topology. In particular, BitTorrent [6] received a lot of attention from both users and the research community. BitTorrent introduced an incentive mechanism to achieve Pareto efficiency, trying to improve network utilization achieving a higher level of robustness. We consider networks such as Napster, Gnutella, Freenet, BitTorrent, and many more as first generation P2P data networks, which primarily focus on file sharing.

The recent advancements in P2P technologies and the popularity of the first generation of data networks, affected the areas of distributed file systems [7] and content distribution technologies [8]. This trend also falls under the umbrella of data networks in general and P2P data networks in particular.

One component which seemed to be missing in P2P file sharing systems was a way to improve long-term storage and availability of files. With the introduction of Bitcoin [9] in 2008, the P2P idea in general and the joint data replication in particular gained new traction. Distributed ledger technologies provide availability, integrity, and byzantine fault tolerance in a distributed system. In particular, cryptocurrencies showed their potential as a monetary incentive mechanism in a decentralized environment. These and additional trends and developments, e.g., Kademlia [10] and information-centric networking [11], lead to the invention of what we denote as the next generation of P2P data networks.

In this survey paper, we provide a technical overview of the next generation of P2P data networks. Starting with the introduction of IPFS [12] in 2014, we define this next generation of data networks as systems and concepts for *decentralized sharing and storing of data*, which appeared in the last decade. We show how these new systems are built, how they utilize the experience and research results from previous systems, as well as new developments and advancements over the last decade. We identify building blocks, similarities, and trends of these systems. While some of the systems are building blocks themselves for other applications, e.g., decentralized applications (DApps), we focus on two main system aspects: *content distribution* and *distributed storage*. Furthermore, we provide insights in the incentive mechanisms, deployed for retrieving or storing files, or both. Since many new data networks were developed, we cannot provide a full overview of all data networks. Instead, we focus on a few select systems with sophisticated or unique mechanisms, different use cases, and different degree of content and user privacy. Our overview

Erik Daniel and Florian Tschorsch are with the Department of Distributed Security Infrastructures at Technische Universität Berlin, 10587 Berlin, Germany; e-mail: erik.daniel@tu-berlin.de and florian.tschorsch@tu-berlin.de

focuses on concepts and abstracts from implementation details to extract general insights. Yet, it should be noted that the systems are prone to change due to ongoing development. Our survey paper makes use of a wide range of sources, including peer-reviewed papers, white papers as well as documentations, specifications, and source code.

Specifically, we focus on IPFS [12], Swarm [13], the Hypercore Protocol [14], SAFE [15], Storj [16], and Arweave [17]. In particular, the InterPlanetary File System (IPFS) has gained popularity as storage layer for blockchains [18, 19, 20, 21, 22, 23, 24] and was subject of a series of studies [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]. Furthermore, we put our overview of these systems in context to preceding systems and research directions, namely BitTorrent, information-centric networking, and blockchains. By contrasting precursor systems, we sketch the evolution of data networks and are able to profoundly discuss advancements of the next generation.

Based on this overview, we extract the building blocks and some unique aspects of P2P data networks. While all systems allow distributed content sharing and storage, they seem to focus on either of the aspects. That is, each system aims to serve a slightly different purpose with different requirements and points of focus. This leads to different design decisions in network organization, file look up, degree of decentralization, redundancy, and privacy. For example, Storj aims for a distributed cloud storage while the Hypercore protocol focuses on distributing large datasets. Similarly, IPFS aims to replace client-server structure of the web and therefore needs a stronger focus on data look up than BitTorrent where mainly each file is located in its own overlay network. At the same time, we found many similarities in the approach of building data networks, for example, using Kademlia to structure the network or finding peers, split files into pieces, or incentivizing different tasks to increase functionality.

The remainder is structured as follows: The survey transitions from a system view, over a component view to a research perspective on data networks. As part of the system view, we first provide background information of technological precursors of data networks (Section III). Subsequently, we introduce "IPFS and Friends" and provide a detailed technical overview of the next generation of data networks (Section IV and Section V). Lastly, we mention related systems and concepts (Section V-F). As part of the component view, we derive the building blocks of data networks and share insights gained from the technical overview (Section VI). Finally, we transition to a research perspective and identify research areas and open challenges (Section VII). Section II references related survey papers and Section VIII concludes this survey.

II. Related Surveys

In this section, we guide through the broad landscape of data networks and provide additional references to related survey papers. In contrast to the existing literature, we provide a comparative overview of next generation data networks, i.e., P2P data networks. We focus on storage and content sharing independent of the utilization of a blockchain.

In their 2004 survey paper, Androutsellis-Theotokis and Spinellis [8] provide a state of the art overview of P2P content distribution technologies providing a broad overview of the previous generation. Other previous works also provide closer looks at the previous generation with a closer focus on specific P2P data networks (e.g., FreeNet and Past) [7, 36] or decentralized files systems in general (e.g., Google FS and Hadoop Distributed FS) [37].

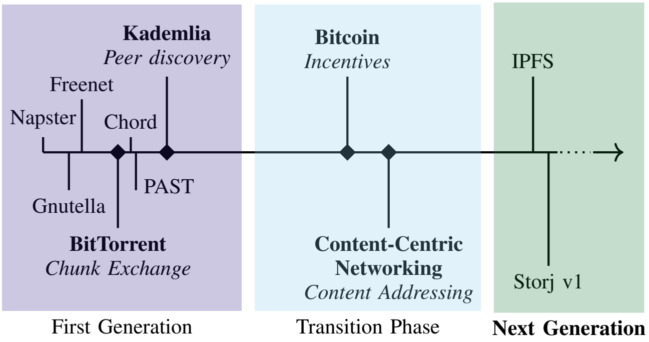


Fig. 1: Precursor technologies of the next generations of P2P data networks.

Research on next generation data networks particularly focus on the interaction with blockchains. Huang *et al.* [38] mainly cover IPFS and Swarm and Benisi *et al.* [39] with an even stronger focus on the blockchain aspects. Casino *et al.* [40] take a closer look at the immutability of decentralized storage and its consequences and possible threats. Some data networks, however, make a clear decision against the usage of blockchains, due to scalability or latency problems. In our survey paper, we therefore take a broader perspective on data networks, looking at the design decisions of data networks beyond blockchains.

A more general view on recent P2P networks is given by Naik and Keshavamurthy [41]. They describe next level P2P networks, the evolution of classic networks like BitTorrent and Chord, and discuss performance aspects under churn. It should be noted that, their definition of next level network is different from our next generation definition as they define IPFS as a "classic P2P network". We instead argue that P2P data networks evolved, incorporating, for example, explicit incentive mechanisms.

III. Technological Precursors

It has been more than two decades since the first appearance of P2P data networks. During this time the technology evolved and influenced the development of new networks. We observe that there are basically three "eras" of P2P data networks: It started with P2P file sharing and networks such as BitTorrent and Kademlia in 1999–2002, which we consider the first generation. This era is followed by a "transition phase", where new ideas such as information-centric networking and cryptocurrencies emerged. Approximately since 2014 with the invention of IPFS, we see a new generation of P2P data network gaining traction. For a better understanding of and appreciation for the influences, we provide an introduction to four important "precursor" technologies that paved the ground, namely, BitTorrent, Kademlia, information-centric networking, and blockchains. In Fig. 1, we provide a rough timeline of the data networks.

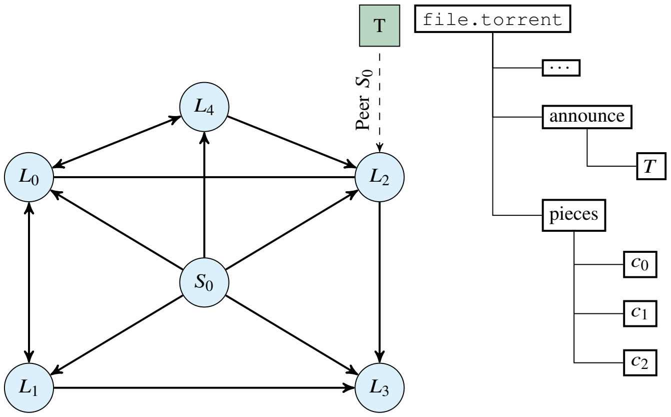


Fig. 2: Conceptional overview of BitTorrent.

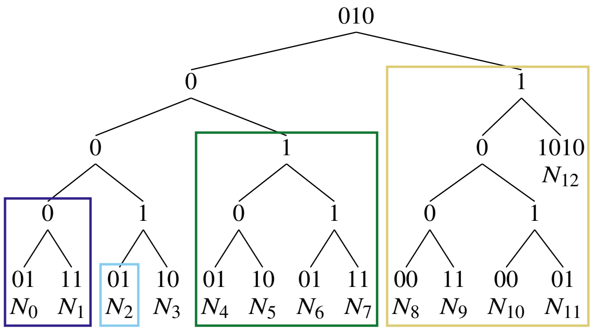


Fig. 3: Kademlia tree with 13 nodes and random ids. Highlighting the buckets for N_3

A. *BitTorrent*

The BitTorrent protocol [6] is a P2P file sharing protocol. It has an incentive structure controlling the download behavior, attempting to achieve fair resource consumption. The goal of BitTorrent is to provide a more efficient way to distribute files compared to using a single server. This is achieved by utilizing the fact that files are replicated with each download, making the file distribution self-scalable.

Files are exchanged in torrents. In general, each torrent is a P2P overlay network responsible for one file. To exchange a file with the BitTorrent protocol a `.torrent` file, containing meta-data of the file and a contact point, a tracker, is created. It is also possible to define multiple files in a `.torrent` file. The torrent file needs to be made available, e.g., on a web server, before the file can be shared. The tracker serves as a bootstrapping node for the torrent. Peers that have complete files are called seeders. Peers that still miss chunks are called leechers. Leechers request chunks and serve simultaneously as download points for already downloaded chunks.

A conceptual overview of how BitTorrent deals with files can be seen in Fig. 2. The roles and their interaction are as follows: a peer gets the `.torrent` file, contacts the tracker T listed in the `.torrent` file, gets a list of peers, connects to the peers and becomes a leecher. In the figure, the peer S_0 serves as a seed of the file and the peers L_i represent the leechers requesting the different chunks. As illustrated for the `.torrent` file, the file is split into chunks c_i. After a leecher successfully acquired all chunks, it becomes a new seed. Seed S_0 and leechers build the torrent network for the file. Other files are distributed in different torrent networks with possibly different peers.

Instead of the presented centralized trackers, there are also trackerless torrents. In a trackerless torrent, seeds are found with a distributed hash table (DHT). The client derives the key from the torrent file and the DHT returns a list of available peers for the torrent. The BitTorrent client can use a predetermined node or a node provided by the torrent file for bootstrapping the DHT.

The feature that made BitTorrent unique (and probably successful) is the explicit incentivization of peers to exchange data, which are implemented in the file sharing strategies rarest piece first and tit-for-tat. Rarest piece first describes the chunk selection of BitTorrent. It ensures a minimization of chunk overlap, making file exchange more robust against node churn. The chunks that are most uncommon in the network are preferably selected for download. Tit-for-tat describes the bandwidth resource allocation mechanism. In BitTorrent peers decide to whom they upload data based on the downloaded data from a peer. This should prevent leechers from only downloading without providing any resources to others.

BitTorrent is well researched [42, 43, 44] and has proven its test of time. Despite its age, it is still actively used by millions of people [45] for sharing files and also serves as a role model for newer peer-to-peer file distribution systems. In addition, the BitTorrent Foundation and Tron Foundation developed BitTorrent Token [46], which serves as a blockchain-based incentive layer to increase the availability and persistence of files. The new incentive structure extends tit-for-tat by bid data. The bid data determines BTT/byte rate, which the peer pays for continued service. In exchange for the payment, the peer is unchoked and eligible to receive data. The exchange of tokens is handled by a payment channel.

B. *Kademlia*

From today's perspective, Kademlia [10] is probably the most widely used DHT. As we will see later, the majority of P2P data networks builds upon Kademlia one way or another. Kademlia also influenced protocols for P2P file exchange like BitTorrent, which enables trackerless torrents by using a Kademlia-based DHT [47].

In general, Kademlia can be classified as a structured overlay network, which specifies *how* to structure and maintain the P2P network. To this end, peers are assigned an identity, which determines its position and consequently its neighbors. For the neighbor selection, an XOR metric is used. The advantage of the XOR metric is that it is symmetric and unidirectional. Depending on their XOR distance nodes are sorted into k-buckets. The buckets are arranged as a binary tree, where the shortest prefix determines the bucket. If a new node belongs to a bucket which contains k nodes including itself, the bucket gets split into smaller buckets, otherwise the new node is dropped. An exemplary Kademlia tree with 8 bit identifiers is shown in Fig. 3.

C. Information-Centric Networking

Another precursor worth mentioning is information-centric networking (ICN). Even though ICN is not a P2P data network, some of its ideas and concepts are at least similar to some data networks. Contrary to P2P data networks, ICN proposes to change the network layer. The routing and flow of packets should change from point-to-point location search to requesting content directly from the network. As an example, let us assume we wanted to retrieve some data, e.g., a website, and we know that this website is available at `example.com`. First, we request the location of the host of the site via DNS, i.e., the IP address. Afterwards, we establish a connection to retrieve the website. In ICN, we would request the data directly and would not address the host where the data is located. Any node storing the website could provide the data immediately.

One way to enable such a mechanism and to ensure data integrity is to use hash pointers (or more generically content hashes) to reference content. The content of a file is used as input of cryptographic hash function, e.g., SHA-3 [48]. The resulting digest can then be used to identify the content and the client can verify the integrity of the file locally. The cryptographic properties of the hash function, most importantly preimage and collision resistance, ensure that nobody can replace or modify the input data without changing its digest.

Jacobson *et al.* [49] proposed content-centric networking, where these content requests are interest packets. Owner(s) of the content can then directly answer the interest packet with data packets containing the content. This requires other mechanisms for flow control, routing, and security on an infrastructure level. Interest packets are broadcasted and peers sharing interest in data can share resources. There are multiple projects dealing with ICN, e.g., Named Data Networking [50] (NDN). With Ntorrent [51], Mastorakis *et al.* propose an extension of NDN to implement a BitTorrent-like mechanism in NDN. Further information on ICN can be found in [11]. Due to the content-centric nature of data networks, they could be broadly interpreted as overlay implementations of ICN.

D. Blockchain

The introduction of Bitcoin [9] in 2008 enabled new possibilities for distributed applications. Bitcoin is an ingenious, intricate combination of ideas from the areas of linked timestamping, digital cash, P2P networks, byzantine fault tolerance, and cryptography [52, 53]. One of the key innovations that Bitcoin brought forward was an open consensus algorithm that actively incentivizes peers to be compliant. Therefore, it uses the notion of coins, generated in the process, i.e., mining.

While the term blockchain typically refers to an entire system and its protocols, it also refers to a particular data structure, similar to a hash chain or hash tree. That is, a blockchain orders blocks that are linked to their predecessor with a cryptographic hash. This linked data structure ensures the integrity of the blockchain data, e.g., transactions. The blockchain's consistency is secured by a consensus algorithm, e.g., in Bitcoin the Nakamoto consensus. For more details on Bitcoin and blockchains, we refer to [53].

Since blockchains suffer from problems such as scalability, different designs have been developed to mitigate these problems. The different designs opened a new category, which is referred to as Distributed Ledger Technologies (DLT). DLTs provide distributed, byzantine fault tolerant, immutable, and ordered logs. Unfortunately, the feasibility of a purely DLT-based data network is limited, due to a series of scalability problems and limited on-chain storage capacity [54, 55]. Moreover, storing large amounts of data in a blockchain that was designed as medium of exchange and store of value, i.e., cryptocurrencies such as Bitcoin, leads to high transactions fees. However, research and development of DLTs shows the feasibility of blockchain-based data networks, e.g., Arweave (cf. Section V-E).

In general, however, cryptocurrencies allowing decentralized payments can be used in P2P data networks as an incentive structure. As we will elaborate in the following, such an incentive structure can increase the robustness and availability of data networks and therefore address weaknesses of previous generations.

IV. Interplanetary File System (IPFS)

The Interplanetary File System (IPFS) [12] is a bundle of subprotocols and a project driven by Protocol Labs. IPFS aims to improve the web's efficiency and to make the web more decentralized and resilient. IPFS uses content-based addressing, where content is not addressed via a location but via its content. The way IPFS stores and addresses data with its deduplication properties, allows efficient storage of data.

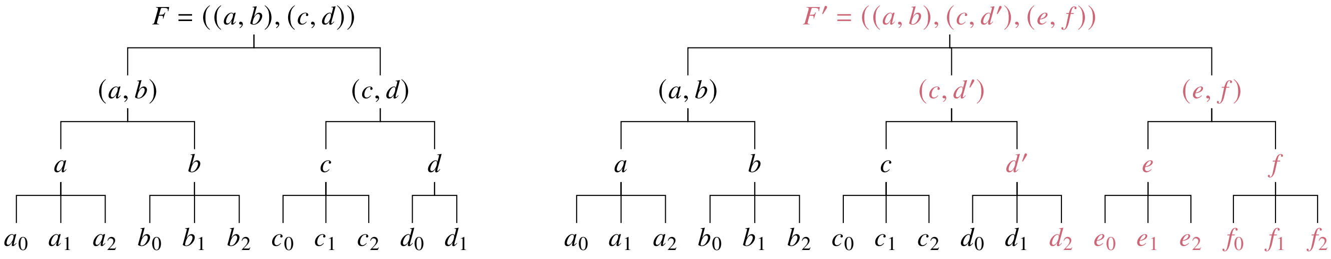
Through IPFS, it is possible to store and share files in a decentralized way, increasing censorship-resistance for its content. IPFS can be used to deploy websites building a distributed web. It is used as a storage service complementing blockchains, enabling different applications on top of IPFS.

Since IPFS uses content-based addressing, it focuses mainly on immutable data. IPFS however supports updatable addresses for content by integrating the InterPlanetary Name System (IPNS). IPNS allows to link a name (hash of a public key) with the content identifier of a file. IPNS entries are signed by the private key and can arbitrarily be (re)published to the network (default 4 h). Each peer maintains its own LRU cache of resolved entries (default 128 entries). An IPNS entry has a specific lifetime, after which it is removed from cache (default 24 h). By changing the mapping of fixed names to content identifiers, file updates can be realized. Please note, content identifiers are unique and file specific.

In addition, IPFS employs its own incentive layer, i.e., Filecoin [56], to ensure the availability of files in the network. Yet, IPFS works independently from Filecoin and vice-versa. This is a prime example of how a cryptocurrency can be integrated to incentivize peers.

A. General Functionality

IPFS uses the modular P2P networking stack *libp2p* [57]. In fact, *libp2p* came into existence from developing IPFS. In IPFS nodes are identified by a node id. The node id is the hash of their public key. For joining the network, the


$F = ((a,b),(c,d))$
(a,b)
(c,d)
a
b
c
d
a_0 a_1 a_2 b_0 b_1 b_2 c_0 c_1 c_2 d_0 d_1
$F' = ((a,b),(c,d'),(e,f))$
(a,b)
(c,d')
(e,f)
a
b
c
d'
e
f
a_0 a_1 a_2 b_0 b_1 b_2 c_0 c_1 c_2 d_0 d_1 d_2 e_0 e_1 e_2 f_0 f_1 f_2

(a) File F

(b) File F'

Fig. 4: Simplified IPFS file structure visualizing Merkle DAGs of CIDs and the concept of deduplication.

IPFS development team deployed some bootstrap nodes. By contacting these nodes, a peer can learn new peers. The peers with which a node is connected, denoted as its swarm. Peers can be found via a Kademlia-based DHT and local node discovery. The communication between connections can be encrypted. While IPFS uses Kademlia, its connections are not completely determined by Kademlia. In IPFS, a node establishes a connection to newly discovered nodes, trying to put them into buckets [32]. Idle connections are trimmed once a `HighWater` threshold is achieved (default 900) until the `LowWater` threshold is reached (default 600). A connection is considered idle, if it is not explicitly protected and existed longer than a grace period (default 20 s). Protected connections are, for example, currently actively used connections, explicitly added peers, or peers required for DHT functionality.

An object in IPFS (file, list, tree, commit) is split into chunks or blocks. Each block is identifiable by a content identifier (CID), which can be created based on a recipe from the content. From these blocks, a Merkle directed acyclic graph (DAG) is created. Merkle DAGs are similar to Merkle trees. Each node of the Merkle DAG has an identifier determined by the node's content. In contrast, however, Merkle DAGs are not required to be balanced, a node may carry a payload, and a node can have multiple parents. The root of the Merkle DAG can be used to retrieve the file. IPFS employs block deduplication: each stored block has a different CID. This facilitates file versioning, where newer file versions share a lot of blocks with older versions. In this case, only the differences between the versions need to be stored instead of two complete Merkle DAGs. The blocks have an added wrapper specifying the UNIXFS type of the block.

In order to illustrate the Merkle DAGs and the mechanism of deduplication, let us assume that our survey paper and an earlier draft of the paper are stored on IPFS. Fig. 4 shows a simplified representation of the Merkle DAGs of the two files. Each node represents a chunk and the label represents the node CID, the content hash. The DAG is created from bottom to top, since the intermediate node's CID depends on its descendants. The actual data is located in the leaves. In the final version, we assume additional information was appended to the content, which results in a different root node and additional nodes. Therefore, in our example, F is the root CID of the draft and F' the root of the finished survey.

The blocks themselves are stored on devices or providers. The DHT serves as a look-up for data providers. As in Kademlia, nodes with node ids closest to the CID store the information about the content providers. A provider can announce that it is storing specific blocks. Possession of blocks is reannounced in a configurable time frame (default 12 h).

1) Bitswap: The actual exchange of blocks is handled by the *Bitswap* Protocol. Each node has a "want", "have", and "do not want" list. The different lists contain CIDs which the node wants/has or does not want. CIDs on a do not want list are not even cached and simply dropped on receive. A node sends the CIDs on its want list to its swarm. Neighbors in possession of this block send the block and a recipe for creating the CID. The node can then verify the content by building the CID from the recipe. If no neighbor possesses a wanted CID, IPFS performs a DHT lookup. After a successful DHT lookup, a node possessing the CID is added to the swarm and send the want list.

For a peer to download, a file it needs to know the root CID. After acquiring the CID of an object's Merkle DAG root, it can put this root CID on the want list and the previously described Bitswap/DHT takes over. The root block gives information about its nodes, resulting in new CIDs which have to be requested. Subsequent CID requests are not send to all neighbors. The neighbors answering the root CID are prioritized and are grouped in a session. Since version 0.5, Bitswap sends a `WANT-HAVE` message for subsequent requests to multiple peers in the session and to one peer an optimistic `WANT-BLOCK` message. The `WANT-HAVE` message asks if the peer possesses the block and `WANT-BLOCK` messages request the block directly. If a block is received, other pending request can be canceled with a `CANCEL` message [34]. Previously, neighbors were asked for the block simultaneously, resulting in possibly receiving a block multiple times. Once all leaves of the tree are acquired, the file is locally available. Files are not uploaded to the network only possession is announced.

A typical file exchange using IPFS is illustrated in Fig. 5. Here, the author of a survey paper uses IPFS to exchange it with a reviewer. The author with the node identity N_3 provides the survey via IPFS, which splits it into a Merkle DAG (see also Fig. 4). The author shares the resulting root CID F of the DAG with the Reviewer via an out-of-band channel. The reviewer, whose node has the identity N_{11}, requests F from the network. Bitswap deals with the exchange asking neighbors for the blocks, subsequently requesting the DAG. Since nobody

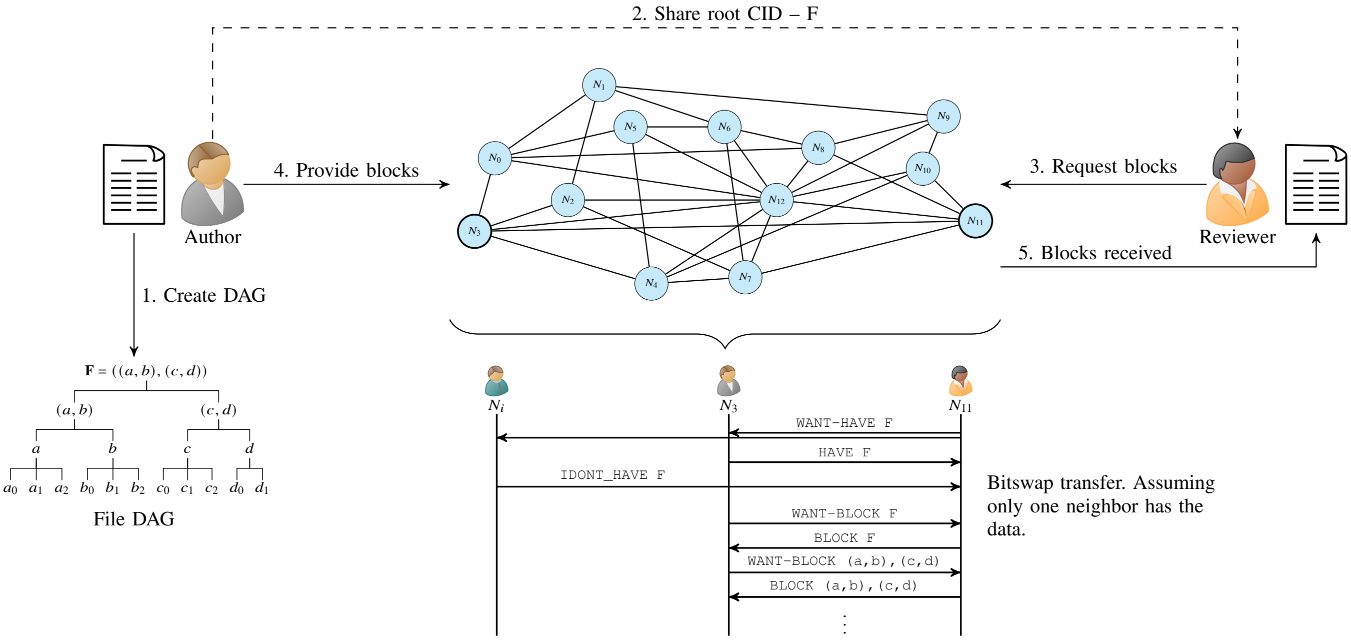


Fig. 5: Conceptual overview of IPFS.

except for the author of the survey can answer the reviewer's request, the author eventually provides the file to the reviewer using Bitswap. When the reviewer acquired all blocks, she assembles the file and can read the survey.

2) Availability: IPFS does not have any implicit mechanisms for repairing and maintaining files or ensuring redundancy and availability in the network. In our previous example, only the author N_3 and the reviewer N_{11} hold all blocks of the survey. There is no active replication due to the protocol. Files can, however, be "pinned" to prevent a node from deleting blocks locally. Otherwise, content is only cached and can be deleted via garbage collection at any point in time. Furthermore, files cannot be intentionally deleted in other nodes, deletes always happen locally only. For a file to disappear, it needs to be removed from every cache and every pinning node.

For storage guarantees Filecoin [56] exists. Filecoin employs a storage and retrieval market for storing and retrieving files. The storage market is responsible for storing data, e.g., match clients to storage miners and record their deals on the ledger, reward/punish the storage miners, and verify the continuous storage. The retrieval market is responsible for retrieving files. Retrieval miners serve data in exchange for Filecoin and only need the data at the time of serving the request. To ensure that both parties, clients and retrieval miners, can cooperate and are compensated, data retrieval is ensured with payment channels, data is sent in small pieces and compensated with micro-payments before the next piece is sent.

While the storage and retrieval market handle their tasks slightly different, the main principle is the same. There are three different orders: bid, ask, and deal. The bid order is a service request of the client that it wants to store or retrieve files. The ask order is a service offer from a storage or retrieval node announcing storage or retrieval conditions. The deal order is the statement closing the deal between bid and ask orders. Orders are stored in the Orderbook. For the storage market the Orderbook is stored on-chain, to ensure informed decision making of clients and inform the market of the trends. The orders are added in clear revealing information. The Orderbook of the retrieval market is off-chain to increase retrieval speed.

The execution of deals is maintained using a distributed ledger with Proof-of-Replication (PoRep) and Proof-of-Space-Time (PoST). The PoRep is a proof that a storage node replicated data, ensuring that the data is stored on separate physical storage. PoST proves the continuous storage over time. For more information on the different proofs, we refer to the tech report [58].

B. Features and Extensions

IPFS supports multiple transport/network protocols, or cryptographic hash functions to increase its adaptability. This is possible through the usage of multi-address and multi-hash data structures. Multi-address is a path structure for encoding addressing information. They allow a peer to announce its contact information (e.g., IPv4 and IPv6), transport protocol (e.g., TCP and UDP) and port. Multi-hash is used to provide multiple different hash functions. The digest value is prepended with the digest length, and the hash function type. Multi-hashes are used for the node id and part of the CID.

The CID in IPFS is used for identifying blocks. A CID is a cryptographic hash of its content with added meta data. The meta data includes the used hashing algorithm and its length (multi-hash), the encoding format (InterPlanetary Linked Data) and the version. More specifically, the multi-hash prepended with encoding information is InterPlanetary Linked

Data (IPLD), and IPLD prepended with version information is the actual IPFS CID.

While IPFS itself has no mechanism to ensure redundancy/availability, IPFS Cluster allows the creation and administration of an additional overlay network of nodes, separate from the IPFS main network. IPFS Cluster helps to ensure data redundancy and data allocation in a defined swarm. The cluster manages pinned data, maintains a configured amount of replicas, repinning content when necessary, and considers free storage space while selecting nodes for pinning data. IPFS Cluster needs a running IPFS node. IPFS Cluster uses *libp2p* for its networking layer.

IPFS Cluster ensures horizontal scalability of files without any incentives. It can be used by a content provider to increase availability without relying on caching in the network. Filecoin can be used to incentivize others to store files.

C. Use Cases

In the following, we provide a brief overview a few representative areas to showcase some use cases of IPFS. Please note, however, that this is by far not an exhaustive list and also not the focus of this paper. Nevertheless, it offers insights into current use cases and potential of data networks in general and IPFS in particular.

IPFS is often combined with blockchains in many areas concerning data exchange. In this case, the blockchain can fulfill various purposes. The blockchain can provide integrity, authenticity, or serve as a pointer to the data. Application areas include medical data [24, 59], tracking agricultural products [60], or in general data from the Internet of things (IoT) [18]. The blockchain can also serve as a mechanism to provide access control to IPFS data [20, 21, 61], or an audit trail for the data [29]. Data networks in general can also be used to improve storage issues of blockchains, e.g., by off-chaining transaction data [19]. Furthermore, some researchers propose new content sharing mechanism based on IPFS and blockchains [22, 23, 62]. Another proposed use case uses IPFS for making scientific papers publicly available and a blockchain to provide a review mechanism matching reviewers, reviews, and papers [63].

However, there are also use cases for IPFS that do not involve a blockchain. In compliance with its goal of decentralizing the Internet, IPFS can be used for archiving websites in an InterPlanetary Wayback [64]. Other use cases without a blockchain are in combination with ICN as content delivery network [28], combining IPFS with scale-out network attached storage for Fog/Edge computing [65], or for storing IoT data in combination with IPFS Cluster for increased availability [27].

Lastly, there is also the possibility for misuse of IPFS for malicious activities, e.g., for ransomware as a service [66] or for the coordination of botnets [25].

D. Discussion

We believe that the integration of concepts such as content addressing and deduplication are promising as they have the potential to improve retrieval times and storage overhead.

The wide support of different protocols increases the difficulty to grasp the finer details of IPFS, though. While encryption is supported in IPFS there are no additional mechanisms for increasing the privacy of its participants. The want and have list provide sensitive information about the participants. As shown by Balduf *et al.* [67] monitoring data requests in the network reveals information about its users. Therefore, IPFS could have similar privacy problems to BitTorrent. Furthermore, for good and bad it is not possible to prevent replication or enforce deletion of content once released.

IPFS became a popular research topic. In particular, IPFS itself and its performance, efficiency, and general usability are subject of a series of papers [25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35]. Please note that the referenced papers are pointers only. More details on research concerning IPFS' functionality and other data networks is discussed in a later section.

V. Related P2P Data Networks

Next to IPFS, many data networks are in development. We give an overview of five other data networks, pointing out their main concepts. A summary and comparison of BitTorrent, IPFS, and following data networks can be seen in TABLE I.

A. Swarm

Swarm [13] is a P2P distributed platform for storing and delivering content developed by the Ethereum Foundation. It provides censorship-resistance by not allowing any deletes, as well as upload and forget properties. Swarm is built for Ethereum [70] and therefore in some parts depends on and shares design aspects of Ethereum. The aim of Swarm is the provision of decentralized storage and streaming functionality for the web3 stack, a decentralized, censorship-resistant environment for sharing interactive content. The Ethereum Foundation envisions Swarm as the "hard disk of the world computer".

Similar to IPFS, Swarm uses content-based addressing. Contrary to IPFS, the content-based addressing in Swarm also decides the storage location. To ensure availability, Swarm introduces areas of responsibility. The area of responsibility are close neighbors of the node. The nodes in an area of responsibility should provide chunk redundancy. Mutability is supported through versioning, keeping each version of the file. Feeds, specially constructed and addressed chunks, and the Ethereum Name Service (ENS) are used for finding the mutated files. ENS is a standard defined in the Ethereum Improvement Proposal 137 [75]. It provides the ability to translate addresses into human-readable names. In contrast to IPNS, ENS is implemented as a smart contract on the Ethereum blockchain.

Fig. 6 provides a conceptual overview of Swarm, where we continue to use the exchange of a survey paper between author and reviewer as running example. Swarm splits a file, i.e., the survey into chunks which are arranged into a so-called Swarm hash Merkle tree. The resulting chunks are uploaded to the network. Swarm employs a Kademlia topology, where the neighbors are determined by their identifiers distance. It should be noted that additionally to the bucket connections, Swarm

TABLE I: General overview of the different data networks.

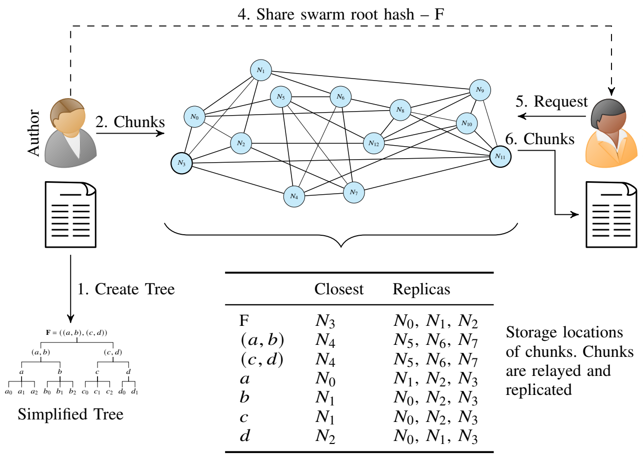
System	Main Goal and Distinct Feature	File Persistence	Token	Mutability
BitTorrent [6]	Efficient file distribution utilizing tit-for-tat to provide Pareto optimality	not guaranteed	BitTorrent-Token [46]	–
IPFS [12, 68]	Decentralized web achieving fast distribution through content addressing and wide compatibility	not guaranteed	Filecoin [56]	IPNS
Swarm [13, 69]	Decentralized storage and communication infrastructure backed by a sophisticated Ethereum-based incentive mechanism	not guaranteed	Ethereum [70]	ENS, Feeds
Hypercore [14, 71]	Simple sharing of large mutable data objects (folder synchronization) between selected peers	not guaranteed	–	yes
SAFE [15, 72]	Autonomous data and communications network using self-encryption and self-authentication for improved decentralization and privacy	public guaranteed, private deletable	Safecoin	specific
Storj [16, 73]	Decentralized cloud storage that protects the data from Byzantine nodes with erasure codes and a reputation system	determined lifetime, deletable on request	Centralized Payments	yes
Arweave [17, 74]	Permanent storage in a blockchain-like structure including content filtering	blockweave	Arweave token	–

relies on a nearest neighbor set, which are the remaining nodes of the neighborhood. A neighborhood is basically the lowest amount of buckets that contain at least 3 other peers. This nearest neighbor set is responsible for replication and is not necessarily symmetric. The uploaded chunks are relayed, stored, and replicated at the location closest to their address. To retrieve the survey the swarm root hash is necessary. The network relays requests according to the content address.

To ensure compliant node behavior, Swarm provides an incentive layer. The incentive structure is based on SWAP, SWEAR, and SWINDLE. The SWarm Accounting Protocol (SWAP) handles the balancing of data exchange between nodes. Each node maintains local accounting information. If a peer sends a retrieval request, it is charged and the serving node is rewarded. The price for chunks is negotiable between the peers. Requests are served until certain imbalance thresholds. If the first threshold is reached, the node expects compensation for further service. If the second threshold is reached, due to not sending compensation, the node is disconnected. The balance can be settled with cheques, which can be interpreted as a simple one-way payment channel. SWarm Enforcement And Registration (SWEAR) and Secured With INsurance Deposit Litigation and Escrow (SWINDLE) shall ensure persistence of content. Furthermore, Swarm's incentive structure has postage stamps, which provide a mechanism against junk uploads and also a lottery mechanism to incentivize the continued storage of chunks.

Postage stamps can be acquired in batches from a smart contract. The postage stamps are attached to an uploaded chunk and signed by the owner of the stamp. This serves as a proof of payment for uploading chunks. Stamp usage can only be monitored by relaying or storing nodes. This allows reusage/overusage of stamps. To reduce the risk of overusing stamps, the stamps are only for certain prefix collisions, limiting stamps to chunks in certain storage areas.

The postage stamps are used in a lottery. The lottery provides value to chunks preventing early deletes of chunks. Through the lottery, storage nodes can gain part of the initial costs for the stamp. In the lottery an address area is chosen. Nodes in the proximity of the area can apply for reward. By applying, nodes testify possession of chunks in the area. The nodes define a price for storing a chunk. After proving possession of the chunks, the node with the cheapest prize gets the reward.


4. Share swarm root hash – F
Author
2. Chunks
N_0
N_1
N_2
N_3
N_4
N_5
N_6
N_7
N_8
N_9
N_{10}
N_{11}
N_{12}
5. Request
6. Chunks
Reviewer
1. Create Tree
$\mathbf{F} = ((a, b), (c, d))$
(a, b)
(c, d)
a
b
c
d
a_0 a_1 a_2 b_0 b_1 b_2 c_0 c_1 c_2 d_0 d_1
Simplified Tree

	Closest	Replicas
F	N_3	N_0, N_1, N_2
(a, b)	N_4	N_5, N_6, N_7
(c, d)	N_4	N_5, N_6, N_7
a	N_0	N_1, N_2, N_3
b	N_1	N_0, N_2, N_3
c	N_1	N_0, N_2, N_3
d	N_2	N_0, N_1, N_3

Storage locations of chunks. Chunks are relayed and replicated

Fig. 6: Conceptual overview of Swarm.

Discussion: Swarm provides sophisticated incentive concepts. Settling unbalanced retrieval with cheques provides a faster and cheaper way to settle discrepancies than relying on blockchain transactions. The postage stamps with the lottery give an additional incentive for storing chunks. Additionally, while it does cost to upload content, nodes can earn the cost by actively serving chunks to participants. However, postage stamps link a user to uploaded content. While Swarm provides a certain degree of sender anonymity, the upload pseudonymity might limit available content.

Considering the determined storage locations by the Distributed Immutable Store for Chunks (DISC), the network might face storage problems. Feeds, which can provide user-defined space in the network, in the form ofrecovery feeds and pinning, might be able to mitigate these disadvantages.

Overall, Swarm clearly depends on the Ethereum ecosystem. While it is advantageous for the incentive structure, since Ethereum is actively developed and has a broad user base, it also requires users to depend on Ethereum. While having this potentially large user base, research of use cases or research investigating Swarm's mechanism is rare. The connection of Swarm and Ethereum could be one reason for the lack of research, since Swarm seems less complete than IPFS and

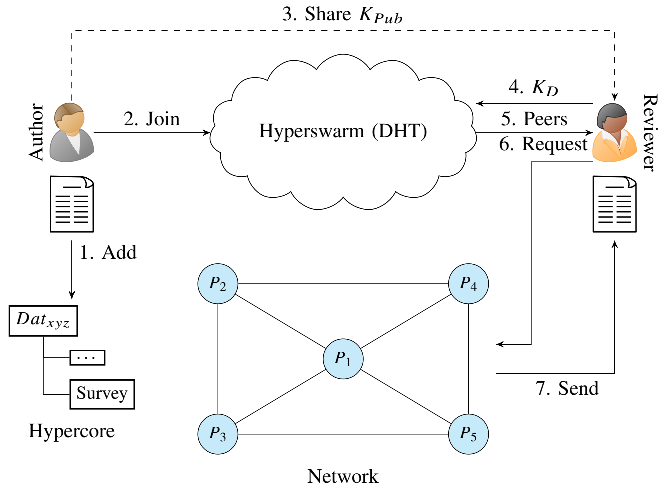


Fig. 7: Conceptual overview of Hypercore.

Ethereum itself still maintains many research opportunities.

B. Hypercore Protocol/Dat

The Hypercore Protocol [14, 76] (formerly Dat Protocol) supports incremental versioning of the content and meta data similar to Git. The Hypercore Protocol consists of multiple sub-components. While strictly speaking Hypercore is one of the sub-components, for simplicity we use the term to reference the Hypercore Protocol in general. In Hypercore, data is stored in a directory like structure and similar to BitTorrent each directory is dealt with its own network. The protocol supports different storage modes, where each node can decide which data of a directory and which versions of the data it wants to store. Furthermore, the protocol supports subscription to live changes of all/any files in a directory. All communication in the protocol is encrypted. In order to find and read the data, it is necessary to know a specific read key.

The protocol is designed to share large amounts of mutable data. The motivation for creating the protocol was to prevent link rot and content drift of scientific literature. The protocol allows sharing of only part of the data with random access.

Hypercore can be understood as sharing a folder. Files in a folder can be modified, added, and deleted. This also includes and allows mutable files.

A conceptual overview of Hypercore can be seen in Fig. 7. For peer discovery, Hypercore uses Hyperswarm, a Kademlia-based DHT. If the author wants to share the survey using the Hypercore Protocol, the author needs to create a Hypercore and add the survey. To be able to be found by Hyperswarm, it is necessary to join the Hyperswarm overlay network. By sharing the public key K_{Pub}, the reviewer can calculate the discovery key K_D and after finding peers and joining the data network decrypt the messages. Once the other overlay network is joined the unstructured network of volunteers can share the data and the survey can be retrieved.

Discussion: Hypercore allows sharing of data by exchanging a public key. It is possible to acquire a specific version and only specific regions of the data. This makes it simple, especially for large datasets, and allows mutable data. The protocol natively concentrates on sharing collection of files, which broadens the usability of the protocol.

Due to the encryption and a discovery key, the protocol ensures confidentiality. A public key allows the calculation of the discovery key, but it is not possible to reverse the public key. This prevents others from reading the data. A downside of Hypercore is the lack of additional authentication mechanisms beyond the public key, which prevents additional fine-grained access control. Furthermore, it still leaks meta data since the discovery key is only a pseudonym.

Hypercore has no incentive structure for replicating data and the data persistence relies on its participants. Research utilizing or analyzing Hypercore/Dat is rare. While the protocol seems well developed and usable, research seems to focus on IPFS, instead.

C. Secure Access For Everyone (SAFE)

The Secure Access For Everyone (SAFE) network [15, 77] is designed to be a fully autonomous decentralized data and communication network. Even authentication follows a self-authentication [78] mechanism, which does not rely on any centralized component. The main goal of SAFE is to provide a network which everyone can join and use to store, view, and publish data without leaving trace of their activity on the machine. This would allow participants to publish content with low risks of persecution.

SAFE supports three different data types: Map, Sequence, and Blob. The data can be further divided into public and private data. Map and sequence are Conflict-free Replicated Data Types, which is important in case of mutable data to ensure consistency. The Blob is for immutable data. All data in the SAFE network is encrypted, even public data. The used encryption algorithm is self-encryption [79], which uses the file itself to encrypt the file. That is, a file is split into at least three fixed size chunks. Each chunk is hashed and encrypted with the hash of the previous chunk, i.e., $n - 1$ where n is the current chunk. Afterwards, the encrypted chunk gets obfuscated with the concatenated hashes of the original chunks. In case of SAFE, the obfuscated chunks are stored in the network. For decryption, a data map is created during the encryption process. The data map contains information about the file and maps the hash of obfuscated chunks to the hash of the real chunks. For public data, the decryption keys are provided by the network. While private data can be deleted, public data should be permanent. Therefore mutable data can only be private. A Name Resolution System allows human-readable addresses for retrieving data.

The network itself is organized by XOR addresses according to a Kademlia-based DHT. Furthermore, the network is split into sections. When a new vault wants to join the network, the new vault needs to prove, that it can provide the required resources and is then randomly assigned a XOR address and therefore to a section. The sections are maintained dynamically. According to the amount of vaults in the network, sections are split and the vaults are reassigned to new sections. Sections that grow too small are prioritized by getting new nodes or can request relocation of nodes to balance section

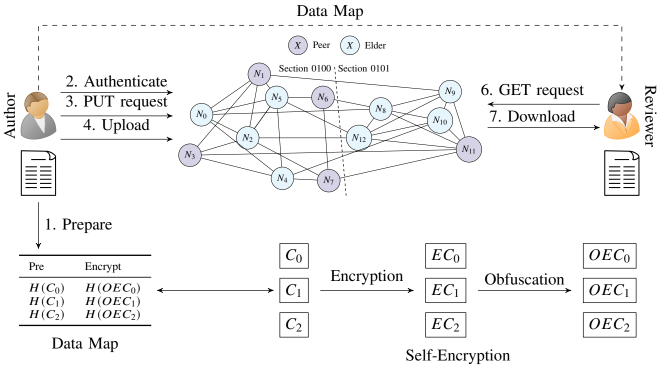


Fig. 8: Conceptual overview of SAFE.

size. Changing the section increases the vault's node age. Node age is a measurement of trust, can be lost and must then be re-earned. Only a certain amount of nodes can make decisions in a section, the elders. Elders are the oldest nodes in the section. The elders can vote on accepting or denying events in a section and a certain quorum of elders has to approve and group sign the decision for it to become valid. Events in a network section are, for example, joining/leaving of a node or storing a chunk. The authenticity of the elders is ensured by a SectionProofChain, which holds the elders' group signatures and is a sequence of public keys proving the validity of a section. The sequence is updated and signed everytime the group of elders changes.

A conceptual overview of the SAFE network can be seen in Fig. 8. Considering our running example, the survey is split into chunks, self-encrypted, and used to generate a data map. After the self-authentication process, a *PUT* request is sent to the network. When the elders in the section responsible for storing chunks agree, the data is stored. For downloading the file, the data map is required. The data map is used for *GET* requests to acquire the obfuscated encrypted chunks. For downloading public data, authentication is not necessary. After acquiring the chunks the file can be recreated with the help of the data map.

In the SAFE network, storing data is charged with the network's own currency, i.e., Safecoin. The Safecoin balance of the clients is monitored by client managers and approved/rejected with the help of SAFE's consensus mechanisms. Nodes can earn Safecoin by farming, i.e., providing content to requesters.

Discussion: The self-authentication, self-encryption, and the network organization give the user a high degree of control over their data. The absence of central components reduce single points of failure. Furthermore, privacy and to a certain degree anonymity are key features of the SAFE network. The network requires authentication for storing data only. Retrieving data is mediated via a client-selected proxy, which provides pseudonymous communication. Safecoin is intended to provide an incentive layer which ensures the availability and reliability of the network.

Paul *et al.* [80] provide a first security analysis of SAFE in 2014, concerning confidentiality, integrity and availability as well as possible attacks. In 2015, Jacob *et al.* [81] analyzed the security of the network with respect to authenticity, integrity, confidentiality, availability, and anonymity. The authors explained how the self-authentication and the decentralized nature could be potentially exploited to reveal personal data of single entities.

SAFE is in development since 2006 and considers recent research and developments, but remains (at the time of writing) in its alpha phase. We feel that SAFE has a potential to establish the topic of anonymity as a distinct feature when compared to the other data networks.

D. *Storj*

Storj [16] is a P2P storage network. In the following, we refer to version 3.0. It concentrates on high durability of data, low latency, and high security and privacy for stored data. End-to-end encryption for communication, file locations, and files is supported. For the high durability of files or in other words better availability of files in the network, Storj uses erasure codes. Furthermore, low bandwidth consumption is also one main design goal. The protocol assumes object size of $4\,MB$ or more, while lower object sizes are supported the storage process could be less efficient. In Storj, decentralization is interpreted as no single operator is solely responsible for the operation of the system. In a decentralized system, trust and Byzantine failure assumptions are important. Storj assumes no altruistic, always good behaving nodes, a majority of rational nodes, behaving only malicious when they profit, and a minority of Byzantine malicious nodes.

Storj aims to be a decentralized cloud storage. Storj Labs Inc. wants to provide an alternative to centralized storage providers. For this purpose, Storj provides compatibility with Amazon S3 application programming interface to increase the general acceptance and ease the migration for new user. Since Storj provides cloud storage, user are allowed to store and retrieve data as well as delete, move, and copy data.

The Storj network consists of three node types, satellite, storage, and uplink nodes. The satellite nodes administrate the storage process and maintenance of files. The encryption of meta data and even file paths adds an additional protection of meta data. Uplink nodes are end users, who want to store and access files. Storage nodes store the data. Storage and uplink nodes choose with which Satellite nodes to cooperate. This results in a network similar to BitTorrent where satellites become central parts.

A conceptual overview of Storj can be seen Fig. 9. To upload the survey paper, the author needs to split it into segments, which are then encrypted. The author requests the satellite to store a segment. The satellite checks capacity of the storage nodes and returns a lists of adequate storage candidates. The segment is then split into stripes, which are erasure encoded and arranged into pieces. The pieces are then uploaded to the storage nodes in parallel.

For the erasure encoding, Storj uses Reed-Solomon erasure codes [82]. For erasure codes the data is encoded in a (k,n) erasure code. This means, that an object is encoded into n pieces, in such a way that only k pieces are necessary to recreate the object. Storj chooses four values for each object:

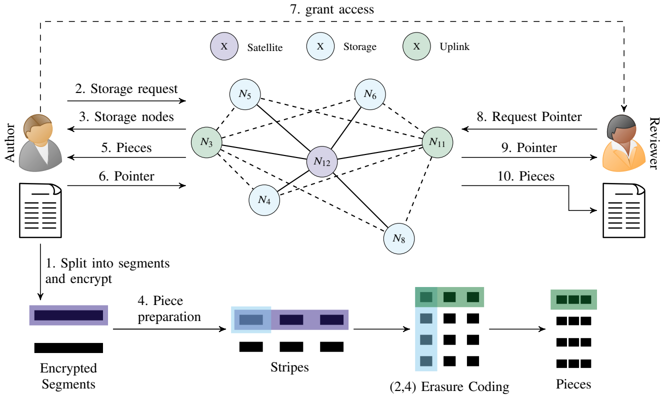


Fig. 9: Conceptual overview of Storj using (2,4) erasure coding.

k, m, o, and n. k represents the minimum of required pieces to reconstruct the data, m is a buffer for repair, o is a buffer for churn and n is the total number of pieces. Erasure codes provide a higher redundancy with less overhead compared to storing the pieces multiple times. Furthermore, since only k pieces are required to retrieve the file, the latency till the file is available can be reduced.

After the upload, a pointer containing meta data of the segment (e.g., hash of pieces, storage location, erasure encoding scheme) is returned to the satellite. This is repeated for each segment and the last segment contains additional meta data about the survey. For downloading the survey paper, the pointer for the segments are requested. The pieces are requested in parallel from the storage nodes. Once enough pieces to assemble the segments are gathered, the survey can be read.

To ensure the cooperation of the rational nodes, Storj provides an incentive system. The incentive system rewards storage nodes for storing and providing content. Nodes are monitored with audits and evaluated via a reputation system. A goal of Storj is low latency, which lead to avoiding a blockchain dependent incentive mechanism.

Discussion: Storj employs some concepts that are unique when compared to other P2P data networks. In particular, the Amazon S3 compatibility might promote Storj as decentralized storage system. The erasure codes add overhead to storing files, but during a file retrieval only the necessary amount of pieces need to be downloaded. The decentralization of storage, through the erasure codes, with adequate storage node selection and the help of a reputation system increases the protection against data breaches.

The satellite nodes are important parts of the network and partition the network, since files available at one satellite are not available at another satellite. This promotes centralization in form of the satellite. While satellites cannot share the meta data with possible third parties due to the encryption, it is still possible to leak access patterns.

While Storj is deployed and can indeed be used, applications and research on the topic is rather rare. De Figueiredo *et al.* [83] analyzed the Storj network and identified the satellite nodes as possible vectors for Denial-of-Service attacks. They modified the implementation of storage node's connection handling and successfully took down a satellite node in the test environment, rendering payment and file retrieval impossible for some time. However, the production system should be resistant to such an attack. Another study also showed a different attack on data networks. Zhang *et al.* [84] showed, in Storj v2.0, the possibility to upload unencrypted data to storage nodes, which can be used to frame owner's of storage nodes. Nonetheless, Storj's provided privacy guarantees, resilience, acquirable meta data or the possibility to deploy the different nodes by everyone could provide valuable insights for cloud storage.

E. *Arweave*

The Arweave protocol [17] utilizes a blockchain-like structure, called blockweave, to provide a mechanism for permanent on-chain data storage as well as payment for storage. In the blockweave, a block points to the direct preceding block and a recall block, which is deterministically chosen based on the information of the previous block. While the weave is immutable and provides censorship-resistance of its data, every node can decide to refuse accepting content. Refusing content by a sufficiently large amount of nodes prevents inclusion of unwanted content.

Arweave utilizes Wildfire, a protocol similar to BitTorrent's tit-for-tat to rank peers. Through Wildfire each node maintains a list of peers and scores and subsequently ranks the peers based on their responsiveness, e.g., answering requests or send transactions. The score is basically determined by received bytes per second averaged over recent requests. High ranking and therefore best-performing peers receive messages first in parallel, sequentially followed by the rest. Connections to low ranking peers are periodically pruned. This incentivizes nodes to be highly responsive themselves to receive messages as fast as possible. Furthermore, it should optimize resource utilization of the nodes and reduce communication time.

At the heart, Arweave is a blockchain-based network. While Wildfire introduces a ranking that favors certain connections, it remains an unstructured P2P network. A conceptual overview of Arweave and how to archive/retrieve a file can be found in Fig. 10. To archive the survey paper in Arweave, it is necessary to send a transaction to the network. The peers confirm the transaction by including it in a block. If someone wants to read the survey the network is asked. If a peer stores the block containing the survey, it can be returned and the survey can be read.

Arweave's goal is to provide eternal permanent storage of data, preserving and time-stamping information in an immutable way. The data is stored on-chain on the blockweave, therefore, immutable and only removable through forking the weave. The blockweave provides decentralized storage for the permaweb.

Storage and maintenance of the blockweave and its data is ensured through Arweave's cryptocurrency: Arweave tokens. The tokens are used for rewarding miners and payment for sending transactions.

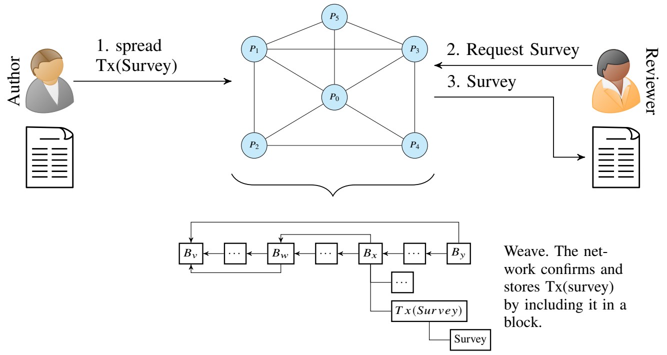


Fig. 10: Conceptual overview of Arweave.

Discussion: The Arweave protocol provides on-chain storage on a blockchain-like structure. This gives the storage similar advantages and disadvantages of a blockchain. Arweave provides time-stamping, transparency, incentives, and immutable storage. The data is stored through transactions providing pseudonymous authors of data.

One of the biggest problems of blockchains is the scalability. Arweave tries to reduce these problems by utilizing blockshadows, a mechanism similar to compact blocks, explained in Bitcoin Improvement Proposal 152 [85], and Wildfire for fast block propagation reducing fork probability. Furthermore, the usage of Block Hash List and Wallet List should reduce the initial cost of participation. With version 2.0 Arweave introduced a hard fork to improve scalability, decoupling data from transactions. Instead of including the data in the transaction, a Merkle root of the data is included. This improves transaction propagation speed, since the data is no longer necessary to forward the transaction.

Due to the pseudo-random recall block, nodes are incentivized to store many blocks to maximize their mining reward. This increases the replication of data. However, not every node necessarily stores every block or content, every node decides for itself based on content filter which data it stores. Requesting content might become complicated, since nodes are requested opportunistically in hope they store the content.

Research about Arweave directly is at most sparse. However, this can be explained by the broad range of emerging blockchain-based protocols and research about blockchains can be at least partly applied to Arweave as well.

F. Honorable Mentions and Related Concepts

Next to our detailed overview of select P2P data networks, we provide additional literature on other systems and concepts concerning the current generation of P2P data networks. In particular, there are some paper concepts providing different and sophisticated ideas for P2P content sharing.

Sia [86] aims to be a decentralized cloud storage platform. A file is split into chunks, which are encrypted and then stored via erasure coding on multiple storage nodes. The location of chunks is stored as metadata. Sia uses a blockchain to incentivize storage and retrieval of data. The conditions for and duration of storing the data is fixed in storage contracts. The data owner is responsible for file health.

The Open Storage Network (OSN) is a distributed network for transferring and sharing research data. It is comparable to a distributed cloud service dedicated to large amount of research data. Data is stored in centrally monitored and maintained pods. These OSN pods are specially configured server racks and require a high bandwidth Internet connection. Institutions that want to contribute to the network can house pods. As a consequence, researchers can store and share their research data in the OSN network. The connectivity of the OSN pods shall ensure fast access to the data. Data can be shared with selected participants or via open access. The central management and strict conditions for pods differentiates the OSN from the rest of the presented data networks, where decentralization and arbitrary participation is a key feature.

Fukumitsu *et al.* [87] propose a peer-to-peer-type storage system, where even meta-data, necessary for reconstructing the stored files, is stored in the network and can be retrieved with an ID, a password, and a timestamp. The authors assume an unstructured P2P network where each node can offer different services. Nodes broadcast regularly necessary information about themselves, e.g., offered services and its IP address. An important component of the scheme are storage node lists stored on a blockchain. The storage node list is a randomly ordered list of selected nodes offering storage services. Data is stored in parts and the storage process is split into two phases: storing user data and storing data necessary for reconstructing user data. User data is encrypted, divided into parts and the parts are stored on nodes selected from the currently available storage nodes. The parts can be requested using restore keys. For reconstructing user data the decryption key and pairs of storage node and restore keys are necessary. Therefore, data is replicated on other nodes. A user creates an ID, password pair, and selects a storage list. The data is encrypted with the hash of ID, password and storage list. Storage nodes are chosen deterministically from the storage list. The restore key for the parts is the hash of the storage list and the hash of a piece index, the ID and password. This scheme allows fetching data without storing information on the user device.

Jia *et al.* [88], propose *OblivP2P* a mechanism implementing ideas from oblivious RAM to hide data access patterns. While the authors mention that their mechanism is applicable to other peer-to-peer systems, they focus on a BitTorrent like system with a tracker.

Qian *et al.* [89] propose Garlic Cast, a mechanism for improving anonymity in an overlay network. Peers do not request and search content directly. Instead, a peer searches for proxies and the proxies exchange and request the content. Messages between a peer and its proxy are exchanged via a security-enhanced information dispersal algorithm (IDA). An IDA is a form of erasure coding where k of n pieces are sufficient to reconstruct the object. The security-enhanced IDA first encrypts a message, splits the message and key into n fragments with a k-threshold IDA, and sends cloves, messages containing a key and message fragment. Proxies are discovered via random walks: Cloves are send to its neighbors, requesting peers to be a proxy with a random clove sequence number, each neighbor randomly forwards the clove and maintains the state of successor and predecessor, A peer with two cloves

TABLE II: Summary of the building blocks.

Category	BitTorrent	IPFS/Filecoin	Swarm	Hypercore	SAFE	Storj	Arweave
Network							
Topology	Unstructured	Hybrid	Kademlia	Unstructured	Kademlia	Kademlia	Unstructured
File Handling							
File Look-up	DHT, Central	DHT, Opportunistic	DHT	DHT	DHT	Central	Opportunistic
Storage	File	Blocks	Chunks	Files	Chunks	Segments	Files
Storage Location	Random	Random	Addressed	Random	Addressed	Random	
File Replication	Passive	Passive, Caching	Active/Passive, Caching	Passive	Active, Caching	–	Passive
Information Security							
Confidentiality	–	–	Manifests	Public-key	Self-authentication	Satellite nodes	–
Integrity	Meta-data file	Content-addressing	Content-addressing	Meta-data file	Content-addressing, self-encryption	Satellite nodes	Blockweave
Availability	Replication, Incentives	Replication, Incentives	Replication, Erasure Codes, Incentives	Replication	Replication, Incentives	Erasure Codes, Incentives	Replication, Incentives
Incentivization							
Upload	Free	Free	Charge	Free	Charge	Free	Charge
Reward (Storing)	–	For Time	For/Over Time	–	–	For Time	Over Time
Punish (Storer)	–	Misbehavior	Misbehavior	–	–	Misbehavior	–
Chunk/File Trade	Monitor	Monitor	Monitor	–	–	Monitor	Monitor
Retrieval Only	Charge (optional)	Charge (optional)	Charge imbalance	–	Reward	Charge	–

with the same sequence number can recover the request, and if it volunteers to be a peer returns a reply to the requester.

Other paper concepts utilize a blockchain for access control and to store data locations instead of a supplement as an incentive mechanism, e.g. Blockstack [90], which maintains meta-data on the blockchain and relies on external data stores for actual storage of data. There are also concepts using distributed ledger technologies for access control e.g. Calypso [91], which uses a skipchain-based identity and access management allowing auditable data sharing. However, these systems and systems concentrating only on selling data via the blockchain are outside of the scope of this survey.

VI. Discussion of Building Blocks

After gaining an initial understanding of each system, we take a closer look at all systems, identifying similarities and distinct differences. In this discussion, we also include BitTorrent as prominent example from a previous generation of data networks. By comparing these systems and reviewing literature on the topic, we identify building blocks and open challenges in P2P data networks. In particular, we identified the areas, network architectures, file handling, information security, and incentivization as most relevant technical aspects. We take these building blocks and derive a taxonomy. In TABLE II, we provide a summary of building blocks.

A. Network Architecture

Each of the considered data network builds an overlay network to communicate with other peers. An overlay network is a logical network of nodes on top of the real network. While many ways exist to organize an overlay network [3, 5], we clearly see a dominance of Kademlia [10]. Each network uses a Kademlia-based DHT one way or another; this can result in two different overlay networks, one for peer discovery and one for the data exchange.

Despite using Kademlia, the networks are organized differently upon closer inspection. IPFS, Swarm, and SAFE use the DHT also to structure the network. SAFE, however, separates the network additionally in sections, where each section organizes itself with so-called elders. Swarm creates a Kademlia topology, where the identity directly decides the neighbors. SAFE and Swarm can therefore be classified as structured overlay networks. While IPFS also uses a DHT, a peer connects to every peer it encounters, which leads to an unstructured network. If the number of connections exceeds a certain limit, connections are pruned with the exception of actively used or DHT required connections [32]. This structures the IPFS network to some degree. Storj used a DHT for peer discovery in the past. Storage nodes decide how much resources are provided to a satellite and with which satellite it cooperates. Therefore, Storj replaced the initial peer discovery with a direct communication of satellite and storage nodes. Storage and satellite nodes maintain their own peer list. Furthermore, cooperation between satellites and storage nodes, is controlled with a reputation system. In BitTorrent and Hypercore, the DHT is used for peer discovery only. Once peers are discovered, a separate unstructured overlay network is responsible for data exchange. In BitTorrent, the connection between the peers are decided based on tit-for-tat.

Arweave is an exception as it does not use a DHT at all. Arweave uses a gossip protocol similar to Bitcoin, where peers announce their neighbors and known addresses. Concerning network organization, Arweave has no strict structure for its neighbor selection, although it uses Wildfire, a tit-for-tat based mechanism to rank peers and drop connections from

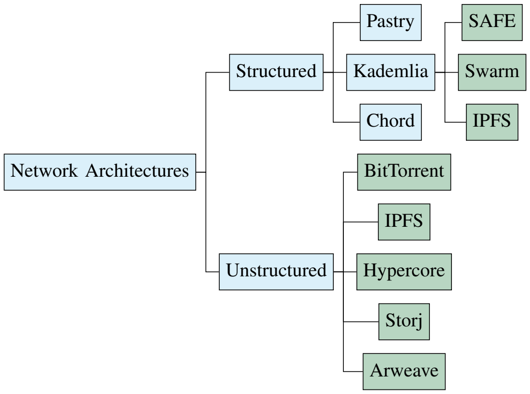


Fig. 11: Overview of the different network architectures.

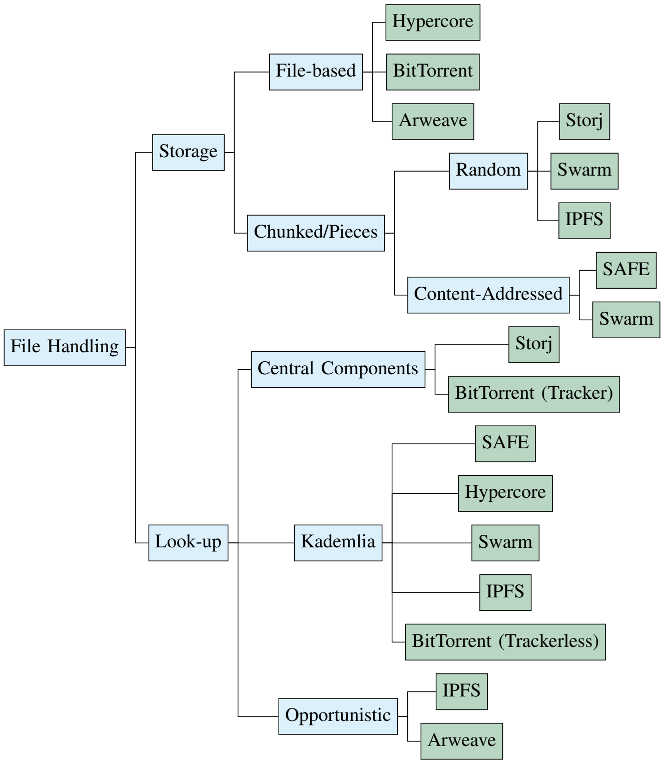


Fig. 12: Overview of file storage and look-up mechanisms.

unresponsive/unpopular peers.

An overview of the presented categorization with respect to the network architecture is provided in Fig. 11.

B. File Handling and File Size

The file handling is another core component of a data network and clearly more diverse than the network organization. We provide an overview of our taxonomy in Fig. 12, which we divide in storage and file look-up mechanisms.

A common pattern with respect to storage is that in each data network, immutable files or at least immutable data blobs are preferred. Mutability and intentional deletion of files is rather a feature than the default.

Due to the respective protocol, the files are split into pieces either during the exchange (BitTorrent, Hypercore) or the file is stored in pieces located on potentially different devices. Splitting files into pieces increases the storage overhead due to additional meta data. At the same time, though, it improves the retrieval process in case of large files. Arweave does not split files into pieces. Instead, it uses transactions to store files, which become part of a block in the blockweave.

While chunking is in general a common feature, the storage is irregular. BitTorrent and Hypercore concentrate more on exchanging data than using the network to store data on their behalf. This results in a high probability of all chunks being present on one device. The storage is rather file-based since the aim is the possession of all chunks to possess the file.

IPFS and Swarm split the files into pieces and build a Merkle Tree/DAG. The root is then sufficient to retrieve the file. Each piece can be addressed and retrieved by itself and individually stored on separate nodes. In IPFS, the location of chunks is “random” in the sense that each node can determine by itself, if it stores a certain chunk. In Swarm a chunk’s storage location is tied to its address. However, similar to IPFS other nodes can also decide to additionally store chunks.

SAFE splits the chunks into pieces and encrypts the chunks with each other. Similar to Swarm a chunk is content addressed and the content decides the storage location.

Storj splits the files in erasure encoded pieces, reducing the required trust in single nodes. The storage location of the pieces is decided randomly and distributed on the available storage nodes, cooperating with the responsible satellite node.

The chunking of files also influences the look-up process. The request is either referencing a chunk/file directly or a chunk pointing to other chunks. The chunks are in general retrieved from neighbors. The request to neighbors can be directed or random via a broadcast. In case of Arweave and IPFS, the file look-up can be considered opportunistic as peers are queried without knowledge about the peers’ possession of the chunks/file. In Storj a central component is available to send direct requests. In the other data networks, however, peers utilize a DHT for the look-up. In IPFS the DHT is used as a backup look-up, if the opportunistic request fails. Since in BitTorrent and Hypercore the exchange overlay network deals with specific data, we have to differentiate here: a neighbor is expected to possess at least part of a file. Therefore, the peer discovery can be considered as a directed request. To this end, BitTorrent uses either a central component (i.e., a tracker) or a DHT (i.e., trackerless). Hypercore uses a DHT.

Due to the increasing amount of collected data, it is also appropriate to consider limits of the data networks concerning file sizes. In BitTorrent, IPFS, and Hypercore data is stored on the data source’s node first and shared later, while Swarm, SAFE, and Storj store chunks in the network directly. This limits possible data sizes in the first case to a node’s owned storage capacities and in the latter case to the network’s and peer’s willingness to provide storage. Furthermore, Storj assumes an object size of 4 MB or more and files are erasure encoded. Swarm chunks are split into 4 kB and are distributed in the network based on their hash, which could make it difficult to retrieve large files. In Arweave, the data is stored on-chain similarly limiting size to the network. Arweave had a file size limit of 3 MB. However, it was removed with the upgrade to version 2.1.

Hypercore is designed for large data sets and partial data sharing. Partial data sharing can improve possible file sizes. IPFS's Merkle DAG allows retrieval of partial data as well. In contrast, BitTorrent's chunking, SAFE's self-encryption, and Storj's erasure codes prevent partial data sharing.

For the reasons outlined before, we believe that most data networks are unsuitable for large, single datasets in the range of Petabytes and are rather designed for data in the range of Megabytes and Gigabytes. However, future performance measurements are necessary to confirm or deny our reasoning.

C. Information Security

Confidentiality, integrity, and availability (CIA) are important aspects of information security. These aspects provide additional challenges and gain additional importance in the distributed setting of data networks. In a distributed system where data is potentially stored on different unsupervised devices, it is hard to protect the data or control access to data. Since the data comes from many untrusted devices, the integrity needs to be guaranteed. We can generally expect improved availability, e.g., due to the redundant storage and distribution of data. However, considering availability as long term file persistence remains a challenge. Any node could delete content and arbitrary join or leave the network, which results in files becoming unavailable.

1) Confidentiality: To keep content and meta-data of data confidential from other participants is difficult in a distributed environment. Even nodes storing data are possible information leaks. Encryption is the main instrument to protect the data in distributed systems. The encryption prevents other parties from reading the content of files despite fetching or storing the data. An additional protection against storage nodes is chunking of files. By chunking the file and ideally distributing the chunks on different nodes a storage node is unable to identify content. Swarm, SAFE, and Storj distribute the chunks during the storage process. In the other data networks, the distribution is less prominent, or in case of Arweave not present at all.

Another aspect which protects the content of data is access control. Access control in the presented data networks is mostly realized through distributing decryption keys. The exchange of the decryption key is mainly handled by the concerned parties directly outside of the data network. BitTorrent, IPFS, and Arweave employ no additional access control. However, some data networks also provide additional mechanisms. In Storj, satellite nodes verify and authorize access requests. Data access is additionally restricted by satellites, where another satellite cannot grant access to data submitted to another satellite. SAFE uses self-authentication to authenticate access to private data. Swarm provides access control through so-called manifests. In Hypercore, it is necessary to know the public key of the directory for discovering peers and decrypting the communication. This provides an additional distinction between write and read access.

2) Integrity: For the integrity of data, it is possible to rely on and trust the data provider. However, in a distributed system it is hard to trust all peers. The presented data networks utilize hash functions to ensure integrity. The hash value has to be known in advance and therefore might require out-of-band communication. Given a hash and the algorithm used for the hash, content can be verified by regenerating the hash and comparing it with a given hash.

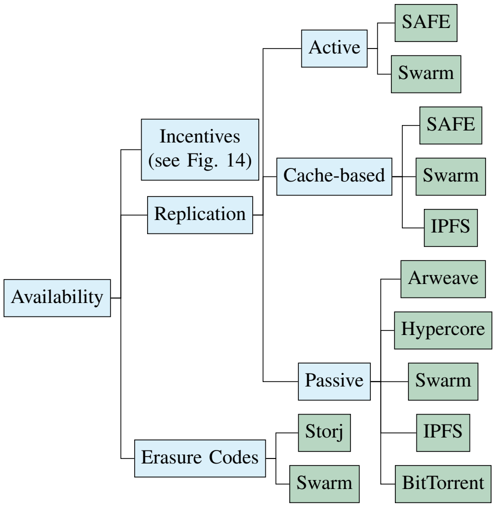


Fig. 13: Overview of availability mechanisms.

The usage of hash functions is different. In BitTorrent and Hypercore, the hash is provided by a file containing meta-data. IPFS, Swarm, and SAFE use the hash for content-addressing, meaning the content decides the address and content is retrieved by their address. Therefore, the acquired data can be directly verified. Additionally, SAFE uses self-encryption, where data is only restorable if it is the right data. Storj relies on the satellite nodes, which perform random audits on storage nodes utilizing hashes. Furthermore, satellite and storage nodes are evaluated with a reputation system to increase their credibility. In Arweave, data is stored in a blockweave, which is similar to a blockchain. Each block confirms its predecessor by including a hash pointer and therefore provides data integrity.

3) Availability: Due to node failure or maintenance, nodes can become unavailable, eventually decreasing the availability of stored chunks. Furthermore, node churn, e.g., due to unstable conncetions or Denial of Service attacks can render service unavailable over a certain time period. Therefore to improve availability, multiple copies of chunks might be required. Long term availability is a serious problem of P2P systems in general. The availability of content can be increased through active, passive, and cache-based replication. In Fig. 13, we provide an overview of the different availability mechanisms used by data networks. Popular content profits from cache-based replication, which can happen naturally through requests and as an optimization. Next to replication erasure codes can also increase the availability. While they introduce a per chunk storage overhead, files and missing chunks can be reconstructed without acquiring all chunks. Incentive mechanisms can improve replication mechanisms and ensure redundancy through monetary means. Note, that we discuss incentivization in a separate section.

BitTorrent and Hypercore rely only on passive replication and therefore volunteers hosting files. Arweave's blockweave is utilizing passive replication, ensuring replicas of blocks on the participants and therefore the content. However, every node can decide which content it stores based on its content policies. This means that not all content is available on all nodes. IPFS uses cache-based replication, additionally to the passive replication through pinning of chunks. SAFE uses cache-based replication and has data managers which are responsible to actively maintain a few redundant copies of chunks. Storj uses erasure codes instead of replication providing a certain safety margin against segment loss. Furthermore, the satellite nodes are responsible for auditing storage nodes repairing files as necessary. Swarm utilizes four methods: erasure codes, passive replication through pinning, cache-based replication, and active replication with the nearest neighbor set.

Replication is a simple method to increase availability, especially cache-based and passive replication rely on volunteers and require no coordination. Active replication requires some degree of coordination and communication to ensure that if a peer leaves the network, a handover of the data is ensured. Erasure codes are an alternative (or as shown by Swarm) an additional method to ensure replication. Erasure codes introduce less overhead while ensuring a similar degree of availability. However, erasure codes require coordination for sustaining the required amount of pieces and are applied on a file level, which also removes some possibility considering multiple files or file versioning. In IPFS, blocks are possibly used by different files, which adds additional replication and reduces overall storage requirements. This would not be possible with erasure code as they work as a set. Both methods, erasure coding and replication, have their advantages and disadvantages. The presented data networks seem to prefer replication.

D. *Incentivization*

Incentives are crucial in open/public P2P networks to motivate compliant behavior. Otherwise, we have to rely on altruism and benign peers. In the presence of “selfish” or malicious peers, this however might lead to an deteriorated data network. Most of the presented data networks employ some kind of incentive mechanism. An exception is Hypercore, which does not employ an incentive mechanism and is excluded from the following observation. An overview of the different incentive mechanisms is provided in Fig. 14.

One aspect of the incentive mechanism is compensation. While actions can be rewarded or punished with preferential treatment or depriving services, the data networks employ their own additional compensation methods. The compensation can be considered as a monetary incentive. The data networks use cryptocurrencies or crypto-tokens, which can be earned by or used to pay for services. In BitTorrent, BitTorrent Token supplements the service. The BitTorrent Token [46] is a TRC-10 utility token of the TRON blockchain [92]. IPFS itself does not employ a currency. But it uses Filecoin [56] to complement its protocol to incentivize data reliability/availability. Likewise, the other data networks use a cryptocurrency or token one way or the other to compensate services. Specifically, Swarm uses Ethereum (ether) [70, 93], SAFE uses Safecoins [77], Storj uses ERC-20 STORJ tokens [16, 94], and Arweave [17] uses its own cryptocurrency.

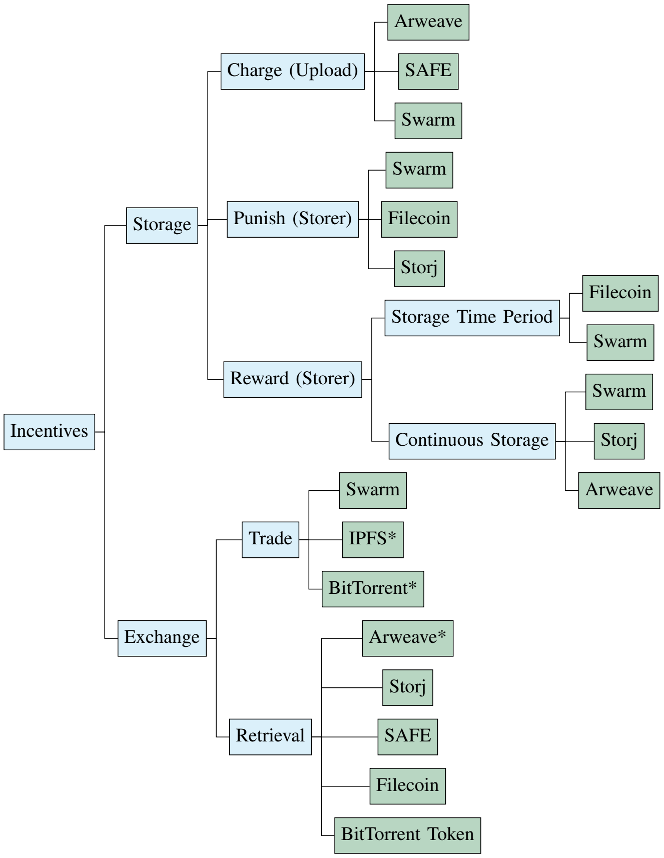


Fig. 14: Overview of different incentive mechanisms (data networks marked with an asterisk do not use monetary incentivization in this category).

Another aspect is the purpose of the incentive mechanism. We observe two different incentive purposes: promoting participation and increasing availability. Participation is stimulated by regulating content retrieval. In all presented data networks, peers keep track of the exchanged data. They can be further differentiated by a trade relationship, where the received and send data are compared and one sided observations, where peers are evaluated based on retrieved data.

Except for SAFE, all presented data networks use reputation or monetary incentive to prevent free-riding and promote active cooperation. SAFE has a reputation system and a certain reputation is necessary to actively participate in decisions. However, concerning the exchange of file, while SAFE rewards peers for answering request it does not punish peers for slow responses or charge clients for reading/consuming bandwidth. BitTorrent, IPFS, and Swarm compare send and received data. BitTorrent punishes unresponsive, free-riding peers by disconnecting from these peers, refusing further service. Additionally, the BitTorrent Token can be used to compensate peers which offer chunks. Swarm similarly punishes uncooperative peers by disconnecting them, however, Swarm also allows rebalancing the scale by issuing cheques to peers compensating a lack of send pieces. In IPFS, the Bitswap protocol ranks peers based on send and received data. Additionally, in Filecoin content retrieval is charged and peers providing the content are compensated with filecoin.

Arweave monitors the responsiveness of peers, ranking the peers, rewarding high ranking peers with preferential treatment. In Storj, satellite nodes compensate storage nodes for the provided bandwidth. Storj does not directly compensate the storage node and instead cumulates the used bandwidth.

It is noticeable that the compensation of file retrievals, in Filecoin, Swarm, and Storj is similar to a payment channel [95, 96], i.e., a bilateral channel between two peers used to exchange (micro-)payments instantaneously. Payment channels are backed by a cryptocurrency but do not require to commit every update to the blockchain and therefore promise improved scalability. Filecoin uses payment channels for the retrieval process, files are retrieved in small pieces and each piece is compensated. Swarm's chequebook contract behaves similar to a payment channel, where off-chain payment can be cashed in at any point in time. In Storj the bandwidth is monitored by allocating a pre-determined amount of bandwidth. The compensation of pieces in BitTorrent with BitTorrent Token also follows a payment channel.

The availability of files also benefits from the participation. By compensating file retrieval, nodes gain an incentive to cache files and answer requests. However, long-term availability is also important. Additionally, storing data on other devices might require an additional incentive for peers to accept the content. Therefore, the incentive mechanism sometimes focuses on rewarding and punishing storage nodes.

IPFS's Filecoin, Swarm, Storj, and Arweave reward nodes storing data. The reward is either for storing the data over time or for a specific time period. The time period is defined and nodes are pre- or postpaid, misbehaving storage nodes are then punished or not compensated. In IPFS's Filecoin, users rent specific storage for a time period. In Swarm, storage guarantees are sold. Swarm, Storj and Arweave reward nodes for storing data over a long time without defined time constraints. In Swarm, storage nodes can participate in a lottery, if they store certain chunks and might be rewarded for the continued storage. In Storj, storage nodes are compensated in time intervals for the data they stored during the interval, in case of storage failures the reward is instead used for file repair compensating the new nodes. In Arweave, the network is paid to store data for a long term. When a node creates a new block, proving storage of data, the node is compensated for its continued provision of storage capacity.

Punishment of nodes is used to guarantee storage in case of prepaid storage. If a node breaks its storage promises, it looses funds. A missed audit in Filecoin or failing to proof storage in Swarm reduces an escrow deposit of the storage node. In Storj part of the payment to new storage nodes is used as an escrow until the storage nodes gained enough reputation. The escrow will be kept if the node leaves the network too early. In Arweave, instead of punishing nodes, nodes can no longer be rewarded, if they stop storing blocks.

SAFE and Swarm charge the initial upload of data. However, this is a protection against arbitrary uploads rather than an increase in availability. Swarm finances the lottery with the upload fee. In Arweave, the upload of data is paid with transaction fees. Part of the fees go to the miner and part is kept by the network.

TABLE III: Overview of research on data networks.

Paper	System	Short Description
Performance and Structure		
[26]	IPFS	Read and write performance
[27]	IPFS	Cluster IoT data sharing
[28]	IPFS	Enhancing with ICN
[29]	IPFS	Meta-Data storage on blockchain
[30]	IPFS	On mobile devices
[32]	IPFS	Network mapping
[33]	IPFS	Network crawler
[34]	IPFS	Improving Bitswap
[35]	IPFS	Node identity, data availability
Confidentiality and Access Control		
[20]	IPFS	Blockchain-based, encryption
[21]	IPFS	Blockchain-based, modified client
[61]	IPFS	Blockchain-based, encryption
[22]	IPFS	Blockchain-based, modified application
[23]	IPFS	Blockchain-based, encryption
[97]	IPFS	Delegated content erasure
Security and Privacy		
[25]	IPFS	Botnet coordination
[66]	IPFS	Ransomware
[31]	IPFS	Eclipse attack
[67]	IPFS	Monitor data request
[80]	SAFE	CIA and possible attacks
[81]	SAFE	Security analysis
[83]	Storj	Denial-of-Service attack
[84]	Storj	Storing unencrypted data

VII. Research Areas and Open challenges

Previous generation of data networks had different network architectures, structured and unstructured, and used an incentive mechanism mainly to promote cooperation and prevent uncooperative behavior, e.g., free-riders, mainly with reputation systems [8]. Other incentive structures where also explored. The next generation uses mainly Kademlia-based architectures, and employs an incentive structure to increase availability and long term persistence.

The previous generation already faced some challenges, which still apply to the next generation data networks. In 2005, Hasan *et al.* [7] identified certain challenges that peer-to-peer systems have to overcome to gain acceptance for real-life scenarios. This includes deployment, naming, access control, DDoS attack protections, preventing junk data, and churn protection. We observe that the next generation data networks address these problems and provide possible solutions. However, the degree of maturity, the interaction with other mechanism, and the adoption rate need more consideration. In the literature review for the search of current generation data networks, we found a large body of literature utilizing or analyzing IPFS. Analyses of other systems are at most sparse. One reason could be lack of actual deployment, small user base or lack of implementation. Another reason, which this survey tries to address, is in our opinion a lack of concise and structured documentation. Some of the presented systems make it hard to get into the system, understand the concepts and show that the system is valid.

We observe five main challenges of data networks, which provide new opportunities for research: performance, confidentiality and access control, security, anonymity, and naming. An overview of existing research can be found in TABLE III.

A. Performance

A research direction which is already pursued by some researchers is the performance of the systems. Investigating the performance, read/write times, storage overhead, file lookup, churn resistance through simulations or tests, can be used to identify new use cases and fortify claims that a system might replace centralized counterparts. IPFS developed "Testground" for testing and benchmarking P2P systems at scale. In that sense the performance of Testground and its ability to replicate real systems, is also an area worthy to be researched. There exist other research analyzing the performance of IPFS, e.g., the read and write latency [26, 29], using IPFS cluster for Internet of Things data sharing [27], improving the system [28, 34], or analyzing the network [32, 33, 35]. Heinisuo *et al.* [30] showed that IPFS needed improvement to be used on mobile device due to high network traffic draining the battery. Research concerning IPFS's competitors is lacking. Additionally, Naik and Keshavamurthy [41] focus on the topic of churn in P2P networks.

B. Confidentiality and Access Control

The past and present generation of data networks provide some confidentiality and access control, but the systems are rather designed for public data than private data. The knowledge gained of nodes while storing data needs to be researched, this concerns not only information about the content of data but also meta-data like access patterns. The security of the existing access control needs to be investigated. There are research proposals for access control with blockchains [20, 21, 22, 23, 61], however the immutability of blockchains makes this questionable for private and personal data. Another aspect concerning private data is deleting data. While it is useful for censorship-resistance to prevent deletion of data, the possibility to delete personal, malicious or illegal data might raise acceptance of data networks. For example, Politou *et al.* [97] propose a mechanism for deleting content in IPFS. Investigating and improving the existing systems increases the trust in data networks. An increased trust in the confidentiality and the protection from unwarranted access can open these systems for storing private and personal data.

C. Security

As typical for security research, work in this area is in a constant back and forth between finding and fixing new vulnerabilities. In addition, research is also concerned with malicious activities using P2P data networks to exchange data with malware [25, 66].

With respect to security vulnerabilities, Prünster *et al.* [31] disclose an eclipse attack on IPFS and De Figueiredo *et al.* [83] showed a Denial-of-Service attack on Storj's test network. Furthermore, it is not only necessary to investigate known attack vectors, but also to investigate the existence of new attack vectors. For example, Storj acknowledges the possibility of an "Honest Geppetto" attack, where an attacker operates many storage nodes (honestly) for a long time, effectively controlling a large part of the storage capabilities. This control allows taking data "hostage" or taking down the data in general, rendering the data network inoperable. Another example is Frameup [84], where unencrypted data is stored on storage nodes, which could lead to legal issues. Storing arbitrary data might also pose a risk to the storage device. Security is the research area where we observe research beyond IPFS.

D. Anonymity

Next to confidentiality, which concerns data security and privacy, protecting the privacy of individuals is another relevant aspect; in particular, anonymity, which describes the inability to identify an individual in a group of individuals, i.e., unlinkability [98].

With respect to anonymity, various entities can be protected in data networks: the content creator, the storage node, and the user requesting content. From previous generation data networks, especially Freenet [2] and GNUnet [99] focused on protecting the identity of the different entities. Balduf *et al.* [67] already showed for IPFS the continued existance of privacy problems by identifying content requesters through monitoring data requests.

Due to the incentive mechanisms and the resulting charge of individuals it is hard to guarantee anonymity as at least pseudonyms are required. As soon as the incentive mechanism is used, information about the requester is gained. A distributed ledger recording transactions, e.g., Filecoin, Ethereum Swarm, Arweave, can reveal additional information and as a result participants are pseudonymous. When a central component authorizes requests and deals with incentivization, e.g., satellite nodes in Storj, requester, storage node and central component know each other. In case of incentivizing requests, the requesting node and storage nodes are revealed. The identity of requesters can be partly secured via forwarding strategies or proxies, e.g., Swarm, SAFE.

The first generation had systems like Freenet which aimed for anonymity and censorship-resistance. The anonymity of the current generation seems to fall behind the first generation. Despite advances in anonymous communication with mixnets or Tor [100], there are no data networks providing strong anonymity. In general, the provided anonymity guarantees and further enhancements need to be investigated. This includes the anonymity-utility trade-off and an analysis of different attacker models. Anonymity is not only important to protect the privacy of individuals, but is also important to guarantee the claimed censorship-resistance. If the identity of storage nodes can be easily inferred it is possible that, even though the network protects against deletion, law enforcement can enforce the censorship. This is a concern especially for systems like Swarm, where the location of a stored chunk is predetermined and node identity is linked to Ethereum pseudonyms.

E. Naming

Naming, in particular providing human-readable names in a distributed system, is a known challenge. The problem and its adjacent challenges is captured by Zooko's Triangle [101]. It describes the difficulty of building a distributed namespace, which is distributed (without a central authority), secure (clear-cut resolution), and human-readable.

In all systems, the addressing of data lacks either distribution (tracker-based BitTorrent and Storj) or human-readability (trackerless BitTorrent, Hypercore, IPFS, Swarm and SAFE). BitTorrent is a good example where the tracker is a central authority and in the case of trackerless BitTorrent the human-readable torrent is addressed with the not so readable infohash (hash of the torrent). In the v3.0 of Storj, the satellite is a central component.

The lack of human-readability is a result of self-authenticating data, where the data determines the address or the name of the data. If the data is changed the address changes. Therefore, human-readability is supported through a different mechanism, a naming independent of the content. An exception is Hypercore. In Hypercore, the data group is bound to the public key and the mutability inside the group is secured through versioning.

One solution to provide human-readability is name resolution. Name resolution allows the mapping of keys to self-authenticating content. The name resolution can provide human-readability and provide support for versioning of files. However, due to the possibility of updating the value and delays in propagation one could argue that security is violated, even if the key is unique. Independent of Zooko's Triangle, the name resolution announces content and gives ambiguous character strings meaning and should only be used for public data, unless the name resolution provides access control.

To this end, IPFS, Swarm, and SAFE provide some kind of naming service. In fact, IPFS provides two naming services, IPNS and DNSLink, which are used for different purposes. IPNS is used for mapping the hash of a public key to an IPFS CID, allowing mutable data. DNSLink uses DNS TXT records for mapping domain names to an IPFS address.

Swarm also provides two naming systems: single-owner chunks and ENS [75]. Single-owner chunks provide a data identification based on an owner and an identifier, providing a secure, non human-readable key with an updatable value. The Ethereum Name System is similar to DNS, where a record is mapped to an address.

Swartz [102] argued that a blockchain-based name service provides all three properties of Zooko's triangle. Anybody can register the name on the blockchain providing decentralization, the name can be anything providing human-readability, and the tamperproof ledger ensures unique names providing security. Following this line of argument, systems like Namecoin, Blockstack [90], and ENS, which adopt the idea of a blockchain-based name system, are developed. Although these systems exist, except for Swarm with ENS, none of the systems seem to provide a solution for Zooko's triangle. However, due to the lack of transaction finality and possible blockchain forks, it could be argued that blockchain-based system violate strong security aspects and only provide eventual security.

VIII. Conclusion and Lessons Learned

The first generation of P2P data networks taught us that P2P-driven file exchange works and has some major advantages, e.g., self-scalability. Another indicator for the persistence of this technology is BitTorrent's continued existence and wide user base. The first generation however also taught us weaknesses, e.g., a lack of long term availability. The next generation data networks builds upon and improved the previous generation, taking advantage of technological advancements and concepts to address the weaknesses.

In this survey paper, we studied this emerging new generation of P2P data networks. In particular, we investigated new developments and technical building blocks. From our qualitative comparison, we can conclude that except for the overlay structure the various data networks explore different solutions with respect to file management, availability, and incentivization. Most notably, explicit incentive mechanisms, e.g., using a cryptocurrency or some sort of token, seem to be ubiquitous to ensure long-term availability and the participant's engagement. We also see different measurements to ensure availability in the face of Denial-of-Service attacks or churn beyond incentive mechanisms, i.e., replication, erasure codes, or even a combination of both. Moreover, since many systems combine naming services and content addressing in a distributed architecture, they have the potential to reconcile the system properties of human readability, security, and decentrality as conjured by Zooko's triangle.

An important open task is now to investigate and evaluate the various building blocks. Especially, incentive mechanisms are notoriously difficult to design To some extent, we can consider the different deployments of P2P data networks as a large field test where we can observe the effects of certain design decisions. In general, P2P data networks have become part of the research agenda, either as a basis for other applications or as research object itself.

Yet, many challenges and open research questions remain, e.g., investigating anonymity, participant's privacy and access control, opening P2P data networks to a wider range of possible use cases. We therefore believe that this new generation of P2P data networks provide many exciting future research opportunities.

References

[1] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "Measurement study of peer-to-peer file sharing systems," in *Multimedia Computing and Networking 2002*, International Society for Optics and Photonics, vol. 4673, SPIE, Dec. 2001, pp. 156 –170.

[2] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *PET '00: Proceedings of the International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, Berkeley, CA, USA, Jul. 2000, pp. 46–66.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *SIGCOMM'01: Proceedings of the 2001 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, San Diego, CA, USA, Aug. 2001, pp. 149–160.

[4] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *SIGCOMM'01: Proceedings of the 2001 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, San Diego, CA, USA, Aug. 2001, pp. 161–172.

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Middleware '01: Proceedings of the 2001 IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, Nov. 2001, pp. 329–350.

[6] B. Cohen, "Incentives build robustness in bittorrent," in *P2PEcon '03: Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, USA, Jun. 2003, pp. 68–72.

[7] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, and R. Campbell, "A survey of peer-to-peer storage techniques for distributed file systems," in *ITCC '05: Proceedings of the 2005 International Conference on Information Technology: Coding and Computing*, vol. 2, Las Vegas, NV, USA, Apr. 2005, pp. 205–213.

[8] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[9] S. Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf.

[10] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the XOR metric," in *IPTPS'02: Proceedings of the 1st International Workshop on Peer-to-Peer Systems*, Cambridge, MA, USA, Mar. 2002, pp. 53–65.

[11] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[12] J. Benet, "IPFS - content addressed, versioned, P2P file system (draft 3)," Protocol Labs, Tech. Rep., Jul. 2014.

[13] V. Trón, *The book of swarm*, online, v1.0 pre-release, Jun. 2020.

[14] M. Ogden, K. McKelvey, M. Buus Madsen, and Code for Science, "Dat - distributed dataset synchronization and versioning," Dat Foundation, Tech. Rep., Jan. 2018.

[15] N. Lambert and B. Bollen, "The safe network a new, decentralised internet," 2014.

[16] Storj Labs Inc., "Storj: A decentralized cloud storage network framework v3.0," Storj Labs, Inc., Tech. Rep., Oct. 2018.

[17] S. Williams, V. Diordiiev, L. Berman, I. Raybould, and I. Uemlianin, "Arweave: A protocol for economically sustainable information permanence," arweave.org, Tech. Rep., Nov. 2019.

[18] M. S. Ali, K. Dolui, and F. Antonelli, "Iot data privacy via blockchains and IPFS," in *IOT '17: Proceedings of the 7th International Conference on the Internet of Things*, Linz, Austria, Oct. 2017, 14:1–14:7.

[19] R. Norvill, B. B. F. Pontiveros, R. State, and A. Cullen, "IPFS for reduction of chain size in ethereum," in *iThings/GreenCom/CPSCom/SmartData '18: Proceedings of the 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*, Halifax, NS, Canada, Aug. 2018, pp. 1121–1128.

[20] S. Wang, Y. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 6, pp. 38 437–38 450, 2018.

[21] M. Steichen, B. Fiz, R. Norvill, W. Shbair, and R. State, "Blockchain-based, decentralized access control for IPFS," in *iThings/GreenCom/CPSCom/SmartData '18: Proceedings of the 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data*, Halifax, NS, Canada, Aug. 2018, pp. 1499–1506.

[22] S. Khatal, J. Rane, D. Patel, P. Patel, and Y. Busnel, "Fileshare: A blockchain and ipfs framework for secure file sharing and data provenance," in *MoSICom '20: International Conference on Modelling, Simulation & Intelligent Computing*, Dubai, United Arab Emirates, Jan. 2020.

[23] V.-H. Hoang, E. Lehtihet, and Y. Ghamri-Doudane, "Privacy-preserving blockchain-based data sharing platform for decentralized storage systems," in *Networking '20: Proceedings of the 19th IFIP Networking Conference*, Paris, France, Jun. 2020, pp. 280–288.

[24] J. Xu, K. Xue, S. Li, H. Tian, J. Hong, P. Hong, and N. Yu, "Healthchain: A blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8770–8781, 2019.

[25] C. Patsakis and F. Casino, "Hydras and IPFS: a decentralised playground for malware," *International Journal of Information Security*, vol. 18, no. 6, pp. 787–799, 2019.

[26] J. Shen, Y. Li, Y. Zhou, and X. Wang, "Understanding I/O performance of IPFS storage: A client's perspective," in *IWQoS '19: Proceedings of the International Symposium on Quality of Service*, Phoenix, AZ, USA, Jun. 2019, 17:1–17:10.

[27] S. Muralidharan and H. Ko, "An interplanetary file system (IPFS) based iot framework," in *ICCE '19: Proceedings of the 37th IEEE International Conference on Consumer Electronics*, Las Vegas, NV, USA, Jan. 2019, pp. 1–2.

[28] O. Ascigil, S. Reñé, M. Król, G. Pavlou, L. Zhang, T. Hasegawa, Y. Koizumi, and K. Kita, "Towards peer-to-peer content retrieval markets: Enhancing IPFS with ICN," in *ICN '19: Proceedings of the 6th ACM Conference on Information-Centric Networking*, Macao, SAR, China, Sep. 2019, pp. 78–88.

[29] E. Nyaletey, R. M. Parizi, Q. Zhang, and K.-K. R. Choo, "Blockipfs - blockchain-enabled interplanetary file system for forensic and trusted data traceability," in *BLOCKCHAIN '19: Proceedings of the 2019 International Conference on Blockchain*, Atlanta, GA, USA, Jul. 2019, pp. 18–25.

[30] O.-P. Heinisuo, V. Lenarduzzi, and D. Taibi, "Asterism: Decentralized file sharing application for mobile devices," in *MobileCloud '19: Proceedings of the 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, Newark, CA, USA, Apr. 2019, pp. 38–47.

[31] B. Prünster, A. Marsalek, and T. Zefferer, "Total eclipse of the heart – disrupting the interplanetary file system," 2020.

[32] S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Mapping the interplanetary filesystem," in *Networking '20: Proceedings of the 19th IFIP Networking Conference*, Paris, France, Jun. 2020, pp. 289–297.

[33] S. Henningsen, S. Rust, M. Florian, and B. Scheuermann, "Crawling the IPFS network," in *Networking '20: Proceedings of the 19th IFIP Networking Conference*, Paris, France, Jun. 2020, pp. 679–680.

[34] A. De la Rocha, D. Dias, and Y. Psaras, "Accelerating content routing with bitswap: A multi-path file transfer protocol in ipfs and filecoin," p. 11, 2021.

[35] B. Guidi, A. Michienzi, and L. Ricci, "Data persistence in decentralized social applications: The ipfs approach," in *CCNC '21: Proceedings of the 18th IEEE Annual Consumer Communications & Networking Conference*, Las Vegas, NV, USA, Jan. 2021, pp. 1–4.

[36] F. Ashraf, A. Naseer, and S. Iqbal, "Comparative analysis of unstructured P2P file sharing networks," in *ICISDM '19: Proceedings of the 3rd International Conference on Information System and Data Mining*, Houston, TX, USA, Apr. 2019, pp. 148–153.

[37] T. D. Thanh, S. Mohan, E. Choi, S. Kim, and P. Kim, "A taxonomy and survey on distributed file systems," in *NCM '08: Proceedings of the 4th International Conference on Networked Computing and Advanced Information Management*, vol. 1, Gyeongju, South Korea, Sep. 2008, pp. 144–149.

[38] H. Huang, J. Lin, B. Zheng, Z. Zheng, and J. Bian, "When blockchain meets distributed file systems: An overview, challenges, and open issues," *IEEE Access*, vol. 8, pp. 50 574–50 586, 2020.

[39] N. Z. Benisi, M. Aminian, and B. Javadi, "Blockchain-based decentralized storage networks: A survey," *Journal of Network and Computer Applications*, vol. 162, p. 102 656, 2020.

[40] F. Casino, E. Politou, E. Alepis, and C. Patsakis, "Immutability and decentralized storage: An analysis of emerging threats," *IEEE Access*, vol. 8, pp. 4737–4744, 2019.

[41] A. R. Naik and B. N. Keshavamurthy, "Next level peer-to-peer overlay networks under high churns: A survey," *Peer-to-Peer Networking and Applications*, vol. 13, no. 3, pp. 905–931, 2020.

[42] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, "The bittorrent P2P file-sharing system: Measurements and analysis," in *IPTPS '05: Proceedings of the 4th International Workshop on Peer-To-Peer Systems*, Ithaca, NY, USA, Feb. 2005, pp. 205–216.

[43] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and improving a bittorrent networks performance mechanisms," in *INFOCOM '06: Proceedings of the 25th IEEE International Conference on Computer Communications*, Barcelona, Catalunya, Spain, pp. 1–12.

[44] R. L. Xia and J. K. Muppala, "A survey of bittorrent performance," *IEEE Communications Surveys and Tutorials*, vol. 12, no. 2, pp. 140–158, 2010.

[45] S. Ramanathan, A. Hossain, J. Mirkovic, M. Yu, and S. Afroz, "Quantifying the impact of blocklisting in the age of address reuse," in *IMC '20: Proceedings of the ACM Internet Measurement Conference*, Virtual Event, USA, Oct. 2020, pp. 360–369.

[46] BitTorrent Foundation, "Bittorrent (btt) white paper," BitTorrent Foundation, Tech. Rep., Feb. 2019.

[47] A. Loewenstern and A. Norberg, *Bep_0005.rst_post*, http://bittorrent.org/beps/bep_0005.html, Accessed: 2021-06.

[48] M. J. Dworkin, *SHA-3 standard: Permutation-based hash and extendable-output functions*, Aug. 2015.

[49] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNext '09: Proceedings of the 5th ACM International Conference on Emerging Networking Experiments and Technologies*, Rome, Italy, Dec. 2009, pp. 1–12.

[50] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[51] S. Mastorakis, A. Afanasyev, Y. Yu, and L. Zhang, "Ntorrent: Peer-to-peer file sharing in named data networking," in *ICCCN '17: Proceedings of the 26th International Conference on Computer Communication and Networks*, Vancouver, BC, Canada, Jul. 2017, pp. 1–10.

[52] A. Narayanan and J. Clark, "Bitcoin's academic pedigree," *ACM Queue*, vol. 15, no. 4, p. 20, 2017.

[53] F. Tschorsch and B. Scheuermann, "Bitcoin and beyond: A technical survey on decentralized digital currencies," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2084–2123, 2016.

[54] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, "On the security and performance of proof of work blockchains," in *CCS '16: Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, Oct. 2016, pp. 3–16.

[55] B. Joseph, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *SP '15: Proceedings of the 36th IEEE Symposium on Security and Privacy*, San Jose, CA, USA, May 2015, pp. 104–121.

[56] Protocol Labs, "Filecoin: A decentralized storage network," Protocol Labs, Tech. Rep., Jul. 2017.

[57] ——, *Libp2p - github*, https://github.com/libp2p, Acessed: 2021-07.

[58] J. Benet, D. Dalrymple, and N. Greco, "Proof of replication," Protocol Labs, Tech. Rep., Jul. 2017.

[59] R. Kumar, N. Marchang, and R. Tripathi, "Distributed off-chain storage of patient diagnostic reports in healthcare system using IPFS and blockchain," in *COMSNETS '20: Proceedings of 2020 International Conference on COMmunication Systems & NETworkS*, Bengaluru, India, Jan. 2020, pp. 1–5.

[60] J. Hao, Y. Sun, and H. Luo, "A safe and efficient storage scheme based on blockchain and ipfs for agricultural products tracking," *Journal of Computers*, vol. 29, no. 6, pp. 158–167, 2018.

[61] A. A. Battah, M. M. Madine, H. Alzaabi, I. Yaqoob, K. Salah, and R. Jayaraman, "Blockchain-based multi-party authorization for accessing ipfs encrypted data," *IEEE Access*, vol. 8, pp. 196 813–196 825, 2020.

[62] Y. Chen, H. Li, K. Li, and J. Zhang, "An improved p2p file system scheme based on IPFS and blockchain," in *BigData Congress '17: 2017 IEEE International Congress on Big Data*, Honolulu, HI, USA, Jun. 2017, pp. 2652–2657.

[63] A. Tenorio-Fornés, V. Jacynycz, D. Llop-Vila, A. Sánchez-Ruiz, and S. Hassan, "Towards a decentralized process for scientific publication and peer review using blockchain and IPFS," in *HICCS '19: Proceedings of the 52nd Hawaii International Conference on System Sciences*, Grand Wailea Maui, Hawaii, USA, Jan. 2019, pp. 1–10.

[64] S. Alam, M. Kelly, and M. L. Nelson, "Interplanetary wayback: The permanent web archive," in *JCDL '16: Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, Newark, NJ, USA, Jun. 2016, pp. 273–274.

[65] B. Confais, A. Lebre, and B. Parrein, "An object store service for a fog/edge computing infrastructure based on IPFS and a scale-out NAS," in *ICFEC '17: Proceedings of the 2017 IEEE 1st International Conference on Fog and Edge Computing*, Madrid, Spain, May 2017, pp. 41–50.

[66] C. Karapapas, I. Pittaras, N. Fotiou, and G. C. Polyzos, "Ransomware as a service using smart contracts and ipfs," in *ICBC '20: Proceedings of the 2020 IEEE International Conference on Blockchain and Cryptocurrency*, Toronto, ON, Canada, May 2020, pp. 1–5.

[67] L. Balduf, S. Henningsen, M. Florian, S. Rust, and B. Scheuermann, "Monitoring data requests in decentralized data storage systems: A case study of IPFS," *arXiv preprint arXiv:2104.09202*, 2021.

[68] Protocol Labs, *IPFS - github*, https://github.com/ipfs, Accessed: 2021-01.

[69] Ethersphere, *Ethersphere - github*, https://github.com/ethersphere, Accessed: 2021-01.

[70] G. Wood. (2014). "Ethereum: A secure decentralised generalised transaction ledger," [Online]. Available: http://gavwood.com/Paper.pdf.

[71] Hypercore Protocol developers, *Hypercore protocol - github*, https://github.com/hypercore-protocol, Accessed: 2021-01.

[72] MaidSafe, *Safe network - github*, https://github.com/safenetwork, Accessed: 2021-01.

[73] Storj Labs, *Storj labs - github*, https://github.com/Storj/, Accessed: 2021-01.

[74] ArweaveTeam, *Arweave - github*, https://github.com/ArweaveTeam, Accessed: 2021-01.

[75] N. Johnson, *Eip-137 - ethereum domain name service - specification*, https://eips.ethereum.org/EIPS/eip-137, Accessed: 2021-01.

[76] D. Keall, *How dat works*, https://datprotocol.github.io/how-dat-works/, Accessed: 2021-01.

[77] MaidSafe, *The safe network primer*, https://primer.safenetwork.org/, Accessed: 2021-01.

[78] D. Irvine, "Self-authentication," Tech. Rep., Sep. 2010.

[79] ——, "Self encrypting data," Tech. Rep., Jun. 2015.

[80] G. Paul, F. Hutchison, and J. Irvine, "Security of the maidsafe vault network," in *WWRF '14: Wireless World Research Forum Meeting 32*, Morocco, May 2014.

[81] F. Jacob, J. Mittag, and H. Hartenstein, "A security analysis of the emerging p2p-based personal cloud platform maidsafe," in *IEEE Trustcom/BigDataSE/ISPA '15: Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1, Helsinki, Finland, Aug. 2015, pp. 1403–1410.

[82] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[83] S. De Figueiredo, A. Madhusudan, V. Reniers, S. Nikova, and B. Preneel, "Exploring the storj network: A security analysis," in *SAC '21: Proceedings of the 36th ACM/SIGAPP Symposium On Applied Computing*, Gwangju, Korea, Mar. 2021, pp. 257–264.

[84] X. Zhang, J. Grannis, I. Baggili, and N. L. Beebe, "Frameup: An incriminatory attack on storj: A peer to peer blockchain enabled distributed storage system," *Digital Investigation*, vol. 29, pp. 28–42, 2019.

[85] M. Corallo. (Apr. 2016). "Bip 152: Compact block relay." Accessed: 2021-02, [Online]. Available: https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki.

[86] D. Vorick and L. Champine, "Sia: Simple decentralized storage," Nebulous Inc., Tech. Rep., Nov. 2014.

[87] M. Fukumitsu, S. Hasegawa, J.-y. Iwazaki, M. Sakai, and D. Takahashi, "A proposal of a secure p2p-type storage scheme by using the secret sharing and the blockchain," in *AINA '17: Proceedings of the 31st IEEE International Conference on Advanced Information Networking and Applications*, Taipei, Taiwan, Mar. 2017, pp. 803–810.

[88] Y. Jia, T. Moataz, S. Tople, and P. Saxena, "Oblivp2p: An oblivious peer-to-peer content sharing system," in *USENIX Security '16: Proceedings of the 25th USENIX Security Symposium*, Austin, TX, USA, Aug. 2016, pp. 945–962.

[89] C. Qian, J. Shi, Z. Yu, Y. Yu, and S. Zhong, "Garlic cast: Lightweight and decentralized anonymous content sharing," in *ICPADS '16: Proceedings of the 22nd IEEE International Conference on Parallel and Distributed Systems*, Wuhan, China, Dec. 2016, pp. 216–223.

[90] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *USENIX ATC '16: Proceedings of the 2016 USENIX Annual Technical Conference*, Denver, CO, USA, Jun. 2016, pp. 181–194.

[91] E. Kokoris-Kogias, E. C. Alp, S. D. Siby, N. Gailly, L. Gasser, P. Jovanovic, E. Syta, and B. Ford, "Calypso: Auditable sharing of private data over blockchains," *Cryptology ePrint Archive, 2018/209, Tech. Rep.*, 2018.

[92] TRON Foundation, "TRON advanced decentralized blockchain platform - whitepaper version 2.0," TRON Foundation, Tech. Rep., Dec. 2018.

[93] V. Trón, A. Fischer, D. A. Nagy, Z. Felföldi, and N. Johnson, "Swap, swear and swindle incentive system for swarm," Ethereum Foundation, Tech. Rep., May 2016.

[94] F. Vogelsteller and V. Buterin, *Eip-20 - erc-20 token standard*, https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md, Accessed: 2021-01.

[95] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," Jan. 2016.

[96] P. McCorry, M. Möser, S. F. Shahandashti, and F. Hao, "Towards bitcoin payment networks," in *ACISP '2016: Proceedings of the 21st Australasian Conference on Information Security and Privacy*, Melbourne, VIC, Australia, 2016, pp. 57–76.

[97] E. Politou, E. Alepis, C. Patsakis, F. Casino, and M. Alazab, "Delegated content erasure in ipfs," *Future Generation Computer Systems*, vol. 112, pp. 956–964, 2020.

[98] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity - a proposal for terminology," in *PET '00: Proceedings of the International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, Berkeley, CA, USA, 2000, pp. 1–9.

[99] K. Bennett, C. Grothoff, T. Horozov, I. Patrascu, and T. Stef, “Gnunet - a truly anonymous networking infrastructure,” In: Proc. Privacy Enhancing Technologies Workshop (PET, Tech. Rep., 2002.

[100] R. Dingledine, N. Mathewson, and P. F. Syverson, “Tor: The second-generation onion router,” in *USENIX Security ’04: Proceedings of the 13th USENIX Security Symposium*, San Diego, CA, USA, 2004, pp. 303–320.

[101] Z. Wilcox-O’Hearn, *Names: Distributed, secure, human-readable: Choose two*, https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html, Accessed: 2021-01.

[102] A. Swartz, *Squaring the triangle: Secure, decentralized, human-readable names*, http://www.aaronsw.com/weblog/squarezooko, Accessed: 2021-01.