# Towards Decentralised Cloud Storage with IPFS: Opportunities, Challenges, and Future Considerations

**Trinh Viet Doan**
Technical University of Munich

**Yiannis Psaras**
Protocol Labs

**Jörg Ott**
Technical University of Munich

**Vaibhav Bajpai**
CISPA Helmholtz Center for Information Security

***Abstract*—The InterPlanetary File System (IPFS) is a novel decentralised storage architecture, which provides decentralised cloud storage by building on founding principles of P2P networking and content addressing. IPFS is used by more than 230k peers per week and serves tens of millions of requests per day, which makes it an interesting large-scale operational network to study. While it is used as a building block in several projects and studies, its inner workings, properties, and implications have only been marginally explored in research. Thus, we provide an overview of the IPFS design and its core features, along with the opportunities that it opens, as well as the challenges that it faces because of its properties. Overall, IPFS presents an interesting set of characteristics and offers lessons which can help building decentralised systems of the future.**

## 1. Introduction

**THE PREVAILING BASELINE INTERNET INFRASTRUCTURE** is based on centralised cloud storage and management, as large cloud and Content Delivery Network (CDN) providers are seen to store significant amounts of user data in data silos. At the same time, these providers control most of today's Internet traffic, which results in a substantial centralisation of data storage points as well as traffic flow. To counteract this, several initiatives have focused on developing decentralised alternatives. However, the decentralised architecture of these networks also add new challenges and types of complexity. In this paper, we discuss these issues — both the benefits and challenges — in relation to the InterPlanetary File System (IPFS) [1], a protocol stack for decentralised cloud storage.

IPFS is an open-source set of protocols that combines multiple existing concepts from peer-to-peer (P2P) networking, Linked Data, and content-addressing to allow participants to exchange files, similar to Bittorrent. The uniquely named content on IPFS further leverages a self-

 Published by the IEEE Computer Society  

describing datatype that adopts concepts from git's versioning model, cryptographic hashing, and Merkle Trees.

**Deployment figures as of 2022.** The IPFS network has gained constant momentum over the last years: the public `ipfs.io` gateway (hosted by *Protocol Labs*, the primary maintainer of the open-source IPFS project) sees 3.7M unique users and serves more than 125 TBs of data in more than 805M requests per week as of 2022 [2]. Public IPFS gateways act as web servers for users that do not participate as peers themselves and allow such users to access content stored on IPFS over HTTP through a web browser, without needing any additional software (see § 2.5). Further, IPFS network measurements [3] in 2020 report numbers of 6k publicly reachable nodes on average, with a larger number of nodes not being reachable due to NAT. The authors' continued measurements[1] since then show that the number has increased to roughly 17k reachable nodes per crawl as of 2022, suggesting a substantial growth over the last years. Overall, *Protocol Labs* estimates the number of distinct active nodes per week to be higher than 230k nodes, which makes the IPFS network one of the largest permissionless, decentralised P2P storage and delivery networks in operation.

Due to its size, adoption, and unique characteristics, IPFS has also found significant support among several web-related projects: for instance, Cloudflare initially started hosting IPFS gateways in more than 150 of their data centers in September 2018[2] (later increasing to more than 200), which are still operational and serve the IPFS network to date. Mozilla Firefox added the `ipfs://` scheme to the list of allowed custom protocols in March 2018 to support protocol handlers for browser extensions[3]. As of January 2021, the Brave browser added native support for IPFS[4], making it possible to access `ipfs://` links directly from the browser window, similar to the native IPFS support in Opera browsers [2] since March 2020.

[1] https://trudi.weizenbaum-institut.de/ipfs_analysis.html
[2] https://web.archive.org/web/20180917192513/https://www.cloudflare.com/distributed-web-gateway/
[3] https://blog.mozilla.org/addons/2018/01/26/extensions-firefox-59/
[4] https://brave.com/ipfs-support/

Related work primarily studied IPFS as a storage mechanism for specific use cases, such as IoT and edge computing [4], [5], [6], malware [7], [8], or blockchain technology [9], [10] (see § 4.1). These use cases show that IPFS already plays a significant role in paving the way for decentralised applications as the reference P2P storage solution for hundreds of projects. However, the socio-technical impact of decentralised architectures such as IPFS have not been explicitly assessed yet, although IPFS and similar storage architectures are already used in several proposals and services [11].

In this paper, we first provide an overview of the technical design and building blocks of IPFS (§ 2). We then discuss its properties along with associated socio-technical opportunities and challenges (§ 3), before highlighting ongoing work (§ 4) and open challenges (§ 5). Ultimately, the goal of this paper is to distill open challenges and questions, and to encourage future studies and projects on decentralised storage, using the example of IPFS.

Note that we mainly discuss technicalities of IPFS; while we do discuss legal issues, we do so in the context of its technical properties and do not focus on legal or economic solutions.

## 2. Building Blocks and Principles

The design of IPFS, originally described in its whitepaper from 2014 [1], is inspired by various concepts from previous work in networking and file management, combining a set of protocols to build a distributed file system on top of a P2P network. IPFS applies concepts of Information-Centric Networking (ICN) [12], using uniquely identifiable fingerprints as content names to address and retrieve files rather than location-based references such as Uniform Resource Locators (URLs). However, in contrast to ICN, which primarily uses content-centric addressing at the network layer, the fingerprint-based addressing in IPFS happens at the overlay to ease deployment. Content-based addressing in IPFS leverages a set of protocols (called *multiformats*), which can generate *multihashes* that act as as an object's *Content Identifier (CID)* and guarantee backward compatibility (see § 2.2). This allows nodes to download different chunks of a file, based on their CIDs, from any peer in the network instead of

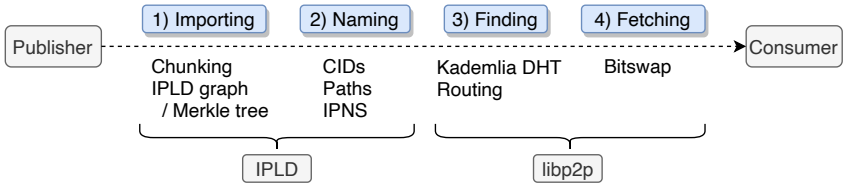


**Figure 1.** Outline of content publication (importing and naming) and retrieval process (finding and fetching), along with the involved IPLD and libp2p protocol stacks.

a single central server. As content is addressed and linked via unique hashes within Merkle trees, IPFS also supports file versioning similar to *git*.

Moreover, IPFS implements its protocols in a stack, i.e., its components can be extended or replaced when required. This modularity in the design is supported by *libp2p*[5], which is a modular P2P networking library for *process addressing* and focuses on data transfer processes, while supporting implementations for several network and transport layer protocols. libp2p also integrates protocols for content and peer routing through a Distributed Hash Table (DHT), a pubsub protocol, and a content-exchange protocol called *Bitswap*.

The modularity and flexibility provided by libp2p are promising features that can help with decentralising Internet services such as cloud storage. For instance, Filecoin (see § 4) leverages libp2p to build an incentivised and decentralised storage network, highlighting the reusability of IPFS' modules.

### 2.1. Overview: Content Transfer

**Figure 1** outlines the process of publishing and retrieving content within IPFS. In this example, a content publisher, Alice, wants to share a file with Bob, a consumer, via IPFS. More technical details for the specific steps are described in the following subsections.

**Joining the IPFS Network and Importing Content.** As a permissionless P2P network, IPFS allows peers to join freely; each node is identified by its public/private key pair. Alice sets up and runs an IPFS node that connects to (pre-defined) bootstrap nodes to get initial connections for its main content routing system, the DHT, which is inspired by Kademlia [13] and Coral DHT [14]. Alice's node then imports the file she wants to share by chunking and uniquely hashing each chunk, arranging them in a Merkle graph (cf. **Figure 2**), which is IPFS' main data model.

**Publishing Content.** The created fingerprints, i.e., the *CIDs*, are then added to *provider records* in the IPFS DHT, which state that Alice's node caches the respective file pieces. The provider records are placed at nodes whose `peerID` is close to the published object's CID; the closeness is based on Kademlia, meaning the distance between the IDs is calculated through the bitwise exclusive or (XOR) function [13]. Her node also adds its *peer records* to the DHT to describe how to connect to it, i.e., IPv4/IPv6 addresses, supported transport protocols, and port numbers. The collection of those protocol-level identifiers is called *multiaddress* of the peer.

**Finding and Fetching Content.** Alice then shares the file's root CID (out of band, i.e., not through IPFS) with Bob, whose IPFS client can lookup the CID in the DHT to find provider records of peers that cache the content (in this case Alice). Looking up her peer records, his node learns about how to connect to Alice's IPFS node, establishes a connection to her node, and requests the CIDs, ultimately retrieving the chunks from Alice's node via the *Bitswap* protocol. It is important to clarify that Bob's node can verify the integrity of the whole file using the hashes and reconstructing the IPLD graph, without relying on certificates or having to put trust on the DNS system, as is the case with HTTP transfers.

**Caching and Re-Serving Content.** Bob's node then caches the file for a configurable amount of time and adds his provider records for the chunks to the DHT, so that another interested consumer (e.g., Carol) could now retrieve the chunks from either Alice or Bob.

[5] https://libp2p.io/

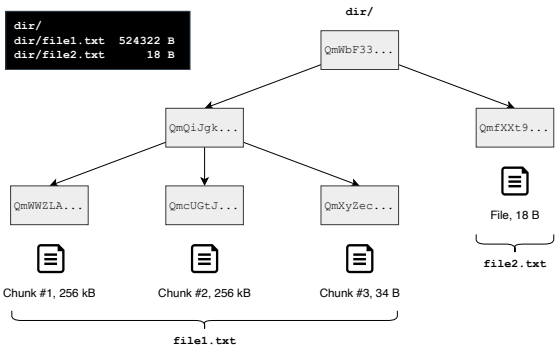


**Figure 2.** Constructed MerkleDAG for an example directory `dir/` containing `file1.txt` and `file2.txt`, which are chunked into addressable blocks of 256 kB.

### 2.2. File Processing and Naming

Any data item added to the network is chunked into blocks of 256 kB (default block size). Each block is named and addressed individually by a unique hash-based CID. These blocks are arranged as leaf nodes in a Directed Acyclic Graph (DAG) to build an *InterPlanetary Linked Data (IPLD) graph*, which represents the main data model of IPFS. The root of this hash-linked graph is the unique CID for the specific input file (or directory), which allows peers to easily verify the integrity through the hash-chained DAG (cf. Merkle tree). Therefore, any modification in the initial data changes the CID of the corresponding chunk and subsequently cascades to other parts of the DAG it is linked to. An example for such a graph and its vertices/edges for a directory with two files is shown in **Figure 2**. Each vertex can be addressed and exchanged individually through its CID, thus, representing either the whole directory, whole files, or individual chunks. With this data model, a hash can also refer to more complex structures such as subgraphs or other graphs. Thus, IPFS' naming and addressing structure adopts Linked Data principles which build the foundation of the Semantic Web.

The CID of each block contains a self-describing *multihash*, which prepends both the used hash function and the digest length to the actual digest value. This construct is referred to as CIDv0; for the newer CIDv1, this multihash is further prepended by: 1) a prefix for the base encoding used (*multibase*) to encode the remaining (binary) fields of the CID, 2) a CID version identifier, and 3) a codec type (*multicodec*), which describes how the actual data is encoded. Hence, the self-describing CID structures facilitate the implicit replacement of hash and encoding functions, which allows different representations for the same data/digest and provides compatibility between those.

### 2.3. Pathnames and IPNS

IPFS adopts a pathname scheme with a global namespace, similar to the Self-certifying File System (SFS) [15]. Once the objects are named, they can be navigated using regular path syntax as commonly employed in file systems or Web URLs. For this, `/ipfs/` is added as a prefix to the object CID to denote that IPFS is used, followed by the fingerprint and the path syntax, e.g., `/ipfs/<CID>/foo/bar.baz`.

A CID does not support mutable content due to its hash-based nature. To avoid having to replace identifiers in cases of file modification and to keep the content name consistent, IPFS employs a naming system called *InterPlanetary Name System (IPNS)*. The fingerprint used for IPNS is derived from a node's public key, i.e., each node can only create a single IPNS identifier. While the identifier itself is static to allow sharing, the CID it refers to can be modified arbitrarily by the publishing node, which enables mutability for the referred object behind the static IPNS identifier. Thus, an IPNS identifier (denoted by an `/ipns/` instead of an `/ipfs/` prefix) can represent any arbitrary CID, such as a chunk, file, or data structure.

Alternatively, if clients have access to DNS records and can modify them, they can store the IPFS or IPNS identifier in a DNS `TXT` record[6] (with low TTL values to avoid stale caches) and update it whenever required. However, leveraging DNS links also inherits potential shortcomings of DNS (e.g., centralisation, censorship, attacks) that need to be considered.

### 2.4. Content Storage and Caching

When a node adds content to the IPFS network, the content is only made available to other peers but *not* replicated. By default, content is only replicated when it is explicitly requested and

[6] https://dnslink.org/

retrieved by another peer, i.e., following a strict pull model: unrequested content is *not* disseminated, which is a major difference to most centralised cloud storage solutions. The requesting node then caches retrieved content locally, with a configurable caching behavior and duration. Therefore, IPFS does not force peers to cache arbitrary content (which they are not interested in themselves) on behalf of other peers. Among other reasons, this is done to avoid legal implications for the caching node.

Locally cached content is removed periodically by the automatic garbage collection. To become a "permanent" provider of some content item, IPFS uses a *pinning* mechanism, which allows nodes to mark files as permanent in local storage to have them excluded from garbage collection. This means that unpopular content, unless explicitly pinned, will eventually disappear from the network. Popular content, on the other hand, will be constantly requested and re-cached by peers and, ultimately, disseminated, which improves content availability. Consequently, due to garbage collection and churn of nodes, IPFS only provides best-effort storage without any guarantees for availability, meaning that responsibility for content availability is *not* inherently delegated to the network. However, dedicated *pinning services* (see § 3.4) can provide availability guarantees for content if necessary.

### 2.5. IPFS Public Gateways

In traditional P2P networks, users that are interested in retrieving information from the network or using its offered services are required to participate. Typically, this comes with the requirement of running some client software. However, this may not be a feasible option, e.g., as devices might be resource-constrained. In IPFS, any peer with a publicly reachable IP address can act as a gateway, which allows external users to access content from IPFS using only HTTP instead. IPFS gateways serve as a web server for clients and as a DHT server for the IPFS network, making access to content seamless. An example for one of these gateways is `ipfs.io`, which can be given either an IPFS or an IPNS CID to request the content: a user would navigate to `https://ipfs.io/ip[f|n]s/<CID>` with their browser to request the object with the respective CID from the `ipfs.io` gateway. If not already cached, the gateway retrieves the content from peers in the IPFS network. After retrieving all chunks from the network, the gateway then serves the file to the requesting user via HTTP.

Note that gateways are not essential but rather supportive building blocks by design: they are designed to provide another way of retrieving files from IPFS, in particular, to assist clients which are resource-constrained or cannot participate as a peer. However, gateways can also lead to centralisation and dependencies (cf. supernodes in traditional P2P networks), along with free-rider problems (see § 3.4).

## 3. Properties of IPFS

The usage of IPFS for file storage in existing applications can bring many benefits, such as built-in integrity checks and inherent content deduplication through content-based addressing, which decouples file retrieval from specific locations. Simultaneously, it presents special characteristics such as providing no authorisation/access control by default owing to its decentralised and permissionless nature. Thus, developers using IPFS need to carefully take its inherent properties into account and accommodate them within the remit of their application. We discuss some of the key properties of IPFS and implications in the following.

### 3.1. Persistence of Names and File Integrity

Identifying content by a unique multihash rather than a location address gives more flexibility to the network in different ways: resources can be used more efficiently since duplicate files, and even duplicate chunks of files, are assigned the same identifier and can be handled, linked, and re-used appropriately to not waste additional resources. In contrast, in traditional host-based addressing, duplicate files might end up being stored redundantly under different file names and different location-based identifiers. Note that although centralised cloud providers may also run data deduplication schemes, these schemes are not always applied at the chunk-level and follow different data models [16].

Explicit content-based addressing also facilitates on-path and in-network caching, as one can verify the integrity of chunks using the multihash.

Thus, there is no need to trust third parties to deliver the correct file pieces, which circumvents potential centralisation by removing dependency on a single network or (original) content provider.

One property of the persistence of content-based names in IPFS is that content identifiers change when the content itself is updated, e.g., in case of dynamic content. This is in contrast to the current HTTP-based model, where the URL remains when the content it represents changes. Persistence of content names is a desirable property in the design of IPFS, which consequently creates the need for additional mechanisms to deal with dynamic content. Examples for those are IPNS (see § 2.3) or libp2p's pubsub protocols, which allow peers to create pubsub channels between each other to dynamically broadcast and listen to events.

Further, there are several applications on top of IPFS that support dynamic and mutable data to ease development: *Textile* provides a set of developer tools and focuses on data ownership, allowing applications to make use of the data their users provide through so called Buckets, which resemble dynamic personalised cloud storage services based on IPFS and libp2p. A similar SDK is *Fission*, which facilitates the publication of frontend apps via built-in backend solutions that handle (user) data management over IPFS.

### 3.2. Data Auditability, Censorship Resistance, and Privacy

One area where IPFS embodies privacy by design principles more closely than HTTP is in allowing more precise and comprehensive auditability of stored data. For example, in the context of attempting to delete a subject's personal data after consent has been revoked, the persistent nature of IPFS content identifiers allows users to know if files that include associated personal data are stored: one can verify whether an asset is cached and shared by scanning for the asset's content fingerprint (as well as fingerprints of chunks), further facilitated by merkle-linking. This has the potential to provide a more usable basis for applications built on IPFS to comply with both the specific provisions and the broader aims of data protection regulations.

At the same time, content cannot be explicitly censored in IPFS, given it operates as a distributed P2P file system with no central indexing entity. Since peers are not organised in a hierarchy, there is no authority node in the network that can prohibit the storage and propagation of a file, or delete a file from other peers' storage. Consequently, censorship of unwanted content cannot be technically enforced, which, e.g., represents an opportunity for users that are suppressed in their freedom of speech. Note that censorship of unwanted content within the borders of an oppressive nation state is different to a globally applicable legal request to remove content. The censorship resistance that IPFS offers is achieved by replicating content (e.g., snapshots of Wikipedia [17]) among different peers, which makes it difficult to censor all provider nodes altogether. Moreover, any public IPFS gateway can also retrieve and deliver the specific content to users, adding further censorship resistance, especially when hosted by larger CDN providers such as Cloudflare; users can simply use another gateway in case one gateway is being blocked, or use Tor to access a gateway. While this is a fairly simplistic censorship model, note that more sophisticated censorship approaches also take more advanced information patterns (such as traffic fingerprinting) into account, which has not been extensively studied for IPFS yet, especially considering IPFS metadata (CID requests, DHT records, ...) are only transport-encrypted or fully public and can, thus, be monitored.

Hence, a lookup of a fingerprint to find out which peers store a specific file (based on provider records) can reveal their IP address via their peer records and identify those peers. There are two important points to stress w.r.t. privacy:

1) The primary current use-case of IPFS is providing storage and access to public data in the network, e.g., datasets or websites. Given the wide range of applications that IPFS envisions to be able to support, IPFS currently does not support access control or privacy at the protocol layer, as applications (such as a public pastebin) might not necessarily require privacy. Instead, privacy designs (like content encryption) currently have to be implemented at the application layer. Modular approaches to enhance privacy that simultaneously support a wide range of applications remains an open research problem at the time of writing.

2) IPFS peers can control what they share with others in the network. A peer by default announces to the network every CID in its cache, i.e., content that it has either published, or requested and fetched previously. A peer can refrain from reproviding content that it has fetched, although the preceding CID requests to peers need to be revealed by design. While it can still fetch and consume content as well as keep it in its local cache for later consumption, it can skip letting the DHT know that it has the particular piece of content locally. Local reprovider settings allows each peer to control what it shares with the network.

### 3.3. Network Partition Tolerance

Due to its decentralised nature and P2P structure with no essential central components, IPFS can still operate in cases of network partitions or in offline scenarios. Partitions do not fully impair the content publication and retrieval process; thus, as long as the requested objects and associated records are known and available within the same subgraph, IPFS is tolerant against partitions and does not require full Internet connectivity if the providing peers are reachable. Similarly, private IPFS networks among a set of machines can be specifically built using *IPFS Cluster*[7], which allows deployment of IPFS in local networks.

### 3.4. Incentives for Participation

IPFS was designed as a permissionless, best-effort P2P network and, therefore, does not integrate any incentive schemes. The operation of an IPFS node incurs costs for infrastructure maintenance in terms of bandwidth, storage, and power. However, after retrieving the desired objects, there is no incentive for a user to keep the node running, resulting in short sessions and high network churn as observed by measurements of the IPFS network [3].

Both free-riding and legitimate consumers can retrieve content conveniently over HTTP through an IPFS gateway, which does not require participation in the network at all. This poses an open research question on whether gateways arise naturally as supernodes in the P2P network, resulting from trying to support all types of clients along with an absence of incentives. Nevertheless, incentivising for consistent and continuous participation (and ultimately a sustainable IPFS network) needs to be considered at the application-layer to avoid free-riding and centralisation around gateways and super nodes. *Pinning services*, which are dedicated third parties that pin files to provide improved/guaranteed availability for monetary compensation, alleviate the problem of lacking incentives but do not fully solve it, while also requiring trust in the service.

## 4. Current Directions & Ongoing Work

### 4.1. Current Directions

**IPFS-based Projects.** The presented properties of IPFS (§ 3) highlight various opportunities for current and future projects as well as things that they should take into consideration. For instance, IPFS is used as off-chain storage for many projects on various blockchains such as Ethereum: due to their append-only nature, blockchains grow continuously in size. Proposed models [9], [10] outsource the transaction and smart contract data to IPFS, while the transactions only contain the immutable IPFS CID; the data with the respective CID itself is stored on IPFS, which reduces the blockchain's size and allows the data to be retrieved from IPFS for transaction validation or contract execution.

The *IPFS Ecosystem Directory*[8] provides an overview of existing projects building on IPFS. These range from content delivery over data persistence/governance to social media, e-commerce, and audio/video entertainment platforms. One project that is closely related and attempts to solve the lack of incentives along with an improvement of its best-effort content storage is *Filecoin*.

**Filecoin.** The Filecoin P2P network builds on top of IPFS and incentivises the storage of objects, aiming to provide distributed storage that is cheaper than centralised cloud storage solutions. Filecoin uses the same building blocks (§ 2) as IPFS, with content addressing via CIDs at its core. Unlike IPFS, which does not replicate content at other peers unless those peers explicitly request the content, Filecoin provides cryptoeconomic incentives to its participants: storage

[7] https://ipfscluster.io/

[8] https://ecosystem.ipfs.io/

and replication of content in the network are rewarded with cryptographic tokens to facilitate higher availability, faster retrieval, and to counteract node churn. In exchange for a fee, peers can close storage deals between each other via smart contracts[9] to provide persistent storage (cf. service level agreements), which is provable through proof-of-replication and proof-of-spacetime[10]. Thus, Filecoin improves IPFS' best-effort storage and delivery service by providing incentive mechanisms; while the aforementioned IPFS pinning services (§ 3.1) can also improve IPFS' best-effort approach, Filecoin does not require trust between the parties of a contract due to its decentralised, blockchain-based foundations. In recent years, the Filecoin network has grown substantially: in 2021[11], Filecoin provided 1.7 EiB of storage capacity, of which 1.6 PiB were used for storage. As of July 2022, the network provides roughly 17.8 EiB of storage capacity[12], of which 123.3 PiB are used for data storage across 5.1M active storage deals[13], indicating significant growth in the last 1.5 years.

### 4.2. Ongoing Work

To tackle some of the technical challenges, the IPFS community has developed a variety of solutions recently, e.g., support of NAT hole punching. However, other challenges, such as the monitoring and benchmarking of the IPFS network in general to provide a basis for future studies and optimisations, remain ongoing work.

**Hole Punching.** IPFS nodes located behind firewalls and NATs, typically in home networks, may face connectivity issues, as direct connections between such peers cannot be established. In recent libp2p updates, this was addressed by introducing hole punching [18]: similar to the ICE, STUN, and TURN protocols used for NAT traversal (cf. RFCs 8445, 8489, 8656), IPFS leverages a (third) *public* node as a rendezvous point $R$ between two private nodes $A$ and $B$. In a nutshell, $R$ relays a connection between $A$ and $B$, while helping them to coordinate time-synchronised dials to achieve hole punching and a direct connection upgrade. Overall, hole punching can improve node reachability and allow for more P2P connections between peers, which will positively impact file availability and transfers.

**Monitoring and Benchmarking.** To determine the baseline of IPFS deployment and performance, a recent study [19] has performed both passive and active measurements from various vantage points around the globe. The authors show that peers are primarily hosted in non-cloud provider networks (97.7%) and on consumer hardware, although the top 10 ASes (out of 2.7k) host around 2/3 of all peers. Regarding performance, the median delay to make content available (i.e., publish the provider records in the DHT) is 33.8 seconds, of which the DHT walks account for nearly 90% of that total delay on average, while the remaining delay consists of actually pushing the records to the responsible peers. On the retrieval side, the median delay is much lower at 2.9 seconds (including CID lookup latency), as the median DHT walk is only 622 milliseconds. Compared to an equivalent HTTPS request for a file, the authors estimate that IPFS takes at least four times as long for content retrieval, although note that the measured numbers have high variance due to regional differences.

Overall, the measurements indicate that the content publication and retrieval times can still be significantly optimised (especially for content publication), which is currently a major action item in the IPFS community. An example for a current line of work to speed up publishing provider records in the DHT is labeled *Optimistic Provide*[14]: when pushing provider records to peers in the DHT, by default, a peer looks for the 20 closest peers to the CID; the publishing peer will walk the DHT and pass many intermediate peers while trying to find the closest ones. To reduce the number of lookups and the required time, Optimistic Provide proposes that a peer first "arrives" at an intermediate peer, then calculates the likelihood of another peer being even closer to the CID (using network size estimations); if this probability is below a certain threshold, the peer will optimistically store the provider records at that intermediate peer instead, trading off some accuracy for reducing the number of

[9] https://fvm.filecoin.io/
[10] https://spec.filecoin.io/#section-algorithms.pos
[11] https://filecoin.io/blog/posts/filecoin-in-2021-looking-back-at-a-year-of-exponential-growth/
[12] https://stats.filecoin.io/
[13] https://storage.filecoin.io/
[14] https://github.com/dennis-tra/optimistic-provide

(iterative) lookups. First experiments have shown that despite the overall publication time taking tens of seconds, primarily due to the DHT walks as mentioned above, around 90% of peers finally chosen (to store the provider record with) are found within the first second, which indicates that an optimistic approach can make the providing process an order of magnitude faster in the future.

## 5. Conclusion and Open Challenges

We provided an overview of IPFS and its core features, presenting how its combination of multiple networking protocols and P2P concepts build a foundation for decentralised cloud storage. Its building blocks enable peer-assisted file distribution and delivery to move away from centralised cloud storage by providing persistence of names, deduplication, and integrity-checks for files through Content Identifiers (CIDs), censorship resistance, network partition tolerance, and ultimately decentralisation, among other properties. Previous studies use IPFS for a variety of projects and proposals [11] within the IPFS ecosystem (see Footnote 8). Nevertheless, IPFS has yet to overcome and solve challenges such as access control/built-in privacy, participation incentives, or persistent availability and replication of content. Future considerations, such as the integration with Filecoin, aim to overcome some of IPFS' challenges regarding incentives and content availability. Together with the native IPFS support in popular browsers, these are important steps in stimulating the growth of the network and moving towards a more decentralised Internet in the future. The performance of these decentralised solutions is a very timely research topic, which we encourage the community to undertake. As such, future work on distributed storage in general, i.e., also beyond IPFS, should consider both opportunities and challenges discussed to develop suitable decentralised storage systems for the Future Internet and its applications. Based on the discussion presented in this paper, we identify several broad, open research questions w.r.t. IPFS and distributed storage in general:

- How does P2P-based content storage and retrieval compare to traditional cloud storage or CDNs technologies?
- How can availability, retrieval, and delivery of content be improved in a decentralised environment?
- What are the important properties that are required for a decentralised CDN?
- How can we achieve full user anonymity when fetching and retrieving content in permissionless P2P networks?
- How can the use and adoption of such decentralised technologies be incentivised?

## REFERENCES


1. J. Benet, "IPFS - Content Addressed, Versioned, P2P File System," *CoRR*, vol. abs/1407.3561, 2014. [Online]. Available: http://arxiv.org/abs/1407.3561
2. Protocol Labs, "IPFS in 2021: The Backbone of Web3's Mainstream Momentum," 2022, accessed 2022-03-31. [Online]. Available: https://blog.ipfs.io/2022-01-11-IPFS-in-2021/
3. S. A. Henningsen *et al.*, "Mapping the Interplanetary Filesystem," in *2020 IFIP Networking Conference*. IFIP, 2020, pp. 289–297. [Online]. Available: https://ieeexplore.ieee.org/document/9142766
4. M. S. Ali *et al.*, "IoT data privacy via blockchains and IPFS," in *IOT 2017*. ACM, 2017, pp. 14:1–14:7. [Online]. Available: https://doi.org/10.1145/3131542.3131563
5. B. Confais *et al.*, "An Object Store Service for a Fog/Edge Computing Infrastructure Based on IPFS and a Scale-Out NAS," in *ICFEC*. IEEE, 2017, pp. 41–50. [Online]. Available: https://doi.org/10.1109/ICFEC.2017.13
6. S. Krejci *et al.*, "Blockchain- and IPFS-Based Data Distribution for the Internet of Things," in *ESOCC 2020*. Springer, 2020, pp. 177–191. [Online]. Available: https://doi.org/10.1007/978-3-030-44769-4_14
7. C. Patsakis and F. Casino, "Hydras and IPFS: a decentralised playground for malware," *Int. J. Inf. Sec.*, vol. 18, no. 6, pp. 787–799, 2019. [Online]. Available: https://doi.org/10.1007/s10207-019-00443-0
8. F. Casino *et al.*, "Immutability and Decentralized Storage: An Analysis of Emerging Threats," *IEEE Access*, vol. 8, pp. 4737–4744, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2962017
9. Q. Zheng *et al.*, "An Innovative IPFS-Based Storage Model for Blockchain," in *WI 2018*. IEEE Computer Society, 2018, pp. 704–708. [Online]. Available: https://doi.org/10.1109/WI.2018.000-8
10. R. Norvill *et al.*, "IPFS for Reduction of Chain Size in Ethereum," in *IEEE Blockchain 2018*. IEEE, 2018, pp.

1121–1128. [Online]. Available: https://doi.org/10.1109/Cybermatics_2018.2018.00204

11. E. Daniel and F. Tschorsch, "IPFS and Friends: A Qualitative Comparison of Next Generation Peer-to-Peer Data Networks," *IEEE Commun. Surv. Tutorials*, vol. 24, no. 1, pp. 31–52, 2022. [Online]. Available: https://doi.org/10.1109/COMST.2022.3143147
12. G. Xylomenos *et al.*, "A Survey of Information-Centric Networking Research," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 2, pp. 1024–1049, 2014. [Online]. Available: https://doi.org/10.1109/SURV.2013.070813.00063
13. P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *IPTPS*. Springer, 2002, pp. 53–65. [Online]. Available: https://doi.org/10.1007/3-540-45748-8_5
14. M. J. Freedman and D. Mazières, "Sloppy Hashing and Self-Organizing Clusters," in *IPTPS*. Springer, 2003, pp. 45–55. [Online]. Available: https://doi.org/10.1007/978-3-540-45172-3_4
15. D. Mazières *et al.*, "Separating key management from file system security," in *SOSP*. ACM, 1999, pp. 124–139. [Online]. Available: http://doi.acm.org/10.1145/319151.319160
16. Y. Shin *et al.*, "A Survey of Secure Data Deduplication Schemes for Cloud Storage Systems," *ACM Comput. Surv.*, vol. 49, no. 4, pp. 74:1–74:38, 2017. [Online]. Available: https://doi.org/10.1145/3017428
17. The IPFS Team, "Uncensorable Wikipedia on IPFS," 2017, accessed 2022-03-31. [Online]. Available: https://ipfs.io/blog/24-uncensorable-wikipedia/
18. M. Inden, "Hole punching in libp2p - Overcoming Firewalls," 2022, accessed 2022-07-14. [Online]. Available: https://blog.ipfs.io/2022-01-20-libp2p-hole-punching/
19. D. Trautwein *et al.*, "Design and Evaluation of IPFS: A Storage Layer for the Decentralized Web," in *SIGCOMM 2022 Conference*. ACM, 2022, pp. 739–752. [Online]. Available: https://doi.org/10.1145/3544216.3544232

**Trinh Viet Doan,** is a PhD student at Technical University of Munich, Germany. His research interests include Internet measurements as well as consolidation and decentralisation of Internet architectures and infrastructures. He received his Master's degree from Technical University of Munich in 2017. Contact him at doan@in.tum.de.

**Yiannis Psaras,** is a research scientist in the Resilient Networks Lab at Protocol Labs. His research interests include Information- or Content-Centric Networks, as well as resource management techniques for current and future networking architectures with particular focus on routing, caching, and congestion control. He received his PhD degree from Democritus University of Thrace in 2008. Contact him at yiannis@protocol.ai.

**Jörg Ott,** has been the Chair of Connected Mobility, Faculty of Informatics, Technical University of Munich, since August 2015. His research interests are in network architectures, (transport) protocols, and algorithms for connecting mobile nodes to the Internet and to each other. He received his PhD degree from TU Berlin in 1997. Contact him at ott@in.tum.de.

**Vaibhav Bajpai,** is an independent research group leader at CISPA Helmholtz Center for Information Security, Hannover. His current research focuses on improving Internet operations (e.g., performance, security, and privacy) using data-intensive methods and by building real-world systems and models. He received his PhD degree from Jacobs University Bremen in 2016. Contact him at bajpai@cispa.de.