Design and Evaluation of IPFS: A Storage Layer for the Decentralized Web

Dennis Trautwein
Protocol Labs &
University of Göttingen
dennis.trautwein@protocol.ai

Aravindh Raman
Telefonica Research
aravindh.raman@telefonica.com

Gareth Tyson
Hong Kong University of Science &
Technology (GZ)
gtyson@ust.hk

Ignacio Castro
Queen Mary University of London
i.castro@qmul.ac.uk

Will Scott
Protocol Labs
will@protocol.ai

Moritz Schubotz
FIZ Karlsruhe – Leibniz Institute for
Information Infrastructure
moritz.schubotz@fiz-karlsruhe.de

Bela Gipp
University of Göttingen
gipp@uni-goettingen.de

Yiannis Psaras
Protocol Labs
yiannis@protocol.ai

ABSTRACT

Recent years have witnessed growing consolidation of web operations. For example, the majority of web traffic now originates from a few organizations, and even micro-websites often choose to host on large pre-existing cloud infrastructures. In response to this, the "Decentralized Web" attempts to distribute ownership and operation of web services more evenly. This paper describes the design and implementation of the largest and most widely used Decentralized Web platform — the InterPlanetary File System (IPFS) — an open-source, content-addressable peer-to-peer network that provides distributed data storage and delivery. IPFS has millions of daily content retrievals and already underpins dozens of third-party applications. This paper evaluates the performance of IPFS by introducing a set of measurement methodologies that allow us to uncover the characteristics of peers in the IPFS network. We reveal presence in more than 2700 Autonomous Systems and 152 countries, the majority of which operate outside large central cloud providers like Amazon or Azure. We further evaluate IPFS performance, showing that both publication and retrieval delays are acceptable for a wide range of use cases. Finally, we share our datasets, experiences and lessons learned.

CCS CONCEPTS

• **Networks** → **Network protocol design**; *Network measurement*; *Naming and addressing*; *Network performance analysis*; • **Computer systems organization** → **Peer-to-peer architectures**; • **General and reference** → Measurement; Design; Evaluation;

SIGCOMM '22, August 22–26, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9420-8/22/08.
https://doi.org/10.1145/3544216.3544232

KEYWORDS

Interplanetary file system, content addressing, decentralized web, libp2p, content addressable storage

ACM Reference Format:
Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. 2022. Design and Evaluation of IPFS: A Storage Layer for the Decentralized Web. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3544216.3544232

1 INTRODUCTION

Economies of scale and technical innovations, such as cloud computing, have led to a growing centralization of web systems [2]. For example, recent trends in name resolution, content hosting, routing [3, 5], protocol development [24, 31, 42] and certificate authorities all point towards the consolidation of ownership and operation [47]. An administrator establishing a new website will likely co-locate their server on a cloud platform such as Amazon EC2; utilize a third-party DNS provider such as GoDaddy; serve their content via a Content Delivery Network like Akamai; and rely on certificates issued by Let's Encrypt. Although each of these services is well-engineered and highly performant, they nevertheless represent single points of organizational failure. In the most extreme cases, such players have gained near-monopoly status and triggered widespread chaos during outages (*e.g.,* OVHcloud, Cloudflare, AWS) [11, 40, 52]. The monetary costs incurred during outages are enormous, with Amazon's eCommerce platform reportedly losing over $66,000 per minute during an outage in 2013 [8].

In response to this, there has been a growing movement, colloquially referred to as the "*Decentralized Web*". This encompasses an array of technologies that strive to provide greater control for users. These technologies tend to rely on open-source, community-led software implementations that decentralize traditional web functionality (*e.g.,* name lookup, hosting, certification), such that no individual administrative entity could hamper overall operations or design decisions. A number of successful projects have already deployed decentralized systems that offer commonly used services, *e.g.,* Mastodon for micro-blogging or PeerTube for video sharing. However, at the core of any web platform is storing and serving media objects at scale. We argue that by decentralizing these core functions, many other applications could readily be built atop without needing to handle the complexity of decentralization themselves.

The *InterPlanetary File System (IPFS)* project aims to achieve this: it is an entirely decentralized content-addressable media object storage and retrieval platform. IPFS is a community-driven, open source effort, which is vital for ensuring community buy-in and creating an open platform for design innovation. IPFS covers 176 git repositories, across which there have been 60.4 k commits by 1185 code contributors, covering 400+ organizations including universities, start-ups and large corporations. This paper reports on our experiences in *Protocol Labs*, driving forward the IPFS effort. Protocol Labs is the largest supporter of the IPFS project, employing or funding most of the full-time contributors. Protocol Labs is also the largest contributor to the open-source codebase, covering 62 % of git commits and 75.4 % lines of code. It is worth noting, however, that large codebase decisions and roadmap setting is led through a public voting process. Thus, we emphasize that the design and implementation work reported in this paper stems from countless community contributions.

IPFS is seeing widespread uptake with more than 3 M web client accesses and beyond 300 k unique nodes serving content in the peer-to-peer (P2P) network every week. IPFS currently underpins various other Decentralized Web applications, including social networking and discussion platforms (Discussify, Matters News), data storage solutions (Space, Peergos, Temporal), content search (Almonit, Deece), messaging (Berty), content streaming (Audius, Watchit), and e-commerce (Ethlance, dClimate) [25]. Support for accessing IPFS has further been integrated into mainstream browsers such as Opera and Brave, allowing widespread and easy uptake.

In this paper, we present the design and implementation of IPFS. At its core, IPFS relies on four main concepts: (*i*) *Content-based addressing*: unlike HTTP, IPFS detaches object names from host location — enabling objects to be served from any peer; (*ii*) *Decentralized object indexing*: IPFS relies on a decentralized P2P overlay for indexing all available locations from which objects can be retrieved reducing the impact of technical or organizational failure; (*iii*) *Immutability and self-certification:* IPFS relies on cryptographic hashing to self-certify objects, removing the need for certificate-based authentication, hence, providing verifiability; and (*iv*) *Open participation:* anybody can deploy an IPFS node and participate in the network without requiring special permissions or privileges.

The contributions of this paper are as follows:

(1) We present the design and implementation of IPFS (Section 2), detailing how it publishes (Section 3.1) and retrieves (Section 3.2) content at scale.
(2) We propose three complementary measurement methodologies that provide vantage into the deployment, usage and performance of the IPFS network (Section 4). This is vital due to the decentralized nature of IPFS. As no individual entity operates the entirety of IPFS, we use these techniques to quantify IPFS across a number of dimensions. We make our datasets and tooling publicly available.
(3) We utilize the above methodologies to evaluate the deployment success of IPFS (Section 5). We find that IPFS infrastructure has been deployed in over 2700 Autonomous Systems, across 464 k IP addresses. This covers 152 countries, with the majority hosted in the US and China. We further observe widespread usage by clients with 7.1 million content retrievals seen from a single vantage point on one day alone.
(4) We finally present a performance evaluation of IPFS (Section 6). We show that, although content retrievals in IPFS are slower than direct HTTP access, delays are still reasonable for a number of use cases. For example, 3/4 of retrievals from Europe are under 2 seconds. This includes looking up the content host and fetching a 0.5 MB file. To improve performance, we show how the introduction of gateway caching can substantially reduce retrieval latency with 76 % of requests being served in under 250 ms.

2 IPFS FUNDAMENTALS

We start by providing an overview of the core building blocks of IPFS. Namely, how IPFS (*i*) addresses content; (*ii*) addresses peers; and (*iii*) indexes content, to enable distributed lookups that map content identifiers to peers hosting the object.

2.1 Content Addressing

At the core of IPFS is a content-based addressing scheme using unique hash-based **Content Identifiers (CIDs)**, similar to BitTorrent [10] or Content-Centric Networking [71] (see related work in Section 7). CIDs are the base primitive that decouple a name for content from the storage location. In contrast, location-based systems, such as HTTP, bind content addresses (URLs) to their primary host. This fundamental design decision enables the decentralization of content storage, content delivery, and address management. In addition, by decoupling the content address from its storage location, CIDs prevent vendor lock-in and remove the need for central authorities to handle address allocation.

Figure 1 shows an example of a CID and its structure. It relies on a set of self-describing data representation protocols [43] and is composed of the following four fields:

Multibase prefix: Indicates one of the 24 currently supported base-encodings with which the binary CID has been encoded ("b" for base32 in Figure 1).

CID-Version identifier: Indicates the CID version (v1). Currently, two versions exist (v0 and v1).

Multicodec identifier: Specifies how the addressed data has been encoded (protobuf, json, cbor, etc.).

Multihash: A self-describing hash-digest of the addressed data. The Multihash includes metadata indicating the hash function used (default sha2-256) and the length (default 32 bytes) of the actual content hash. The term **Multi**hash stems from the fact that it can support any hashing algorithm.

When content is added to IPFS, it is split into chunks (default 256 kB) each of which is assigned its own CID. The CID of each chunk results from hashing its content and adding the above metadata. Once all chunks have a CID, IPFS constructs a **Merkle Directed**

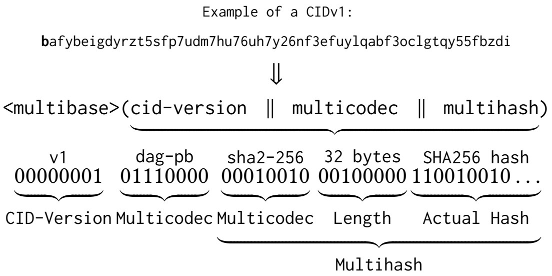

Figure 1: Structure of a CID.

Acyclic Graph (DAG) of the file [35]. This Merkle DAG is the form in which the file is provided by the original content publisher. A Merkle-DAG is a data structure similar to a Merkle-tree but without balance requirements. The root node combines all CIDs of its descendant nodes and forms the final content CID (commonly called *root CID*). In Merkle-DAGs, a node is allowed to have multiple parents, an important property that allows for chunk de-duplication. In turn, content de-duplication means that the same content does not need to be stored or transmitted twice, saving both storage and bandwidth resources. Further, Merkle DAGs are agnostic to where the content is stored. Thus, they do not need to be updated when a file is replicated on or deleted from nodes in the network.

Thanks to their hash-based structure, CIDs are immutable and self-certifying, *i.e.,* content cannot be altered without modifying its CID. This enables self-verification by comparing the CID with the hash of the content itself. Clearly, this property becomes a challenge for dynamically changing digital objects, which we address in Section 3.3.

2.2 Peer Addressing

Upon joining the IPFS network by connecting to a set of canonical bootstrap peers, peers generate a public-private key pair. Every peer in the IPFS network is identified by its unique **PeerID**, which is the hash of its public key (represented as a *Multihash*). The PeerID remains the same, unless the node operator chooses to change it manually. When establishing a secure communication channel, the PeerID is used to verify that the public key used to secure the channel is the same as the one used to identify the peer.

In order to represent the locations of remote peers, IPFS relies on **Multiaddresses**. A Multiaddress is a self-describing, human-readable, hierarchically-separated sequence of protocol choices. The term **Multi**address stems from the fact that the format allows multiple protocols and address types to be included. Each Multiaddress describes an endpoint enabling a peer to be interacted with. IPFS encompasses multiple protocols, from the network layer up to the application layer.

Figure 2 presents the structure of a Multiaddress, showing the network and transport protocols for the communication (`IPv4` and `TCP`) their corresponding location-based address information (IP address `1.2.3.4` and TCP port number `3333`) followed by the protocol to address one particular peer (`p2p`) and its PeerID (`QmZyWQ14...`). As a result, Multiaddresses point to remote processes by encoding multiple layers of addressing information into a path representation. A Multiaddress uses this construct for two reasons. First, not all IPFS nodes share the same subset of protocols. Multiaddresses allow nodes to know if they will be able to connect to a remote peer before attempting the connection. Second, the extensible syntax of Multiaddresses allows for intermediate relaying of communication through prefixing peer addresses. This is used to proxy messages to in-browser nodes that cannot be directly contacted.

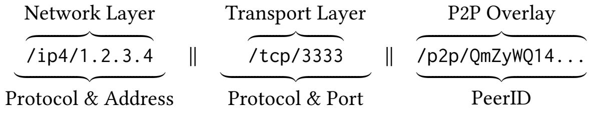

Figure 2: Structure of a Multiaddress.

2.3 Content Indexing

To publish or retrieve an object, it is necessary to create a mapping between a CID and a PeerID that can provide the object (including its Multiaddress). In order to operate in a decentralized fashion and support content and peer discovery, these mappings are indexed on a Distributed Hash Table (DHT), which exposes simple `PUT` and `GET` primitives. IPFS's DHT is based on Kademlia [41], similar to that used by the BitTorrent Mainline DHT [10, 19]. CIDs and PeerIDs reside in a common 256-bit key space by using the SHA256 hashes of their binary representations (see Figure 1) as indexing keys.

Based on our practical experiences with the live network, we have made a number of tweaks compared to the original Kademlia specification. Nodes in the DHT use 256-bit SHA256 keys instead of the 160-bit SHA1 keys. This is to anticipate advances in deliberate hash collisions [67]. We also maintain $i = 256$ buckets of k−nodes each (where $k = 20$) to split the hash space. Finally, we employ reliable transport protocols such as TCP and QUIC (instead of UDP) [41], as this makes connection management in the implementation more straightforward.

New peers join the DHT as either *DHT Servers*, if they have public IP connectivity, or as *DHT Clients*, if they are not publicly reachable, *e.g.,* because they are behind a Network Address Translation (NAT) device. IPFS differentiates between DHT clients and servers through a simple technique called *Autonat* [33]. Autonat works as follows: new peers join by default as clients and immediately ask other peers in the network to initiate connections back to them. If more than three peers can connect to the newly joining peer, then the new peer upgrades its participation to act as a server node. If more than three peers cannot connect, the peer continues as a client. DHT Servers perform all network operations, *i.e.,* storing content, storing mapping records, and providing these to requesting peers. In contrast, DHT Clients only request records or content from the network but do not store or provide any of them. The DHT client/server distinction prevents unreachable peers from becoming part of other peers' routing tables, thus speeding up the publication and retrieval processes.

3 IPFS IN ACTION

In this section we explain how content is published (Section 3.1), and how other peers are able to find and retrieve it (Section 3.2). Both processes are depicted in Figure 3 and marked throughout this section. Then in Section 3.3, we describe how IPFS deals with mutable content.

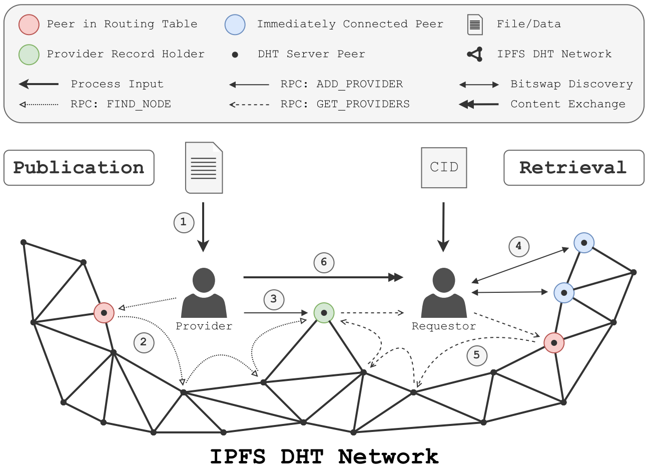

Figure 3: IPFS publication and retrieval. Publication involves: ① Import content to *Providers* local IPFS process and allocate CID ② DHT Walk to find closest peers to CID ③ Store provider record with closest peers. Retrieval involves: ④ Opportunistic Bitswap requests to already connected peers for CID ⑤ DHT Walk to find a provider record storing peer ⑥ *Requestor* connects to *Provider* and fetches the content. The diagram omits *Requestors* second DHT Walk to resolve *Providers* PeerID to their network address.

3.1 Content Publication

To make content available in the IPFS network, the content is first imported to IPFS and allocated a CID ① (see Section 2.1).

After content has been imported into the local IPFS instance, it is neither replicated nor uploaded to any external server. To publish it, the host generates a *provider record* and pushes it into the DHT. This record maps the CID to its own PeerID. The provider record is stored on the $k = 20$ closest peers in terms of their PeerIDs' XOR distance [41] from the SHA256 hash of the CID. The record is replicated on k peers to ensure that records remain available even if some of those k peers leave the network. 20 is selected based on our practical experiences (as well as a recommendation in the original Kademlia specification [41]), serving as a compromise between excessive replication overhead and risking record deletion because of peer churn. We explain how the 20 closest peers are discovered in Section 3.2.

Once IPFS has found the closest peers ②, it attempts to store the provider record with them. It does so by establishing a network connection and then initiating an RPC ③. The process does not wait for a response from each peer but will instead perform the RPCs in a "fire and forget" fashion which will become relevant in the performance evaluation (Section 6). Note, any peer that later retrieves the data becomes a temporary (or permanent, if they so choose) content provider themselves by publishing a provider record pointing to their own node to the DHT. Peers retrieving the content do not need to trust the new providing peer but only verify that the data they were served matches the requested CID.

A peer must also publish its *peer record*, which maps its own PeerID to any Multiaddresses associated with it. This is used by requesting peers to discover the underlying network address of the peer. Publication of the peer record follows the same CID-to-PeerID procedure as described above and happens independently of any content publication.

Apart from the replication factor ($k = 20$), provider records are associated with two other parameters: (*i*) the *republish interval*, by default set to 12 h, to make sure that even if the original 20 peers previously responsible for keeping the provider record go offline, the provider will assign new ones within 12 h; and (*ii*) the *expiry interval*, by default set to 24 h, to make sure that the publisher has not gone offline and is still willing to serve the content. These settings aim to prevent the system from storing and providing stale records. It is worth noting that peers behind NATs cannot host content themselves. Thus, third party hosts, commonly called *pinning services* are used to publish content on behalf of NAT'ed end-users (usually for a fee). Although a NAT hole-punching solution is currently being developed [34], it is still under-test.

3.2 Content Retrieval

Once the provider and peer records have been published, users can retrieve the content. To retrieve the content, the requesting peer performs four steps: (*i*) *Content discovery:* identify PeerID(s) that host the content/CID; (*ii*) *Peer discovery:* map the PeerID to a Multiaddress, *e.g.*, an IP address; (*iii*) *Peer routing*: connect to the peer; and (*iv*) *Content exchange:* fetch the content.

Content Discovery. Content discovery in IPFS is done primarily using the DHT (see Section 2). However, before entering the DHT lookup, the requesting peer asks all peers it is already connected to for the desired CID ④. This is done (using the Bitswap protocol, discussed later in this section) in an opportunistic fashion. It allows a node to resolve content faster in case a peer's immediate neighbours store the desired content. If that initial attempt is not successful, content discovery falls back to the DHT with a timeout of 1 second.

The DHT implements *multi-round iterative lookups* in order to resolve a CID to a peer's Multiaddresses, a process we refer to as a *DHT walk* ⑤. When peer A issues a request for CID x, the request is forwarded to $\alpha = 3$ nodes whose PeerIDs are closest to x in peer A's routing table (as per the original Kademlia specification [41]). The peers receiving those requests reply with the requested content if they have it. If they do not have the requested content, they reply with either the provider record that points to the PeerID that holds the requested item together with the peer's Multiaddress (if they have it), or with the peers it knows of whose PeerID is closer to x. The process continues until the node is returned with the PeerID that has previously declared to hold a copy of the requested CID through a published provider record.

Peer Discovery. After the content discovery phase, a client knows the PeerID(s) hosting the desired content. As mentioned earlier, PeerIDs need to be mapped to a physical network address by looking up the peer record. This procedure is called *Peer Discovery* and is carried out by querying the DHT for a second time.

To further streamline the process, each IPFS node maintains an address book of up to 900 recently seen peers. Nodes check whether they already have an address for the PeerID they have discovered before performing any further lookups.

Peer Routing. Once the PeerID is resolved to a peer record, the requesting node will possess the Multiaddress(es) of the peer that appears to have the content. It therefore uses the list of addresses to connect to the desired peer.

Content Exchange. As provider peers are identified, content fetching is done using *Bitswap*, a simple chunk exchange protocol ⑥. Bitswap issues requests for the content items in *wantlists*. Requests are sent using an `IWANT-HAVE` message. Recipient peers that have the block reply with a corresponding `IHAVE` message. The requesting peer finally responds with an `IWANT-BLOCK` message. Receipt of the requested block terminates the exchange. Recall, the Bitswap protocol is also used to *discover* content available on nearby neighbours opportunistically. The full details of the Bitswap protocol design as well as a number of proposed optimizations can be found in [14, 50].

3.3 Content Mutability

Given their hash-based structure, CIDs are permanent, immutable, and self-certifying. This makes them ideal for decentralizing namespace management, as it avoids the need for a central coordinator. Furthermore, their immutability and self-certification make universal caching (*i.e.,* from any peer) possible. However, this becomes problematic for handling dynamic content, *e.g.,* evolving text documents. In order to cope with mutable content, IPFS provides the option of publishing content based on the hash of the publisher's public key (*i.e.,* PeerID) instead of the hash of the content (*i.e.,* CID) itself. Those, so called *InterPlanetary Name System (IPNS) records*, map the CID of the publisher's public key to another CID signed by the corresponding private key. This way, content can be updated and obtain a different CID, but an immutable reference (*i.e.,* the CID of the publisher's public key) is created and used. As this mechanism makes use of additional constructs, we leave further details out and point the reader to [15, 55] for more details.

3.4 IPFS Gateways

To broaden access to IPFS-hosted content for users who have not installed IPFS software, the IPFS system is complemented with a *gateway* model. Gateways offer (HTTP) entry points into IPFS, enabling users who do not run any IPFS software to access content. Our gateway implementation acts as a bridge: on one side is a DHT Server node, and on the other side is an nginx HTTP web server, which can receive `GET` requests containing the CID as the URL path, *i.e.,* `https://ipfs-gateway.io/ipfs/{CID}`.

By embedding caching within the gateways, we further streamline performance by aggregating user demand. Each gateway server runs two forms of content storage: (*i*) the default nginx web cache, with a Least Recently Used replacement strategy; and (*ii*) The IPFS node store, which holds content manually uploaded by the Web3 and NFT Storage Initiatives.[1] These allow third parties to pin content in the IPFS store of the gateway to make it persistently available. We emphasize that the gateways are entirely optional for the operation of the overall storage and retrieval network and are only used for content retrievals through the browser.

Protocol Labs operates two major gateways, and in total, there are 107 known gateways.[2] Note, operating a gateway does not require authorization or permission by any entity, but in order for it to be useful for the network, it needs to be set up with a public IP address.

4 EVALUATION DATA

We next present the data we use to evaluate IPFS as a system. Due to its decentralized nature, it is challenging to record activities across all IPFS nodes. Particularly, since independent node operators dominate IPFS, no complete record exists. To address this challenge, we compile three datasets comprising a mix of active and passive measurements. Figure 4a presents a time series of our measurement periods, which we describe in more detail below.

4.1 Peer Data

Our first dataset covers information about peers acting as DHT Servers. As there is no central repository of such information, we employ active measurements to gather this data. We implement a crawler [63] to gather a comprehensive list of all peers that are engaged in the DHT. We run the crawler from a server in Germany every 30 minutes. The crawler recursively asks peers in the network for all entries in their k-buckets starting from the six well-known default IPFS bootstrap peers until it finds no new entries. The procedure that yields the list of all k-bucket peers resembles previous work [23].

We started our measurements on 2021-07-09 and upgraded it on 2021-09-24 to collect the association of single peers with their Multiaddresses, agent version, supported protocols, and connection, handshake and crawl duration. We then map all IP addresses to their country and Autonomous System (AS) using GeoLite2, and tag it with its CAIDA AS Rank [48]. In total, we have performed over 9500 network crawls. Figure 4a plots the number of peers we observed each day.

To quantify peer uptime, we periodically revisit all previously discovered and online peers and measure their session lengths (defined as their distinct, continuous time periods online). Due to the scale, we adapt the probe frequency based on how often we observe a peer to be accessible. Specifically, we select an interval of 0.5x the observed uptime, starting at a minimum of 30 seconds and ending at a maximum of 15 minutes. This is because it is more likely for peers to stay online if we observe them to have been online for an extended time period. We make our crawl data available for further research on IPFS with the CID:

`bafybeigkawbwjxa325rhul5vodzxb5uof73neszqe6477nilzziw5k5oj4`

4.2 IPFS Gateway Usage Data

Our second dataset covers all the `GET` requests taken from a public IPFS Gateway run by Protocol Labs (`ipfs.io`), shedding light on large-scale usage patterns of IPFS. This data represents a geographic subset of total gateway traffic, as the sampled traffic comes from one of several gateway instances which load balance inbound traffic using anycast. This particular gateaway is located in the US.

The dataset covers one day of access in January 2022 with 7.1 M user requests. Each entry maps to a single user request and response

[1] https://web3.storage/, https://nft.storage

[2] https://ipfs.github.io/public-gateway-checker/

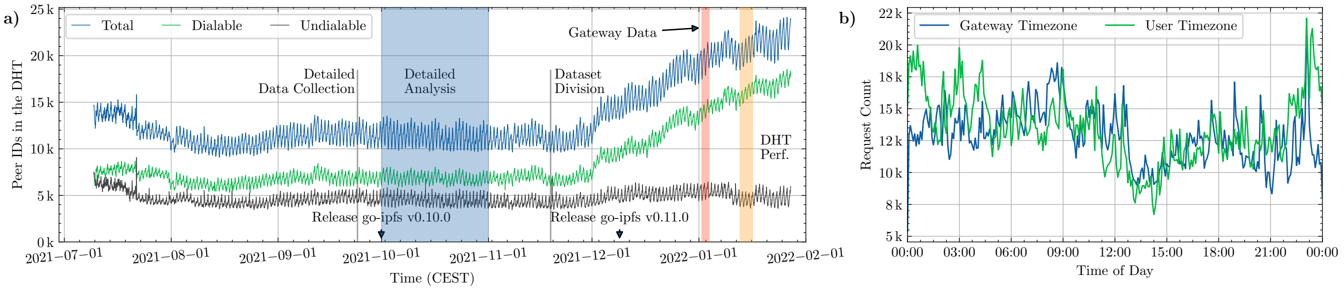

Figure 4: a) Total number of crawled peers over time and their fraction of dialable and undialable peers (one-day periodicity). The graph also depicts several events and measurement time periods which help to assess our measurement results in later sections. b) Request count of a single gateway on 2022-01-02.

Table 1: Number of publication and retrieval operations from each AWS region.

AWS Region	Publications	Retrievals
af_south_1	547	2, 047
ap_southeast_2	547	2, 630
eu_central_1	547	2, 708
me_south_1	547	2, 112
sa_east_1	546	2, 363
us_west_1	547	2, 704
Total	3, 281	14, 564

information to the gateway comprising of request timestamp, user agent, HTTP referrer, (Maxmind-located) city, response sizes, and cache hit/miss information. We aggregate users by unique combinations of IP and user agent. Overall, we find 101 k users accessing 274 k unique CIDs during the day of data collection. For context, Figure 4b shows the number of requests received (5 minutes bins) at the gateway, based on the timezone of the gateway (Pacific Standard Time) as well as the geolocated user timezone. We make the access logs available on IPFS with the CID:

`bafybeiftyvcar3vh7zua3xakxkb2h5ppo4giu5f3rkpsqgcfh7n7axxnsa`

4.3 Performance Data

Our third dataset focuses on benchmarking content publication and retrieval performance. We use six virtual machines in six different regions on AWS. Namely, the `t2.small` machines run in `me_south_1` (Bahrain), `ap_southeast_2` (Sydney), `af_south_1` (Cape Town), `us_west_1` (N. California), `eu_central_1` (Frankfurt) and `sa_east_1` (São Paulo). On each machine we run a `go-ipfs v0.10.0` instance acting as a DHT server node. We use these controlled instances to interact with the public IPFS network and perform tailored performance experiments. The measurements were conducted during the shaded time period in Figure 4a labeled "DHT Perf".

Upon each iteration, a single node announces a new 0.5 MB object (*i.e.,* CID) to the network. Following this, all other nodes retrieve the object. This involves looking up the provider and peer records, connecting to the providing peer and then downloading the object. As soon as all remaining nodes have completed this process, they disconnect to prevent the next retrieval operation being resolved through Bitswap and instead resort to the DHT for lookup and discovery. It is worth noting that this is the closest one can get to a controlled experiment in the public IPFS network. This is because it is very difficult to replicate peers' behaviour (*e.g.,* churn, CPU and traffic load) in a simulation environment. Table 1 lists the number of publications and retrievals we have performed from each region. The varying numbers of publications and retrievals stem from terminating the experiment before the instance in `sa_east_1` finished its latest publication and missed instructions on our control plane respectively. This does not affect the correctness of our measurement but only influences the statistical significance of the results below. The data alongside analysis code is published on IPFS with the CID:

`bafybeid7ilj4k4rq27lg45nceq4akdpetav6bcujgiym6vch5ml24tk2t4`

4.4 Ethical Considerations

The *Peer* and *Performance* datasets raise limited ethical concerns. They involve collecting IP addresses, yet we do not attempt to map these back to personal identities, as such analysis was not within the scope of this study. Furthermore, after performing the geolocation analysis, we have anonymized the datasets that we have made available. The *IPFS Gateway Usage Data* contains personal information, as it covers requests from web clients. This information is collected as part of our routine operations, and in line with IPFS' policies.[3] Further, we do not trigger extra data collection. We anonymize IP addresses, and do not perform lookups on the CIDs to infer the nature of the content exchanged.

5 DEPLOYMENT SCALE

We now measure the scale of the IPFS network by looking at peers, physical machines, and gateway users. Unless noted otherwise, we limit our analysis to the shaded time period labeled "Detailed Analysis" in Figure 4a.

5.1 Geographical Distribution

Geographical Distribution of Peers. Using the Peer dataset, we discover a total of 198,964 IPFS peers (as identified by their PeerID) with 1,998,825 Multiaddresses in the DHT, covering 464,303 unique IP addresses from 152 countries. Of these IP addresses, we were able to establish a connection to 253,198 (54.5 %) at least once while 211,105 (45.5 %) were always unreachable.

[3] https://docs.ipfs.io/concepts/privacy-and-encryption/

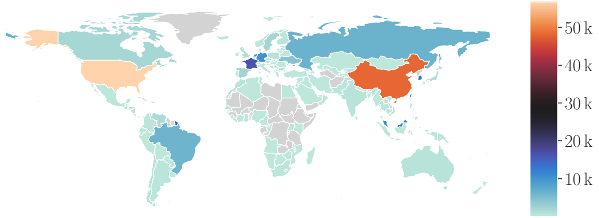

Figure 5: Geographical distribution of peers. "Multihoming" peers were counted repeatedly.

Figure 5 shows the geographical distribution of PeerIDs. Although we see widespread uptake of IPFS, we observe a high concentration in certain regions. The US (28.5 %) and China (24.2 %) dominate the share of peers, followed by France (8.3 %), Taiwan (7.2 %), and South Korea (6.7 %). In addition, "Multihoming" is commonplace: around 8.8 % of all peers advertise Multiaddresses that include multiple IP addresses mapped to multiple countries.

To better understand the deployment in these different regions, Figure 7a presents the distribution of peers with >90 % uptime ("reliable"), whilst Figure 7b presents the distribution of peers that were unreachable during our entire measurement period. We found 1.4 % (2747) of observed peers to be "reliable", whereas around 1/3 of peers are never accessible. The relatively even spread of countries gives us confidence that no individual region could disrupt IPFS single-handedly, without significant strategic effort. For example, if a single country hosted the majority of IPFS nodes, it could become possible for the state to unduly influence the wider network, which does not seem to be the case. The distribution of reliable peers is particularly egalitarian, with the largest player (US) hosting just 0.3 % of peers. We see similar trends for unreachable peers. Although China does contain a large share (12.5 %), these unreachable peers naturally have less impact on the overall system.

Note that a single IP address can host multiple PeerIDs. Figure 7c plots a CDF of the number of peers per host. Although the majority (92.3 %) of IP addresses host a single PeerID, we find that the top 10 IP addresses host almost 66 k distinct PeerIDs. This raises concerns about potentially misbehaving peers that rotate their PeerIDs. Such peers would be able to hamper routing performance, *e.g.,* by persistently dropping requests.

Geographical Distribution of Gateway Users. Using the gateway usage dataset, we also inspect the geographical distribution of incoming requests to get an idea of the scale of gateway usage. In total, we observe requests from 59 countries. Figure 6 shows their geographical distribution. Since the sampled node was located in the US, we find more than three-quarters of user requests to the gateway come from the US (50.4 %). This is followed by China (31.9 %), Hong Kong (6.6 %), Canada (4.6 %), and Japan (1.7 %).

5.2 Autonomous System Distribution

Next, we investigate the Autonomous System (AS) coverage of the IPFS deployment and assess potential centralization in ASes or associated cloud providers.

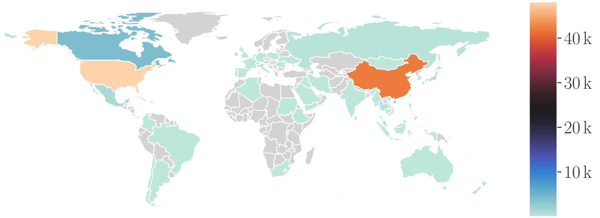

Figure 6: Geographical distribution of users requesting content via the Gateway.

Autonomous System Coverage. In total, we observe peers in 2715 unique ASes. Figure 7d presents the number of IP addresses in each AS. We rank ASes (on the x-axis) by their CAIDA AS Rank. Unsurprisingly, we see a small set of highly ranked ASes containing many IPFS hosts. The top 10 ASes contain 64.9 % of IP addresses, whereas the top 100 contain 90.6 %. This is rather different from our prior observations related to geography, which show more egalitarian trends. Although there is a wide geographic spread, we find that each region is dominated by a small number of ASes. To explore this, Table 2 presents the number of IP addresses observed in the top ASes. Although we observe that IPFS is deployed in a large number of ASes, >50 % of IPs found are in just 5. ASes in China stand out in this regard, with two Chinese ASes containing >30 % of all observed IP addresses. This, again, raises concerns over the concentration of peers in specific networks. However, we argue that the widespread deployment across 2715 ASes, together with the decentralized structure of the P2P system (*i.e.,* with regard to indexing and resolution), would ensure resilience even in cases of severe network fragmentation, or targeted disruption attempts.

Cloud Coverage. Another potential cause of the above trends is the presence of prominent cloud infrastructure, used by IPFS node operators. To this end, we use the Udger data set [64] to get a curated list of 1525 cloud providers and their IP ranges. Table 3 lists the number of nodes that are deployed in each one. Contrary to expectations, we see that just a minority (<2.3 %) of IPFS nodes are hosted in cloud infrastructure. This figure is in stark contrast to other decentralized web platforms such as Mastodon, where 6 % of the infrastructure is hosted on Amazon alone [47]. This is an important finding for a decentralized storage and delivery network, and suggests that the majority of users host their own deployments.

5.3 Churn

Churn refers to the action of peers arriving and departing from the network. The churn rate within a system like IPFS is important as it influences decisions such as for how long should peers retain information about other (online) peers. To calculate churn from our Peer dataset, we follow the method used in [51, 56, 60] for long session handling to minimize bias towards shorter sessions and account for peers that stay online beyond our selected measurement time window. Figure 8 plots CDFs of DHT peer uptimes for countries that stood out in the deployment analysis of Section 5, based on 467,134 session observations that started in the first half of our "Detailed Analysis" time window (see Figure 4a).

We observe that uptime tends to be short, with 87.6 % of sessions under 8 hours and only 2.5 % of sessions exceeding 24 hours. This

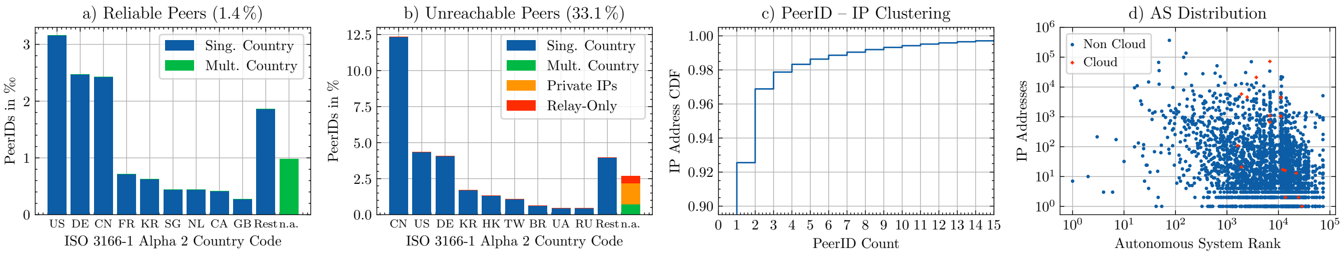

Figure 7: a) Geographical distribution of peers that were reachable for >90 % of the measurement period. Note the ‰ unit on the y-axis. b) Geographical distribution of peers that were always offline and not reachable at all during the measurement period. c) CDF of the number PeerIDs per IP address d) Distribution of IPs across ASes according to their size (measured by their AS rank).

Table 2: Autonomous systems covering >50 % of all found IP addresses.

Share	ASN	Rank	AS Name
18.9 %	4134	76	CHINANET-BACKBONE No.31,Jin-rong Street, CN
12.8 %	4837	160	CHINA169-BACKBONE CHINA UNICOM China169 Back., CN
9.6 %	4760	2976	HKTIMS-AP HKT Limited, HK
6.9 %	26599	6797	TELEFONICA BRASIL S.A, BR
5.3 %	3462	340	HINET Data Communication Business Group, TW

Table 3: Percentage of nodes hosted on cloud providers. The table shows the top ten and selected cloud providers.

Rank	Provider	IP Addresses	IP Address Share
1	Contabo GmbH	2038	0.44 %
2	Amazon AWS	1792	0.39 %
3	Microsoft Azure/Coporation	1536	0.33 %
4	Digital Ocean	836	0.18 %
5	Hetzner Online	592	0.13 %
6	GZ Systems	346	<0.10 %
7	OVH	341	<0.10 %
8	Google Cloud	286	<0.10 %
9	Tencent Cloud	258	<0.10 %
10	Choopa, LLC. Cloud	244	<0.10 %
12	Alibaba Cloud	180	<0.10 %
13	CloudFlare Inc	140	<0.10 %
27	Oracle Cloud	27	<0.10 %
54	IBM Cloud	9	<0.10 %
	235 Other Cloud Providers	2017	0.43 %
	Non-Cloud	453,661	97.71 %

helps explain our design decision to replicate records on a relatively large number of peers ($k = 20$, see Section 2.3). Briefly, we also note that stability varies greatly based on region. For example, whereas the median uptime for Hong Kong is just 24.2 min, it is more than double that figure for Germany.

5.4 Discussion & Takeaways

IPFS is designed to be highly decentralized. Although we observe geographical agglomeration in certain regions, we also see IPFS being widely adopted around the globe. Especially important is the finding that fewer than 2.3 % of IPFS nodes run in major cloud platforms. This suggests that individuals are running IPFS nodes on personal or on-premises commodity hardware. As a downside, this contributes to the high churn rate observed (only 2.5 % of peers stay online for more than 24 h). Further, although we find peers in 2715 ASes, the top 10 contain 64.9% of peers alone. This suggests that we should push harder to broaden deployment and avoid consolidation in a minority of ASes.

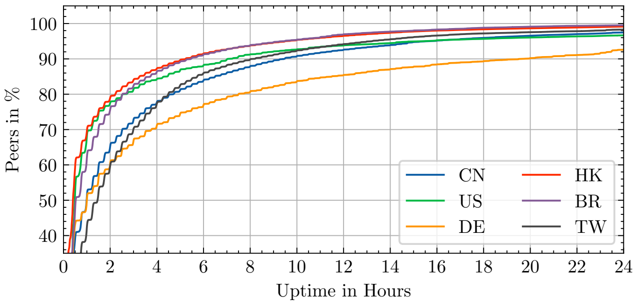

Figure 8: Churn rate by region for 467,134 session observations. Each line shows the cumulative distribution function of the DHT peer uptimes in a given region. The step shape correlates with the sampling interval of our crawler.

6 IPFS PERFORMANCE EVALUATION

In this section, we evaluate the performance of IPFS' two core functions: content publication and content retrieval. We also turn to our Gateway dataset to better understand IPFS's usage through browsers.

6.1 Content Publication Performance

As explained in Section 3.1, the content publication process consists of two steps. First, the content is imported into the local IPFS node and second, provider records are stored with suitable peers. The first (*i.e.,* importing content) step was covered in [1]. Here, we focus on the publication of provider records to the network.

Overall Delay. Figure 9a shows CDFs for the duration of the overall content publication process. We present separate results for each AWS region from which the measurements are launched. The overall publication process across all regions takes 33.8 s, 112.3 s, and 138.1 s in the 50th, 90th, and 95th percentiles, respectively. Note that the publication delay is independent of the content size as only the provider record is being published. Table 4 breaks down the publication duration percentiles for each AWS region individually.

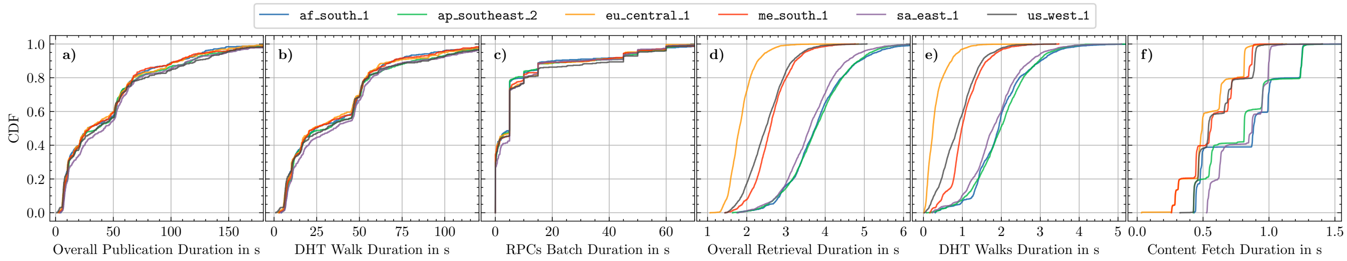

Figure 9: CDFs for content publication (a–c) and content retrieval (d–f) for each AWS region that we employed for our lookup measurements. a) The overall publication duration includes b) walking the DHT, and c) the batch of RPCs to store the provider record at suitable peers. d) The overall content retrieval duration includes e) walking the DHT twice, and f) fetching the data from the providing peer. The sample size is 4324 in all graphs across all AWS regions combined.

Table 4: Latency percentiles of the overall DHT publication and retrieval operations from different AWS regions.

	Publication Percentiles			Retrieval Percentiles		
AWS Region	50th	90th	95th	50th	90th	95th
af_south_1	28.93 s	107.14 s	127.22 s	3.75 s	4.88 s	5.31 s
ap_southeast_2	36.26 s	117.74 s	142.79 s	3.76 s	4.85 s	5.15 s
eu_central_1	27.70 s	106.91 s	133.27 s	1.81 s	2.28 s	2.50 s
me_south_1	29.32 s	105.45 s	130.48 s	2.59 s	3.24 s	3.48 s
sa_east_1	42.32 s	115.45 s	148.04 s	3.60 s	4.56 s	4.93 s
us_west_1	36.02 s	121.13 s	147.59 s	2.48 s	3.17 s	3.42 s

We see similar results across all regions despite the larger build-up of presence in the US and China.

Delay Breakdown. The two main constituents of delay are the DHT walk (to discover the 20 closest peers to the CID) and the RPC operations that push provider records to the chosen peers. Figure 9b presents CDFs of the DHT walk duration. We see that this constitutes the largest component of the delay. On average, the DHT walk covers 87.9 % of the overall delay. Naturally, optimizing this is a target of our future work.

In contrast, Figure 9c shows the CDFs for the duration of the batch of RPCs. This reveals much faster performance, with 43.3 % of RPC batches completing in under 2 s. That said, 53.7 % exceed 5 s, and 11.3 % exceed 20 s. Closer inspection reveals a number of spikes in the distribution. These are driven by various timeout operations in the protocol. For example, the spike at 5 s is caused by dial timeouts on the transport level of the TCP and QUIC implementations, whereas the spike at 45 s is caused by the handshake timeout of the Websocket transport. These timeouts stem from less responsive peers. This points to unavoidable challenges when operating decentralized infrastructure, where the responsiveness of peers can be difficult to predict.

6.2 Content Retrieval Performance

Overall delay. Figure 9d shows CDFs for the duration of the overall retrieval operation. We observe success rate of 100 %, confirming the reliability of the system. Although we control the providing and retrieving peers for this test, this high success rate is not self-evident since retrieval operations involve many interactions with peers outside our control. However, we experience notable performance diversity, with delays on average higher than typical web page loading times. Overall, retrieval performance is much faster than publication. The content retrieval process across all regions takes 2.90 s, 4.34 s, and 4.74 s in the 50th, 90th, and 95th percentiles, respectively. We emphasize that, in contrast to HTTP, this includes the *lookup* time, whereby CIDs are mapped to eligible locations to obtain the content from (in contrast to HTTP URLs that encode the location a priori).

Table 4 further breaks down the retrieval duration percentiles for each AWS region individually. Interestingly, we observe variations across regions. For example, the median retrieval delay in central Europe is just 1.81 s, compared to 3.75 s from South Africa. These slower retrievals are in-line with prior measurements of web performance in Africa [17]. Further, recall that due to Bitswap, DHT queries are only launched after a timeout of 1s (see Section 3.2), setting a minimum baseline of 1s delay. Finding intelligent ways to minimize this initial delay is a key line of future work.

Delay Breakdown. The three main constituents of retrieval delay are (*i*) the opportunistic content discovery through Bitswap, which (if unsuccessful) times out at 1 s[4]; (*ii*) the time spent walking the DHT to find provider and peer records; and (*iii*) the content exchange operation to fetch the object. Figure 9e presents CDFs for the combination of DHT walks to discover provider and peer records. The median duration across all regions for a single DHT walk is 622 ms and, therefore, significantly faster than the DHT walk for the publication operation. This is because a retrieval DHT walk terminates after the discovery of a single record-hosting node, rather than 20 in the case of publication. More generally, it is positive to see that single DHT walks complete with sub-second latency for most regions. In fact, we see that for all regions both DHT walks (first to find the provider record and second to find the peer record) complete in less than 2 s for 50 % of the retrievals. This is in stark contrast to previous DHT deployments, where latency can even exceed a minute [12].

Finally, Figure 9f shows the content fetch duration. This includes peer routing and content download (see Section 3.2). The figure follows a step-wise pattern, with each step corresponding to a unique combination of nodes publishing and fetching the content

[4] At this point, it is also worth highlighting that due to our experiment setup, none of the requests are satisfied at the Bitswap level, hence, in all experiments presented here, retrievals include an extra 1 s for the Bitswap timeout and constitute worst-case scenarios.

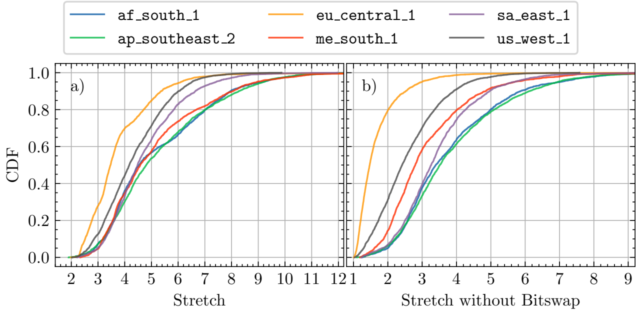

Figure 10: CDFs of the retrieval stretch (defined in equation 2) for each AWS vantage point with (a) and without (b) the initial Bitswap timeout.

(recall that each node is in a different region). Over 99 % of all content exchange operations take less than 1.26 s. We emphasize that this number is dependent on the amount of data being exchanged (which is set to 0.5 MB in our measurement setup), *i.e.,* higher data volume reduces the overall retrieval percentage spent on discovery.

Retrieval Stretch. To compare IPFS retrieval operations against HTTPS, we measure *request stretch* as the ratio of the IPFS retrieval time *vs.* the *estimated* HTTPS retrieval time (*e.g.,* retrieving an object using a URL via HTTPS):

$$\text{Stretch} = \frac{\text{IPFS Content Retrieval Time}}{\text{HTTPS Content Retrieval Time}} \quad (1)$$

$$= \frac{\text{Discover} + \text{Dial} + \text{Negotiate} + \text{Fetch}}{\text{Dial} + \text{Negotiate} + \text{Fetch}}. \quad (2)$$

In IPFS vernacular, Dial is equivalent to the TCP handshake; Negotiate is equivalent to the TLS handshake; and Fetch is equivalent to issuing the `GET` request and byte transfer. Put simply, we roughly estimate the "HTTPS Content Retrieval Time" by subtracting the DHT "Discover" latency from the "IPFS Content Retrieval Time". Figure 10a shows CDFs for the stretch of all content retrieval operations for each AWS region. The majority of content retrieval operations take at least four times as long as the equivalent HTTPS request across all AWS regions. This could be characterized as the "cost of decentralization" when using IPFS. However, recall that the "Discover" step in the IPFS case also includes the Bitswap delay mentioned in Section 3.2, which adds 1s and is always present in our experiments, due to our experimental setup. To inspect the impact of this, Figure 10b shows the stretch CDFs without the initial Bitswap delay. We note that especially for the IPFS instance in central Europe, the "cost of decentralization" is comparably low with a stretch < 2 for 80 % of content retrieval operations. This shows that, arguably, running DHT lookups in parallel to Bitswap could be superior, by trading additional network requests for faster retrieval times.

6.3 Gateway Evaluation

We finally turn to the Gateway data to inspect its performance, as well as usage.

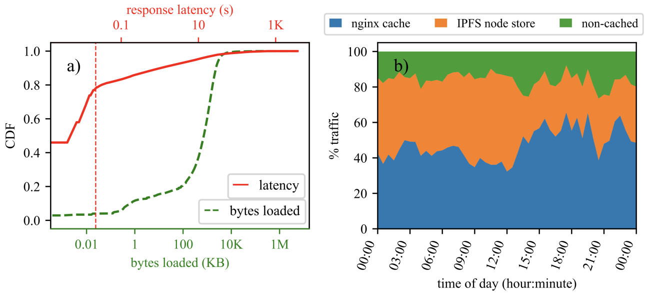

Figure 11: (a) Distribution of upstream response latency at gateway (top x-axis) and bytes downloaded per request to the gateway from peers (bottom x-axis). Latencies incurred for serving non-cached requests are present to the right of dotted red line. (b) Proportion of cached and non-cached traffic binned at every 30 min.

Table 5: Traffic and latencies to the gateway, when content is served from default nginx cache, the IPFS node store, and from other peers in the IPFS network.

	nginx cache	IPFS node store	Non Cached
Latency (Median)	0 s	8 ms	4.04 s
Traffic Served	46.4 %	38.0 %	15.6 %
Requests Served	46.0 %	40.2 %	13.8 %

Object Size. The gateway dataset allows us to obtain a sample of content objects being requested from clients. This provides vantage on the size distribution. In total, we observe 6.57 TB being downloaded during our measurement period. Figure 11a (x_1-axis) presents the distribution of object sizes (bytes loaded). We see a wide variety. The majority (79.1 %) are above 100 KB, with a median of 664.59 KB. We also find there is no correlation between the object size and the latency (Pearson correlation coefficient of 0.13), again confirming that delays are primarily driven by size-agnostic operations (*e.g.,* DHT walks).

Retrieval Delays. The gateway further provides a second source of IPFS retrieval latency measurements. Each `GET` request log entry is accompanied by the delay it took the gateway to retrieve the object (either from IPFS or its cache). Figure 11a (x_2-axis) plots the distribution of retrieval latencies (as defined in Section 6.2).

46 % of fetches have a retrieval delay of 0, which indicates a default nginx cache hit. The remaining 54 % enter IPFS via the peer bridge co-located on the gateway. 67.1 % of these are served from the local IPFS node store itself, resulting consistently in a delay below 24ms (as marked by the vertical line in the figure). Recall, this store contains content manually uploaded by the Web3 and NFT Storage Initiatives. Due to the efficacy of this caching, 76 % of requests are delivered in less than 250 ms. This is substantially better than observed from our prior measurements (see Section 6.2) and exemplifies a benefit of aggregating demand at the gateways.

Cache Hit Rates. Figure 11b presents the fraction of cache hits *vs.* misses across the one day period in our dataset. Table 5 summarizes the key performance results. Hit rates in the nginx cache vary from

32.3 % (at 00:00) to 65.6 % (at 17:30). These hits serve close to 50 % of the total traffic. When combined with pinned content stored in the gateway's local IPFS node, hit rates exceed 80 %. This indicates that the gateway offloads substantial traffic from the remaining IPFS network and (as shown in Table 5) reduces fetch times. That said, content that is not cached experiences slower retrieval times, although this only constitutes 15.6 % of bytes transferred. This suggests that augmenting IPFS with a gateway model *does* offer a meaningful strategy for reducing delays by aggregating demand via the cache.

Gateway Referrals. We finally inspect the HTTP referrals observed in the gateway's nginx logs. These indicate the website that redirected the web client to the gateway. Indeed, the majority of this traffic (51.8 %) is referred by third party websites, confirming the integration of IPFS with more traditional HTTP hosted sites. Interestingly, 70.6 % of this referred traffic belongs to just 72 semi-popular websites (rank 10 k–50 k based on Tranco list[5] [36]). The majority of these parent sites are hosted in the US (47.3 %), Iceland (20.0 %) and Canada (12.7 %). Manual inspection shows that these are primarily video streaming and Non-Fungible Token (NFT) related sites, as many people have chosen to store NFTs on IPFS.[6] This suggests that IPFS is receiving widespread uptake in these targeted use cases.

6.4 Discussion & Takeaways

IPFS is designed to offer publication and retrieval delays capable of supporting a range of applications. IPFS attains shorter retrieval delays compared to prior large-scale DHT deployments. For example, the median lookup latency for the Kademlia implementation in BitTorrent (both Mainline and Azureus) exceeds a minute [12], due to excessive dead nodes, as well as slower links at that time. Our observations indicate that the distinction between server and client peers (see Section 2.3) (after the v0.5 release of IPFS) has given a significant boost to the performance of IPFS, as peers avoid costly operations of attempting to punch through NATs, failing and timing out eventually.

In contrast, for half of our probes, the record lookup stage takes under one second, showing a dramatic improvement over past DHT deployments. That said, the overall IPFS content retrievals have a median stretch of 4.3 (compared to HTTPS), although we find that removing the initial Bitswap timeout can decrease the stretch to < 2 for 80 % of retrievals in well-connected areas. We argue that this makes IPFS suitable for various applications, including video on demand, file sharing and other social networking services. In comparison, publication delays are higher (median of 33.8 s), due to the need to push records onto 20 distinct peers (to protect from churn, see Section 5.3). For many applications, this delay would be acceptable (*e.g.,* publishing a movie), as exemplified by the number of applications already built on IPFS (see [25] for a non-exhaustive list). However, we acknowledge that the performance users experience over the last mile to their homes will differ from what we have observed from AWS data centres.

We have further experimented with several ways to streamline uptake and performance, including the use of gateways. Although arguably a centralized entity, we find that it can substantially speed-up retrievals with 46 % of retrievals being served via the nginx cache. Uptake is wide, with 101 k users hitting the gateway in a single day. We further note that anybody can set up their own gateway, ensuring that no single entity could emerge as a gatekeeper.

7 RELATED WORK

P2P Networks. There have been countless P2P overlay architecture proposals, including dozens of DHT structures [27, 29, 53, 58, 72], and tens of applications, *e.g.,* large-scale content delivery platforms [9], and services such as decentralized social networks [20]. Rather than devising an entirely new system, IPFS utilizes the Kademlia DHT for content indexing [41]. There have been various attempts to optimize the performance and usability of such DHTs via caching [54], network-aware peer selection [30], and parallelizing lookups [59]. IPFS builds on these and currently constitutes one of the largest deployments of peer-to-peer networks "in the wild" (alongside BitTorrent, which also uses Kademlia [9]). IPFS also strives to be censorship resistant. Approaches such as Freenet [7] and Wuala [38] have similar goals. These work by storing encrypted content across an arbitrary subset of peers. In contrast, IPFS takes a BitTorrent-like approach where nodes store only the content they are interested in.

Evaluation of Operational DHTs. Closer to our work are studies that measure operational DHTs [16, 66]. For example, the authors in [16] report churn rates similar to our findings. However, they have not carried out similar controlled experiments from multiple vantage points. Instead, they attempted DHT lookups and report latencies in the order of tens of seconds, significantly slower than IPFS achieves. The authors in [12] and [70] measured performance in the BitTorrent implementation of Kademlia. Their results contrast starkly with our own, showing a substantial number of failed nodes that negatively impact lookup times. Finally, Stutzbach and Rejaie [59], modeled Kademlia performance and proposed a set of improvements. Despite this, both studies revealed substantially worse performance than attained by IPFS.

The Fediverse. The growth of the IPFS network has occurred in tandem with other Decentralized Web technologies, most notably the "fediverse". The fediverse includes a number of server-based federated services, *e.g.,* Mastodon [47], Pleroma [22] and Diaspora [21]. Closest to our own work is Nextcloud, which provides a federated file storage platform, with IPFS integration in addition to server-local storage [26]. This is complementary to our own work, and operates in a similar fashion to IPFS gateways. In contrast, however, IPFS can continue to operate without gateways, whereas fediverse apps are entirely dependent on the uptime of the federated servers (see [47] for detailed analysis).

Incentives. There have been several studies that look at incentivizing participation in P2P systems [46]. There are also several large operational decentralized and incentivized P2P storage networks [18, 65, 69] with Filecoin [18] in particular building directly on top of IPFS and being the largest decentralized and incentivized storage network at the time of writing [68]. IPFS does not incentivize data storage, sharing, or participation in the network. IPFS can be considered a best-effort caching, storage and distribution

[5] List collected a day after the log capture at the gateway

[6] https://docs.ipfs.io/how-to/best-practices-for-nft-data/

layer that sits underneath the incentive structures layer. Section 5 shows that even without direct incentives, participation in IPFS is widespread, in-part because nodes are not required to store other users' content.

In order to go a step further and provide availability guarantees, "pinning services" offer to host and provide user content for a fee (see Pinata, or Infura). Availability guarantees can also be provided by Filecoin [18]. Filecoin's total storage capacity stands at 17.5 EiB at the time of writing (compared to AWS's 194 EiB[7]). Given that Filecoin is built on top of IPFS, it is expected to increase traffic in the IPFS network significantly in the coming years.

Content-Based Addressing. Content-based addressing has been widely used in P2P networks for many years [4, 37, 49]. More recently, content addressing has received significant uptake by the Information-Centric Networking (ICN) community. Prominent architectural proposals in this field include Networking Named Content (NNC) [28], Named Data Networking (NDN) [71], NetInf (which is also DHT-based) [13], Curling [6], as well as Secure Scuttlebutt (SSB) [62], more recently. IPFS is complementary to these proposals. Whereas they largely introduce content-based addressing at the network-layer, IPFS relies on application-layer routing. Clearly, if designed carefully, synergies between network-layer and overlay-based approaches can complement each other and result in a universal content-based addressing networking stack.

Decentralized Web Data Management. Researchers have also looked at data management and decentralized content storage more generally [57, 61]. For example, various projects have attempted to decentralize data control, *e.g.,* DataBox [45], SOLID [39], and SocialGate [32]. These operate local datastores for individual users, *e.g.,* running on a physical home appliance. While these solutions focus on controlling data usage and access, IPFS has a broader focus in providing a decentralized storage system, *e.g.,* by serving as a back-end [44].

8 CONCLUSIONS

This paper has detailed the design, implementation and deployment of the InterPlanetary File System (IPFS). As well as presenting the core design of IPFS, we have also presented measurement tooling that allows us to gain vantage on its decentralized operations. We have shown that IPFS has received widespread uptake, covering 152 countries and 2715 ASes. This uptake is in-part enabled by our hybrid gateway design, which has received considerable usage. The IPFS codebase[8] and datasets are freely available.

There are a number of avenues of future work. Most relevant, we plan to focus on minimizing retrieval and publication latency. For instance, we plan to exploit the Bitswap protocol to preemptively pair peers who may have similar content interests. We also plan to continue our large-scale performance monitoring to gain greater longitudinal insight into the scale and performance of the several components of the IPFS architecture. We plan to expand our studies to components such as the Hydra boosters,[9] which we have not covered here due to space constraints and their limited adoption.

We wish to emphasize that IPFS offers properties that go beyond those discussed within the remit of this paper. IPFS is an open, permissionless system and, as such, moderation remains a challenge. Studying potential forms of misuse is therefore a key line of future work. For instance, there have been reports that the Storm botnet[10] lives on the IPFS network, but the precise intention and activities of the botnet have not yet been identified. Our initial monitoring did not show abusive activity by IPStorm nodes (*e.g.,* frequent switching of PeerIDs, higher than usual rate of churn, or unresponsive behaviour), but investigation is part of our future plan. Similarly, we are yet to evaluate the resilience of IPFS or its capacity to sustain various types of information attacks (*e.g.,* censorship). Although our results shed light on these matters, for example, revealing that IPFS is robust to churn and is sufficiently geographically distributed to avoid single points of failure, this area is ripe for further work. As part of this, it is important to better understand the reliance that IPFS has on other centralized infrastructure (*e.g.,* cloud platforms), and to explore the types of networks that host IPFS nodes (*e.g.,* homes, universities). Thus, we plan to continue developing measurement techniques that can quantify these issues, and feedback into our ongoing community-driven design process.

ACKNOWLEDGEMENTS

We would like to acknowledge and thank the numerous contributors to IPFS, without whom with paper would not be possible. We would like to particularly thank Adin Schmahmann, the core maintainer of the go-ipfs codebase and mastermind of several of the techniques discussed in this paper, and Petar Maymounkov, the original author of Kademlia and also a maintainer of go-ipfs. Both generously provided endless discussions and explanations around the concepts presented here as well as offering thoughtful inputs into our methodologies. We would also like to thank the Pegasys Team (part of Consensys) and especially Zhenyang Shi for developing the basis for some of the measurement tools used here. This work was part-funded by projects EPSRC REPHRAIN "Moderation in Decentralised Social Networks", EP/S033564/1, EP/W032473/1 and EU Horizon 2020 grant agreements No 871793 (Accordion), No 101016509 (Charity). Gareth Tyson is corresponding author.

REFERENCES

[1] O. Abdullah Lajam and T. Ahmed Helmy. "Performance Evaluation of IPFS in Private Networks". In: *2021 4th International Conference on Data Storage and Data Engineering*. Barcelona, Spain: Association for Computing Machinery, 2021, pp. 77–84. DOI: 10.1145/3456146.3456159.

[2] C. Bommelaer de Leusse and C. Gahnberg. "The Global Internet Report: Consolidation in the Internet Economy". In: *Internet Society* (2019).

[3] T. Böttger et al. "The elusive internet flattening: 10 years of IXP growth". In: *RIPE 78* (2018).

[4] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. "A routing scheme for content-based networking". In: *IEEE INFOCOM 2004*. Vol. 2. IEEE. 2004, pp. 918–928.

[7] https://www.storageindex.io/

[8] https://github.com/ipfs/ipfs

[9] https://github.com/libp2p/hydra-booster

[10] https://en.wikipedia.org/wiki/Storm_botnet

[5] I. Castro, R. Stanojevic, and S. Gorinsky. “Using Tuangou to reduce IP transit costs”. In: *IEEE/ACM Transactions on Networking* 22.5 (2013), pp. 1415–1428.

[6] W. K. Chai et al. “Curling: Content-ubiquitous resolution and delivery infrastructure for next-generation services”. In: *IEEE Communications Magazine* 49.3 (2011), pp. 112–120. doi: 10.1109/MCOM.2011.5723808.

[7] I. Clarke et al. “Freenet: A distributed anonymous information storage and retrieval system”. In: *Designing privacy enhancing technologies*. Springer. 2001.

[8] K. Clay. *Amazon.com Goes Down, Loses $66,240 Per Minute.* 2013. url: https://web.archive.org/web/20210307232341/https://www.forbes.com/sites/kellyclay/2013/08/19/amazon-com-goes-down-loses-66240-per-minute/ (visited on 01/31/2022).

[9] B. Cohen. “Incentives build robustness in BitTorrent”. In: *Workshop on Economics of Peer-to-Peer systems*. Vol. 6. Berkeley, CA, USA. 2003, pp. 68–72.

[10] B. Cohen. *The BitTorrent Protocol Specification v2.* 2008.

[11] D. Coldewey. *Cloudflare DNS goes down, taking a large piece of the internet with it.* July 2020.

[12] S. A. Crosby and D. S. Wallach. *An analysis of bittorrent's two kademlia-based dhts.* Tech. rep. Rice Technical Report, 2007.

[13] C. Dannewitz et al. “Network of Information (NetInf) - An Information-Centric Networking Architecture”. In: *Comput. Commun.* 36.7 (Apr. 2013), pp. 721–735. doi: 10.1016/j.comcom.2013.01.009.

[14] D. Dias, J. Johnson, and J. Benet. *Bitswap – Protocol Specification.* 2020.

[15] *DNSLink Standard.* 2021. url: https://www.dnslink.io/ (visited on 02/02/2022).

[16] J. Falkner et al. “Profiling a Million User Dht”. In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement.* San Diego, California, USA: Association for Computing Machinery, 2007, pp. 129–134. doi: 10.1145/1298306.1298325.

[17] R. Fanou et al. “Pushing the frontier: Exploring the african web ecosystem”. In: *Proceedings of the 25th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 2016, pp. 435–445. doi: 10.1145/2872427.2882997.

[18] *Filecoin: A Decentralized Storage Network.* Tech. rep. Protocol Labs, 2017.

[19] Gnutella. *Gnutella Protocol Specification.* 2009.

[20] K. Graffi et al. “LifeSocial. KOM: A secure and P2P-based solution for online social networks”. In: *CCNC.* 2011.

[21] B. Guidi et al. “Managing social contents in Decentralized Online Social Networks: A survey”. In: *Online Social Networks and Media* 7 (2018).

[22] A. I. Hassan et al. “Exploring content moderation in the decentralised web: The pleroma case”. In: *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies.* 2021, pp. 328–335.

[23] S. A. Henningsen et al. “Mapping the Interplanetary Filesystem”. In: *2020 IFIP Networking Conference (Networking)* (2020), pp. 289–297.

[24] R. Holz et al. “Tracking the deployment of TLS 1.3 on the Web: A story of experimentation and centralization”. In: *ACM SIGCOMM Computer Communication Review* 50.3 (2020), pp. 3–15.

[25] *IPFS Ecosystem directory.* https://ecosystem.ipfs.io/. 2022.

[26] *IPFS for Nextcloud.* 2020. url: https://apps.nextcloud.com/apps/files_external_ipfs (visited on 02/02/2022).

[27] T. Isdal et al. “Privacy-Preserving P2P Data Sharing with OneSwarm”. In: *Proceedings of the ACM SIGCOMM 2010 Conference.* New Delhi, India: Association for Computing Machinery, 2010, pp. 111–122. doi: 10.1145/1851182.1851198.

[28] V. Jacobson et al. “Networking Named Content”. In: *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies.* Rome, Italy: Association for Computing Machinery, 2009, pp. 1–12. doi: 10.1145/1658939.1658941.

[29] M. F. Kaashoek and D. R. Karger. “Koorde: A simple degree-optimal distributed hash table”. In: *International Workshop on Peer-to-Peer Systems.* Springer. 2003, pp. 98–107.

[30] S. Kaune et al. “Modelling the internet delay space based on geographical locations”. In: *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing.* IEEE. 2009, pp. 301–310.

[31] P. Khare et al. “The Web We Weave: Untangling the Social Graph of the IETF”. In: *Proceedings of the International AAAI Conference on Web and Social Media.* Vol. 16. 2022, pp. 500–511.

[32] D. Koll, D. Lechler, and X. Fu. “SocialGate: Managing large-scale social data on home gateways”. In: *IEEE ICNP.* 2017.

[33] P. Labs. *AutoNAT.* 2022.

[34] P. Labs. *Direct Connection Upgrade through Relay.* 2021.

[35] P. Labs. *Merkle Directed Acyclic Graphs (DAGs).* 2021.

[36] V. Le Pochat et al. “Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation”. In: *Proceedings of the 26th Annual Network and Distributed System Security Symposium.* Feb. 2019. doi: 10.14722/ndss.2019.23386.

[37] G. Li, V. Muthusamy, and H.-A. Jacobsen. “Adaptive content-based routing in general overlay topologies”. In: *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing.* Springer. 2008, pp. 1–21.

[38] T. Mager, E. Biersack, and P. Michiardi. “A measurement study of the Wuala on-line storage service”. In: *2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P).* IEEE. 2012.

[39] E. Mansour et al. “A demonstration of the solid platform for social web applications”. In: *WWW.* 2016.

[40] E. Mathews. *Amazon cloud outage hits major websites, streaming apps.* Dec. 2021.

[41] P. Maymounkov and D. Mazieres. “Kademlia: A peer-to-peer information system based on the xor metric”. In: *International Workshop on Peer-to-Peer Systems.* Springer. 2002, pp. 53–65.

[42] S. McQuistin et al. “Characterising the IETF through the lens of RFC deployment”. In: *Proceedings of the 21st ACM Internet Measurement Conference.* 2021, pp. 137–149.

[43] *Multiformats – Self-describing values for Future-proofing.* 2021. URL: https://multiformats.io/ (visited on 02/02/2022).

[44] F. Parrillo and C. Tschudin. "Solid over the Interplanetary File System". In: *2021 IFIP Networking Conference (IFIP Networking).* 2021.

[45] C. Perera et al. "Valorising the IoT databox: creating value for everyone". In: *Transactions on Emerging Telecommunications Technologies* 28.1 (2017).

[46] M. Piatek et al. "Do Incentives Build Robustness in BitTorrent?" In: *4th USENIX Symposium on Networked Systems Design & Implementation (NSDI 07).* Cambridge, MA: USENIX Association, Apr. 2007.

[47] A. Raman et al. "Challenges in the decentralised web: The mastodon case". In: *Proceedings of the Internet Measurement Conference.* 2019, pp. 217–229.

[48] C. A. Rank. urlhttp://as-rank.caida.org/. 2022.

[49] S. Ratnasamy et al. "A Scalable Content-Addressable Network". In: *SIGCOMM Comput. Commun. Rev.* 31.4 (Aug. 2001), pp. 161–172. DOI: 10.1145/964723.383072.

[50] A. de la Rocha, D. Dias, and Y. Psaras. "Accelerating Content Routing with Bitswap: A Multi-Path File Transfer Protocol in IPFS and Filecoin". In: (2021).

[51] D. Roselli, J. R. Lorch, and T. E. Anderson. "A Comparison of File System Workloads". In: (), p. 14.

[52] M. Rosemain and R. Satter. *Millions of websites offline after fire at French cloud services firm.* Mar. 2021.

[53] A. Rowstron and P. Druschel. "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems". In: *IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing.* Springer. 2001, pp. 329–350.

[54] O. Saleh and M. Hefeeda. "Modeling and Caching of Peer-to-Peer Traffic". In: *Proceedings of the 2006 IEEE International Conference on Network Protocols.* IEEE, Nov. 2006. DOI: 10.1109/icnp.2006.320218.

[55] V. Santos and S. Allen. *IPNS - Inter-Planetary Naming System.* 2021. URL: https://github.com/ipfs/specs/blob/bab189ee61c316eb3d371c7270abef97641e7ed9/IPNS.md (visited on 02/02/2022).

[56] S. Saroiu, P. K. Gummadi, and S. D. Gribble. "A Measurement Study of Peer-to-Peer File Sharing Systems". In: *Multimedia Computing and Networking 2002.* Vol. 4673. SPIE, Dec. 10, 2001, pp. 156–170. DOI: 10.1117/12.449977.

[57] L. Schwittmann et al. "SoNet–Privacy and Replication in Federated Online Social Networks". In: *Distributed Computing Systems Workshops.* 2013.

[58] I. Stoica et al. "Chord: A scalable peer-to-peer lookup service for internet applications". In: *ACM SIGCOMM Computer Communication Review* 31.4 (2001), pp. 149–160.

[59] D. Stutzbach and R. Rejaie. "Improving lookup performance over a widely-deployed DHT". In: *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications.* IEEE. 2006, pp. 1–12.

[60] D. Stutzbach and R. Rejaie. "Understanding Churn in Peer-to-Peer Networks". In: *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement.* Rio de Janeiro, Brazil: Association for Computing Machinery, 2006, pp. 189–202. DOI: 10.1145/1177080.1177105.

[61] S. Taheri-Boshrooyeh, A. Küpçü, and Ö. Özkasap. "Security and privacy of distributed online social networks". In: *Distributed Computing Systems Workshops.* 2015.

[62] D. Tarr et al. "Secure Scuttlebutt: An Identity-Centric Protocol for Subjective and Decentralized Applications". In: *Proceedings of the 6th ACM Conference on Information-Centric Networking.* Macao, China: Association for Computing Machinery, 2019, pp. 1–11. DOI: 10.1145/3357150.3357396.

[63] D. Trautwein. *Nebula – A crawler for networks based on the libp2p DHT implementation.* Version 2.0.4. July 2021.

[64] *Udger Data v3 – 20220606-01.* 2022.

[65] D. Vorick and L. Champine. *Sia: Simple Decentralized Storage.* Tech. rep. Nebulous Inc., 2014.

[66] L. Wang and J. Kangasharju. "Measuring large-scale distributed systems: case of BitTorrent Mainline DHT". In: *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on.* Sept. 2013, pp. 1–10. DOI: 10.1109/P2P.2013.6688697.

[67] X. Wang, Y. L. Yin, and H. Yu. "Finding Collisions in the Full SHA-1". In: *Advances in Cryptology – CRYPTO 2005.* Springer Berlin Heidelberg, 2005, pp. 17–36. DOI: 10.1007/11535218_2.

[68] Web Storage Index. https://www.storageindex.io/. 2022.

[69] S. Williams et al. *Arweave: A Protocol for Economically Sustainable Information Permanence.* Tech. rep. arweave.org.

[70] S. Wolchok and J. A. Halderman. "Crawling BitTorrent DHTs for Fun and Profit". In: *4th USENIX Workshop on Offensive Technologies (WOOT 10).* Washington, DC: USENIX Association, Aug. 2010.

[71] L. Zhang et al. "Named Data Networking". In: *SIGCOMM Comput. Commun. Rev.* 44.3 (July 2014), pp. 66–73. DOI: 10.1145/2656877.2656887.

[72] B. Y. Zhao et al. "Tapestry: A resilient global-scale overlay for service deployment". In: *IEEE Journal on Selected Areas in Communications* 22.1 (2004), pp. 41–53.

Listing 1: Use the following BibTeX code to cite this article

```
@inproceedings{Trautwein2022a,
  title     = {Design and Evaluation of IPFS: A Storage Layer for the Decentralized Web},
  author    = {Trautwein, Dennis and Raman, Aravindh and Tyson, Gareth and Castro, Ignacio and Scott,
      Will and Schubotz, Moritz and Gipp, Bela and Psaras, Yiannis},
  year      = 2022,
  month     = {August},
  booktitle = {ACM SIGCOMM 2022 Conference (SIGCOMM '22)},
  publisher = {ACM},
  address   = {Amsterdam, Netherlands},
  doi       = {10.1145/3544216.3544232},
  note      = {ISBN 978-1-4503-9420-8/22/08},
  topic     = {misc}
}
```