The Cloud Strikes Back: Investigating the Decentralization of IPFS

Leonhard Balduf, Maciej Korczyński, Onur Ascigil, Navin Keizer, George Pavlou, Björn Scheuermann, Michal Król

The Cloud Strikes Back: Investigating the Decentralization of IPFS

Leonhard Balduf
Technical University of Darmstadt
Darmstadt, Germany
leonhard.balduf@tu-darmstadt.de

Maciej Korczyński
Univ. Grenoble Alpes
Grenoble, France
maciej.korczynski@univ-grenoble-alpes.fr

Onur Ascigil
Lancaster University
Lancaster, United Kingdom
o.ascigil@lancaster.ac.uk

Navin V. Keizer
University College London
London, United Kingdom
navin.keizer.15@ucl.ac.uk

George Pavlou
University College London
London, United Kingdom
g.pavlou@ucl.ac.uk

Björn Scheuermann
Technical University of Darmstadt
Darmstadt, Germany
scheuermann@kom.tu-darmstadt.de

Michał Król
City, University of London
London, United Kingdom
michal.krol@city.ac.uk

ABSTRACT

Interplanetary Filesystem (IPFS) is one of the largest peer-to-peer filesystems in operation. The network is the default storage layer for Web3 and is being presented as a solution to the centralization of the web. In this paper, we present a large-scale, multi-modal measurement study of the IPFS network. We analyze the topology, the traffic, the content providers and the entry points from the classical Internet.

Our measurements show significant centralization in the IPFS network and a high share of nodes hosted in the cloud. We also shed light on the main stakeholders in the ecosystem. We discuss key challenges that might disrupt continuing efforts to decentralize the Web and highlight multiple properties that are creating pressures toward centralization.

CCS CONCEPTS

• **Networks** → **Peer-to-peer networks**; **Network measurement**; **Network structure**; **Peer-to-peer protocols**.

KEYWORDS

ipfs, peer-to-peer networks, decentralization

1 INTRODUCTION

Interplanetary Filesystem (IPFS) [7] is one of the largest peer-to-peer (P2P) filesystem currently in operation. The platform underpins various decentralized web applications [43], including social networking and discussion [38, 41], data storage [45, 49, 58], content search [37, 53], messaging [76], content streaming [2, 3, 87] gaming [26, 39], and e-commerce [15, 17]. IPFS is widely used as external storage for blockchain-based applications, including valuable NFT platforms [4, 14].

The IPFS network currently contains a steady number of ≈ 30,000 [80] online nodes, spread across 2,700 Autonomous Systems and 152 countries, according to a recent study [78] that also observed widespread usage by clients with 7.1 million content retrieval operations observed from a single vantage point and during a single day. Support for accessing IPFS has further been integrated into HTTP gateways (*e.g.*, Cloudflare) and mainstream browsers such as Opera and Brave, allowing easy uptake.

IPFS is being presented as the default storage layer for Web3 with a strong focus on decentralization [46]. Storage decentralization is supposed to offer multiple benefits [30, 63]. Data is spread among many replicas, making privacy-intrusive data mining more difficult. Data ownership is more transparent, and the lack of centralization makes the overall system more robust against technical, legal or regulatory attacks. However, these properties may also bring inherent challenges that are difficult to avoid, particularly when considering the natural pressures towards centralization in both social [82] and economic [72] systems.

Contributions. In this paper, we evaluate the current state of the IPFS network with a focus on decentralization and make the following contributions. We build tools for multidimensional observation of the IPFS network. In contrast to previous studies [78], we not only discover the system participants but also the traffic generated by the network. This includes the distributed hash table (DHT) [51] and Bitswap [16], the two core IPFS protocols used for data discovery and exchange. Furthermore, we observe multiple entry points to the IPFS ecosystems (*e.g.*, HTTP gateways, browser extensions, Ethereum Name System (ENS)) using passive and active DNS measurements. We then analyze our 9-month dataset to provide insights into the state of the network, content exchange patterns and peer behaviour. We assess the centralization of the network and its reliance on cloud components. Finally, we discuss the drivers behind centralization and explore techniques that could reduce this propensity.

Findings. Overall, our main findings include (1) We observe that almost 80% of the IPFS DHT servers are hosted in the cloud with the top 3 cloud providers hosting 51.9% of the servers. Our results paint a different picture of the IPFS network from the one presented in a recent study [78] reporting less than 3% of the cloud-based nodes. We explain the reason behind the differences and show how small changes in the measurement methodology can lead to different conclusions. (2) We found that the network experiences

a high degree of traffic centralization. The top 5% of the nodes are responsible for up to 95% of the traffic with the largest cloud provider, Amazon-AWS, generating 96% of all the content resolution requests. We also found cloud-based storage platforms such as *nft-storage* or *web3-storage* holding a major share of persistent content in the network. (3) We show that content storage is heavily reliant on the cloud infrastructure. Nearly 95% of the content is provided by at least one cloud-based node. Furthermore, many non-cloud providers use cloud nodes as proxies for NAT traversal. (4) We show that major CDN players, such as Cloudflare, dominate the IPFS HTTP gateway ecosystem. Furthermore, even IPFS content referenced by the decentralized Ethereum Name System (ENS) is mostly stored by a handful of major cloud providers.

2 BACKGROUND

In this section, we provide the necessary background information to understand the systems involved, and the methodology used to produce the measurements presented in this work. We start with a description of IPFS in general, followed by detailed explanations of its network protocols, content provision and retrieval mechanisms, as well as DNSLink and Hypertext Transfer Protocol (HTTP) gateway functionality.

IPFS. IPFS is a content-centric network where nodes are identified via their *peer ID*, which is derived from the public key of a unique key pair. By default, nodes maintain the same ID over time but a new one can be generated on request. Each node advertises a set of network endpoints describing their IP address, transport protocol and port number. One peer ID can be associated with multiple endpoints (*e.g.*, multihoming), and one IP address can be associated with multiple peer IDs (*e.g.*, when hosting multiple nodes on a single machine).

In IPFS, each piece of content is identified by a content identifier (CID). A CID for item d is derived by hashing the content of d, so that $\text{CID}(d) = h(d)$ for some cryptographic hash function h.[1]

CIDs do not contain information about the content location, are immutable (changing the content generates a new CID), and are not human-readable. This enables easy content deduplication, data retrieval from the closest available location, and maintaining data integrity. However, it also means that a downloader first needs to resolve a CID to a list of providers, *i.e.*, nodes storing the content, before the actual content transfer. The resolution is done using a Kademlia [51] DHT and the Bitswap protocol. Figure 1 shows typical interactions between entities in the network, which we will now explain in more detail.

Bitswap. Bitswap [16] is a simple protocol used to exchange blocks of data. Typically, IPFS nodes maintain Bitswap connections to a few hundred random peers.[2] The protocol allows one to ask a peer whether it has a target block or to directly request and transfer the target content (cf. Figure 1 ⑤).

DHT. IPFS uses a Kademlia [51] DHT implementing a key-value store. A new participant node joins the IPFS network by contacting one of the hardcoded bootstrap nodes. This bootstrap node provides

[1]In practice, the CIDs include some metadata and are encoded using a self-describing format. We refer the readers to other studies [78] for a more detailed description.
[2]This number may differ depending on the configuration.

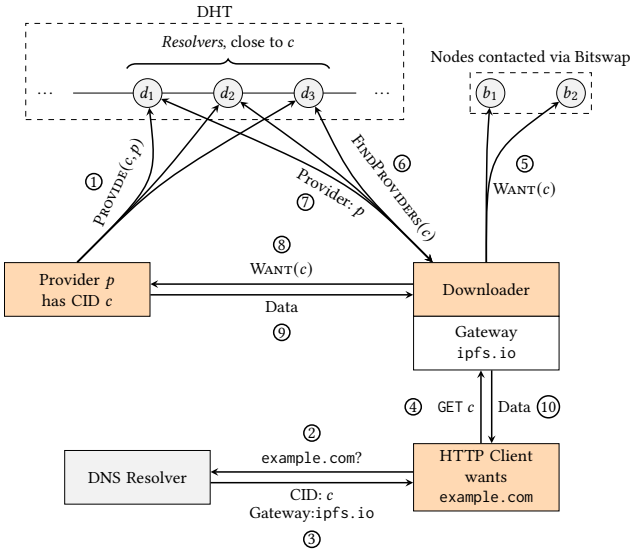


Figure 1: Illustrations of IPFS Provide and Request Functionalities.

the new node with some initial peers allowing it to join the DHT. The new node uses this information to perform a walk through the DHT towards its own peer ID to discover peers and fill its routing table.

The main operation GetClosestPeers(key) traverses the DHT and returns the k closest peers to the target key. In each step, the querying node contacts the closest nodes to key it knows of. Each of these peers returns the k closest peers to key in its own routing table and the addresses of these peers. The querying node again sends requests to the peers closest to key, among peers it just received. This process repeats until the client does not find any more peers closer to key.

Recent versions of the software differentiate between DHT *servers* and DHT *clients*. The latter only use the DHT as a service for resolution and routing, which is provided through DHT servers. To become a DHT server, the software determines whether it is connectable from the internet (as opposed to, *e.g.*, NAT-ed). Only connectable nodes become DHT servers unless the user explicitly modifies their configuration. Generally, the set of DHT clients can be understood as the user-operated fringe of the network, consisting of nodes behind NAT, whereas the DHT servers form the network's core.

Content Advertisement. When a user adds content to the network, it adds it to its local node and uses the DHT to advertise itself as a *provider* for the CID representing the content. First, it creates a *provider record* that contains c and its own network information. During a Provide(c) operation, the provider first uses GetClosestPeers(c) to locate the $k = 20$ peers closest to c, and then sends them a PutProvider message including the provider record. We call the peers that hold provider records for c the *resolvers* for c. (cf. Figure 1 ①)

By default, each IPFS client becomes a provider for each piece of content it downloads and automatically registers itself as a provider

for the corresponding CID. As a result, the system provides an auto-scaling feature with supply automatically rising with demand.

A provider without a public IP address (*e.g.* DHT client) cannot directly receive download requests for the content it provides, unless it is already connected to the downloader via Bitswap. Generally, NAT-ed nodes first establish a connection to a random DHT server supporting the relay protocol that will act as a reverse proxy and NAT-punching introducer. The provider includes the IP address of the proxy in the provider records it generates. As of v0.13, IPFS includes a NAT hole-punching mechanism called direct connection upgrade through a relay (DCUtR), which is functionally similar to the one used in the SSU protocol of the I2P network [60, 77].

Content Retrieval. Downloading a data item d with CID c is a two-step process: (1) providers for c are found (cf. Figure 1 ⑤, ⑥, ⑦), (2) connections to the providers are established and d is downloaded from them directly via Bitswap (⑧, ⑨).

The search for providers begins with a local, 1-hop broadcast via Bitswap (⑤) to all connected neighbours looking for the target CID. Searching via Bitswap is fast, but does not provide reliable content resolution, in particular for less popular or new content. If this does not yield any results, the downloader invokes FindProviders(c) using the DHT (⑥). This operation uses a DHT walk identical to that of GetClosestPeers(c) to find up to k resolvers but also queries encountered nodes for a provider record for c. The process terminates when either 20 providers have been found, or all resolvers have been asked (⑦). The downloader concurrently initiates Bitswap connections to the discovered providers and retrieves the requested content (⑧, ⑨).

Entry Points. Accessing content with IPFS can be considered complex due to a few reasons. Downloading IPFS content requires installing additional software, joining the P2P network and using custom protocols. While CIDs ensure content integrity and prevent tampering, they can be long and difficult for users to remember or share compared to traditional URLs or domain names. Furthermore, the identifier changes with every modification of the content. For instance, modifying a website hosted on IPFS creates a completely new CID that must be communicated to all the website viewers. Another limitation is browser support: traditional web browsers are designed to work with the HTTP/HTTPS protocols and may not have native support for IPFS. To simplify content access, IPFS implements multiple tools bridging the gap with the traditional Web.

HTTP Gateways. Gateways translate HTTP GET requests to content retrievals in IPFS and enable IPFS-agnostic users to access the content (cf. Figure 1 ④, ⑩).

When a gateway receives an HTTP GET for a CID, it (1) checks its local cache (2) finds and downloads the content using IPFS, and (3) returns the content to the client using HTTP. Protocol Labs maintains a list of public gateways [40], some of which are operated by large content delivery networks such as Cloudflare. Previous studies showed extensive usage of gateways and their noticeable share of traffic in the IPFS network [4, 5].

DNSLink. DNSLink [42] enables content publishers to associate domain names with IPFS content. It integrates the traditional DNS with IPFS, enabling users to access IPFS content using familiar domain names instead of intricate CIDs (cf. Figure 1 ②, ③).

DNSLink leverages DNS records to establish a connection between a specified domain name, such as `example.com`, and an IPFS address. It is achieved by storing a DNS `TXT` record within a dedicated subdomain beginning with the `_dnslink` label (e.g., `_dnslink.example.com`). The structure of the `TXT` record follows the guidelines outlined in RFC 1464 [65], which defines a formatted representation as `<key>=<value>`. Within the DNS `TXT` record, one can find either of the following entries: `dnslink=/ipfs/<CID>` or `dnslink=/ipns/<hash of public key>`. The first one associates the CID directly with the domain name, whereas the second one associates the IPNS key's hash value with the domain. The second approach enables redirecting users, e.g., a website visitor, to the most recent version of an object in IPFS, considering that modifying an object alters its CID.

To ensure that content stored on IPFS can be accessed, domain name owners need to configure their root domain (e.g., `example.com`) or subdomain (e.g., `subdomain.example.com`) to point to an IPFS gateway or proxy server. This configuration can be done in two ways: (1) assigning the IP address of the IPFS gateway or proxy server as the value of the `A` record for the domain, or (2) setting a `CNAME` or `ALIAS` record that matches the domain of the IPFS gateway or proxy server. By following either of these methods, domain name owners can establish the necessary connection between their domain or subdomain and the IPFS gateway or proxy server, enabling the retrieval of content stored on IPFS.

If the DNS provider supports `ALIAS` records, they are generally recommended for pointing the root domain to an IPFS gateway or proxy server. For instance, to configure the `example.com` domain, a domain name owner can add the following `ALIAS` record in the zone file: `example.com ALIAS gateway.ipfs.io`, directing `example.com` to a public gateway operated by Protocol Labs. Similarly, a subdomain can be configured with a `CNAME` record such as `subdomain.example .com CNAME cloudflare-ipfs.com`, directing it to a public gateway operated by Cloudflare.

Ethereum Name Service (ENS). The Ethereum Name Service (ENS) [68] is an alternative name-registry service for Web3 which allows users to register name-value pairs directly on the Ethereum blockchain [84]. One of the prominent use cases of ENS is to provide a mapping from human-readable domain names to cryptographic hashes such as IPFS CIDs, without relying on the centralization in the current DNS infrastructure in the form of Top-Level Domain (TLD) ownership and reliance on ICANN.

Namespace management in ENS is governed by several smart contracts. The *Registry* maintains a top-level mapping of all domains and subdomains to their owner, resolver, and caching time-to-live. *Registrar* contracts maintain ownership of individual domains (*e.g.* `.eth`) and their subdomains. Finally, the *resolver* contract for a (sub)domain points towards a value mapping set by the owner such as an Ethereum address or IPFS CID and assists users in resolving names in a decentralized manner.

3 METHODOLOGY AND COLLECTED DATASETS

In this section, we explain the methodology employed in this work, and the datasets derived through it. An architectural overview, showing how different functionalities of IPFS nodes are measured, is shown in Figure 2. We make all code processing data available at [6].

Topology graph. As explained in Section 2, IPFS builds on top of a Kademlia DHT, with nodes being partitioned into clients and servers. A node with address a_n stores its outbound DHT-connections in k-buckets, which form a view of the network as a binary trie. Buckets have a fixed capacity of k connections, which generally leads to the first, furthest, buckets to be filled completely, whereas buckets closer to a_n tend to contain fewer and fewer connections. Only peers providing DHT server functionality are stored in the buckets.

It is possible to enumerate all DHT connections of a node through crafted FindNode messages, sweeping the address space towards the target node's own address a_{target}. This process is generally known as DHT crawling [32, 57].

Using the DHT crawler presented in [32, 79], we can enumerate all outgoing DHT connections of functional DHT server nodes. This results in a snapshot of the DHT graph G_{DHT}, at the time of crawling. In practice, not all nodes are connectable and cooperative, which leads to un-crawlable leaf nodes in the graph. We crawl the network at least twice per day from 2023-04-18 to 2023-05-26, for a total of 101 snapshots. We discover an average of 25771.6 peers per crawl, of which 17991.4 are connectable and crawlable. A crawl takes 5.0m on average, of which the latter half is typically spent waiting on unresponsive peers, with a connection timeout of 3m. Short crawl durations are important to capture accurate snapshots of the network due to churn [13, 22, 73, 74], and long connection timeouts ensure completeness [74].

Counting Methodologies. We propose a methodology to derive properties of a typical snapshot the IPFS DHT, which is dynamic by default. Every node on the IPFS network is identified by a unique peer ID. Nodes can announce multiple IP addresses for themselves, which are stored in the DHT. An example of our dataset, with an additional mapping from IP addresses to geolocation, is shown in Table 1. From this example, we will try to derive the typical client population and its geospatial distribution.

Table 1: Example Crawl Dataset

Crawl ID	Peer ID	IP	Geolocation
1	p_1	a_1	DE
1	p_1	a_2	DE
1	p_2	a_3	US
2	p_2	a_2	DE
2	p_2	a_3	US
2	p_2	a_4	US

Nodes are typically announcing multiple IP addresses, which may differ in geolocation. The easiest way to resolve this is by counting unique IPs and their mappings, over the entire dataset. This ignores the time-discrete nature of the network snapshots but gives an estimate of the population of the network over the entire data collection period. It overcounts peers announcing multiple IPs, especially ones with frequently changing IPs, and includes churning nodes in the count. For the example dataset, this results in DE=2, US=2. We refer to this methodology as *Global, Unique IP* (G-IP), which is conceptually similar to the one employed in [78].

To combat the problem of overcounting nodes with multiple addresses, we propose to assign each *peer* a single value of the derived property in question, in this case, geolocation. We can then count peers, the participants of the overlay network, instead of underlay addresses. We propose to use a majority vote to decide on a property.

Still, counting peers over all crawls fails to estimate the value of a property for a *typical* DHT graph. Specifically, it overcounts nodes regenerating their peer ID and churning nodes. It can be tackled by considering each crawl as a separate snapshot of the network, for which a property is derived, and then average over all crawls. It is not sufficient to examine a single crawl, as this misses nodes that are offline during that point in time. As such, we propose *Average over Crawls, Unique Nodes* (A-N), which assigns each peer a value per crawl and averages over all crawls, which, for the example dataset, produces DE=0.5, US=1. This is intuitively correct: There is one stable node, probably situated in the US, and one node with 50% uptime in Germany.

The single-crawl average number of nodes obtained through is A-N 25771.6. the number of unique peer IDs in aggregate over all 101 crawls is 53898. A peer advertised an average of 1.82 non-local IP addresses across all crawls, which results in 86064 unique non-local IP addresses (G-IP).

Bitswap logs. Content retrieval in IPFS always starts with a provider discovery phase. This, currently, uses a local 1-hop broadcast via Bitswap to connected neighbours and subsequently searches the DHT. Using the monitoring infrastructure described in [5], we can monitor this discovery traffic of a large portion of the network via *monitors*. These are modified Go IPFS implementations with unbounded connection capacity, which log all incoming Bitswap traffic to disk. We run one Bitswap monitoring node which continuously collected data from August 2022 to May 2023. We refer to this dataset as the raw, unmodified *Bitswap traces*. The traces are a subset of all Bitswap traffic in the network, because the monitor, albeit with unbounded connection capacity, is not connected to everyone in the network. Additionally, as explained in Section 2, only the initial provider discovery request is broadcast via Bitswap. Lastly, we only see locally broadcast requests, not unicast responses.

In addition to the raw, unmodified Bitswap traces, we process the data further to obtain a daily sample of requested CIDs. For that, we aggregate all request traffic for a day and extract all requested CIDs. These are then deduplicated and a fixed amount of 200k is randomly sampled. We refer to this dataset as *daily sampled Bitswap CIDs*.

Hydra-booster logs. We set up a modified version of Hydra-booster [48] to collect the IPFS DHT traffic. The Hydra-boosters acts as a DHT server with multiple, virtual peer IDs co-located on a single virtual machine. We use Hydra-booster with 20 virtual peer

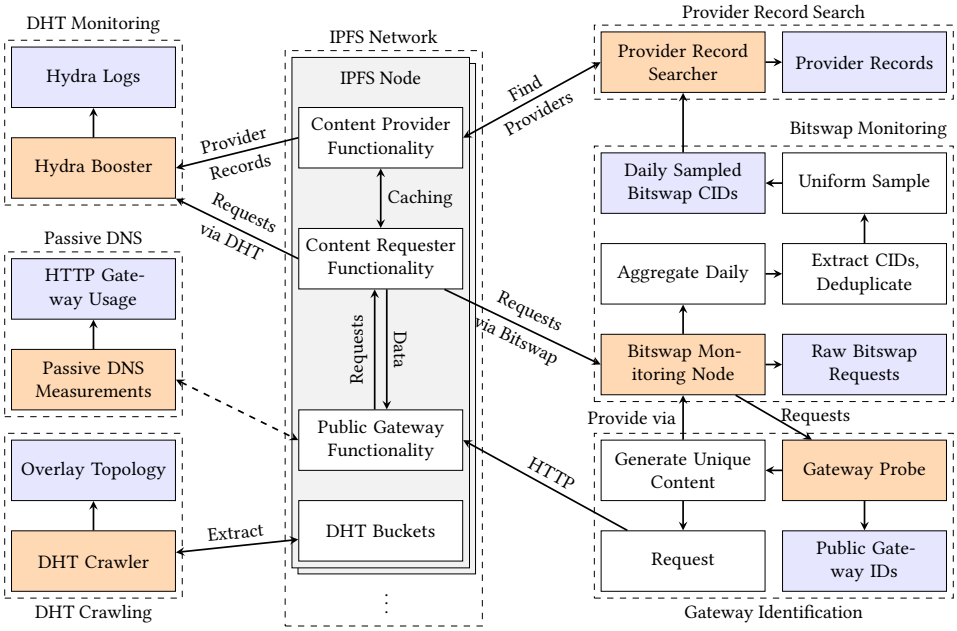


Figure 2: Data Collection Architecture, Overview.

IDs and modify it to write all the incoming DHT requests to disk. We log the timestamp, the sender's peer ID and IP address, the type of the request, and the target key (peer ID or CID depending on the message type). We also log the proxy DHT server if the original sender uses NAT traversal mechanisms (Section 2). Apart from collecting the traffic, our software acts as a regular DHT server following the IPFS specification.

We collect traffic over two time periods: from August 2022 to November 2022 and from February 2023 until April 2023 resulting in 290M messages. Based on our observation, an average DHT query contacts 50 different nodes and the network contains 25,000 DHT servers. We thus estimate that our Hydra node captures around 4% of the entire IPFS DHT traffic.

Provider Records. A peer can initiate a FindProviders(c) operation to retrieve provider records for CID c. This operation collects up to 20 providers for a given CID c, terminating when either 20 providers of c have been found or all the resolvers (*i.e.* 20 closest peers to c) have been queried for provider records of c.

We modified the FindProviders(c) implementation to terminate only when all the resolvers of c have been queried in order to retrieve all the provider records for the CIDs. Using this modified IPFS DHT implementation, we retrieved all the provider records of the CIDs from the daily sampled Bitswap CIDs datasets starting from 23 April 2023 until 20 May 2023 for a total of 28 days. The resulting dataset contained 5.6 million CIDs and their provider records. We retrieved the provider records of each daily set of CIDs on the same day they were collected.

Gateways. Public gateways translate between HTTP and IPFS. While their HTTP endpoints are public, their overlay IDs are generally unknown. We can identify these gateways on the overlay network through our Bitswap monitoring infrastructure in combination with specially crafted requests.

To identify a gateway, we generate a unique, random piece of data, and store it on our monitoring nodes. We can be reasonably certain that we are the only provider of this data in the network. We then request this data through the HTTP-side of a public gateway. This will trigger the usual discovery and request mechanisms, which, eventually, result in a request via Bitswap to our monitoring node. From this request, we can learn the overlay ID and address of the gateway.

Notably, many large gateway providers operate multiple IPFS nodes to serve their traffic reverse proxied and served from a single HTTP endpoint. While we can only identify one of these nodes per request, repeating these probes over time results in the discovery of multiple overlay IDs. Eventually, we can be relatively certain to have identified all operational gateway nodes.

Of the 83 HTTP endpoints listed in the public gateway list, we find 22 gateways that functioned at least once, and 119 unique overlay IDs associated with these. These numbers are in line with the ones reported via the public gateway checker tool [40].

Active and Passive DNS. To estimate the number of domain names utilizing IPFS for content delivery through the DNSLink mechanism, we employ active and passive DNS data sources.

Our active scanning input list comprises domain names collected in April 2023 from the following sources: (1) centralized Zone Data Service [33] for both legacy and new generic TLDs (e.g., `.org` or `.xyz`) (2) publicly available zone files of three country-code TLDs: `.se`, `.nu` [25], `.ch` [75] (3) Google Certificate Transparency logs [9] (4) Tranco top popularity domain name ranking [56], and (5) passive DNS data kindly provided by SIE Europe [69].

We filter the domain names using Mozilla's Public Suffix list [24] and retain only the root domain names (*e.g.*, `example.com` or `example.com.uk`). Next, using the `zdns` [34] scanner and Cloudflare Public DNS, we send DNS `SOA` (Start of Authority) requests to determine registered domain names while excluding those resulting in an `NXDOMAIN` response code, indicating non-existing domains. Our resulting list consists of 286M root domain names.

Administrators need to configure a DNS `TXT` record on a dedicated subdomain starting with the `_dnslink` label to indicate the CID or the IPNS key's hash value associated with the domain. Hence, for each domain name, we append the `_dnslink` label (e.g., `_dnslink.example.com` or `_dnslink.example.com.uk`) and perform an active scan to retrieve the `TXT` records, verifying if they contain properly formatted DNSLink entries. We actively scan for DNS `A` resource records on domains with valid DNSLink entries to ascertain whether the owner has configured a public IPFS gateway or another proxy server. Note that our measurements do not include the identification of subdomains using IPFS for content delivery. The `_dnslink` prefix can be added to any subdomain, such as `_dnslink.subdomain.example.com`.

Finally, our objective is to compare the list of IP addresses of domain names using IPFS for content storage with the IP addresses of public gateways. One approach is to perform active scans by querying DNS `A` resource records for the domain names of public gateways [40]. However, this approach has a limitation because DNS servers may provide different responses based on the geographic location of the querying client. To address this limitation, we leverage one month of passive DNS data provided by SIE Europe from March 2023. From this data, we extract all the observed IP addresses associated with the domain names of public gateways. It is important to note that if a gateway operator utilizes Anycast DNS [52], where the same IP address is advertised on multiple nodes, measurements from a single location would not impact the results.

Ethereum Name Service. We examine ENS records pointing to IPFS CIDs. To collect our ENS dataset, we employ a similar methodology as used in [85]. Resolver contracts maintain information regarding name mappings. Therefore, we start by compiling an exhaustive set of 16 resolver smart contracts from prior work and Etherscan [21], and extract and traverse through the full history of event logs using the Etherscan API.

We filter for the `setContenthash()` function call, as defined in EIP-1577 [20], which allows for a content hash to be set as the value of the record. From these, we specifically filter for records pointing to `ipfs_ns` records, finding a total of 20.6k records. We attempt to resolve the CID for each record to fetch providers of the content, finding 16.8k provider records, out of which there were 9k unique IPs.

4 THE NETWORK

We examine the IPFS overlay through the DHT crawl dataset. We discover an average of 25771.6 peer IDs per crawl, of which 17991.4 are crawlable. Over the entire dataset we observe 53898 peer IDs and 86064 unique IP addresses.

Cloud Nodes. First, we investigate the ratio of nodes hosted on major cloud providers in contrast to the number of non-cloud nodes. Similar to a previous study by Trautwein *et al.* [78], we employ the Udger IP database [81], which maps IP addresses to known cloud providers. If there are no entries for a given address in the database, we mark it as non-cloud. For peers announcing multiple cloud IP addresses, we assign the majority provider. If a peer announces both cloud and non-cloud IP addresses, we assign it a `BOTH` label.

Figure 3 shows the ratio of cloud nodes found in the DHT. Surprisingly, we discover a strong reliance on the cloud infrastructure. An average of 20300 (79.6%) of the nodes are hosted in data centers, with only 4737 (18.6%) non-cloud nodes. Crawls of the IPFS DHT only enumerate DHT servers, which require a public IP address. The move towards cloud nodes can thus be explained by IPFS users having their regular machines hidden behind NAT. However, such strong reliance on the cloud threatens the decentralization property of the system as the DHT forms the core of the platform.

Our results contrast with a previous study from 2022 [78] showing less than 3% of cloud nodes in the DHT. We discovered that the inconsistency is due to a difference in aggregation and counting methodology (cf. Section 3) and a difference in the frequency, and thus number, of crawls. The previous study creates a set of all unique IP addresses found across a large number of crawls (regardless of their relationship with the peer IDs) and then performs cloud provider attribution. This results in 34375 (39.9%) addresses on cloud providers, and 51689 (60.1%) non-cloud addresses. We argue that this approach does not reflect the actual, typical state of the network. As we show later, non-cloud IPFS nodes tend to be short-lived and frequently change their IP addresses, artificially inflating their share in the network.

We further showcase this phenomenon in Figure 4 showing the ratio of cloud to non-cloud nodes as a function of the number of aggregated crawls using both methodologies. Using the approach from Trautwein *et al.* [78] makes the ratio of non-cloud nodes increase with the number of aggregated crawls. This is because non-cloud nodes frequently rotating their IPs and churning nodes are counted multiple times. On the other hand, aggregating the ratio of cloud to non-cloud nodes using our approach result in steady ratio values. For the remainder of this section, we show results using both approaches to showcase how different views of the network can be obtained.

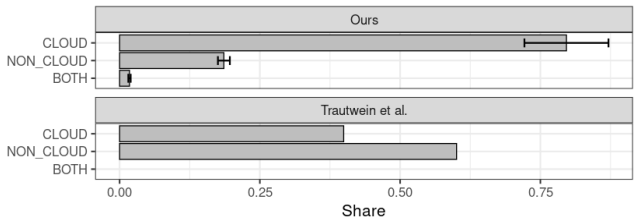


Figure 3: Participants of the IPFS DHT by Cloud Status, Comparison Between Counting Methodologies.

Cloud Providers. In Figure 5, we take a closer look at the cloud providers hosting IPFS nodes. We find that the majority (7492, 29.3%)

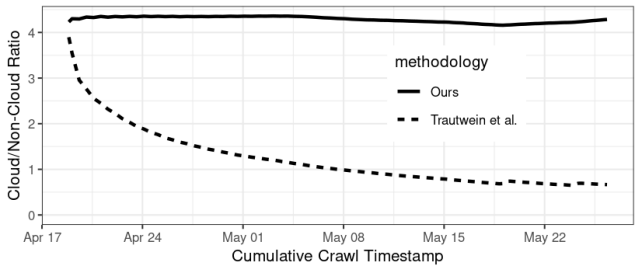


Figure 4: The ratio of Cloud to Non-Cloud Nodes as a Function Over Cumulative Successive Crawls, comparison between counting methodologies.

of nodes are hosted on *choopa* and the three main providers host 13259 (51.9%) of nodes. This is a much stronger dependency on a single cloud provider than for other decentralized networks such as Mastodon, where only 6% of the nodes are hosted on Amazon AWS [64]. Interestingly, using the alternative methodology [78] reduces *choopa*'s share to 13.8%. Furthermore, some providers such as *digital ocean* have a relatively lower share using this methodology. This might suggest that nodes hosted on these providers rotate their IP addresses less frequently.

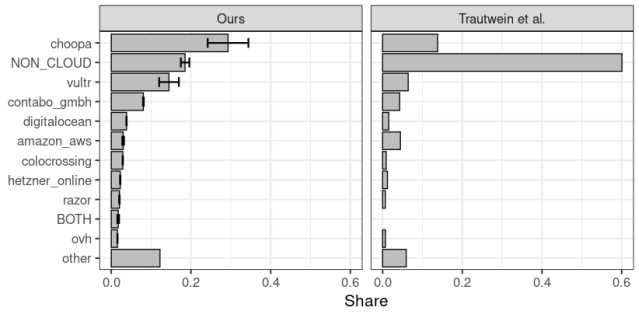


Figure 5: Nodes of the IPFS DHT Graph by Cloud Provider.

Geolocation. Next, we investigate the geospatial distribution of nodes in the overlay (Figure 6). For this, we geolocate each node's addresses using the MaxMind GeoLite2 [50] database. The majority of nodes are situated in the United States (47.4%), Germany (13.7%), and Korea (5.2%) while only 13.3% are located outside the top 10 countries. Note that this is independent of our vantage point, because we crawl the entire DHT. This is in contrast to the results obtained via [78], where the majority of peers reside in the United States (33.0%), China (11.1%), and Germany (8.0%), with non-top-ten countries accounting for 22.9%. This is caused by short-lived IPs located in less represented countries that change frequently between crawls.

Node degree. Using our DHT crawl dataset, we recreate the topology of the overlay network. Out of 25771.6 DHT servers, 17991.4 responded to our crawl requests, on average. For those nodes, we learn their complete k-buckets, *i.e.*, all outgoing DHT connections,

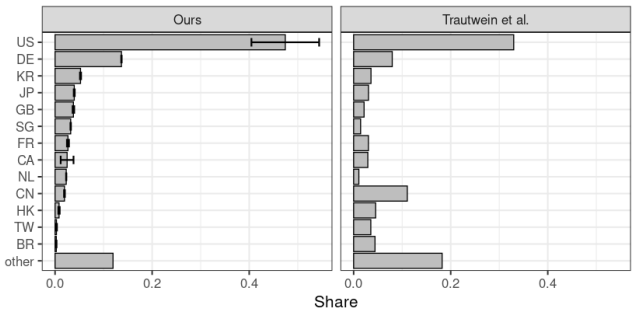


Figure 6: Nodes of the IPFS DHT Graph by Origin Country.

which correspond to the edges in our graph. The incoming connections are not directly available in the k-buckets, we thus estimate every node's in-degree by their presence in other peers' buckets. This undercounts the true in-degree, as we cannot crawl all nodes. The results of these investigations on a single graph from May 12th 2023 are shown in Figure 7.

The out-degree of nodes generally lies within a small band, which is ultimately dictated by the parameter k. Even though IPFS holds a large number of potentially unstructured connections, only a subset of those, accounting for the structured Kademlia overlay, is stored in fixed-size buckets [32].

The in-degree, on the other hand, is not dictated by Kademlia's k-buckets. These are a subset of all connections of a node, which are limited through a connection manager, attempting to keep between 600 and 900 open connections.[3] Nodes may increase this value to improve the chances of discovering content providers through Bitswap (cf. Section 2).

We observe a few high in-degree nodes, pointing to those nodes having a high number of connections in general. These nodes are contacted more often for DHT walks and perform central functionalities for the network. This creates points of centralization, as outages within highly connected nodes will have a disproportionate impact on the overall graph structure [1]. Out of the top 10 in-degree nodes, two are running a modified client by Filebase [23], a pinning service, the others are seemingly regular go-ipfs v0.11 nodes, out of which 8 are hosted on Amazon AWS. In general, though, most nodes have an in-degree of less than ≈ 200, with the 90th percentile being below ≈ 500.

Resistance to node removals. Finally, we investigate the impact of node removals to analyze the tolerance to attacks and failures [1, 31, 35, 64, 86]. We select a random crawl from May 12th 2023 with 24414 total and 16676 crawlable nodes, which we interpret as an *undirected* graph. This allows *all* observable connections of a node (DHT inbound and outbound) to be used for communication, including Bitswap.

We apply two different strategies of node removals to the graph: (1) Random, which picks a node at random, and (2) targeted, which picks the node with the highest degree. After each removal, we compute the connected components of the graph and count what portion of the remaining nodes is part of the largest [31]. We repeat

[3]These are preconfigured values, which were changed between IPFS releases.

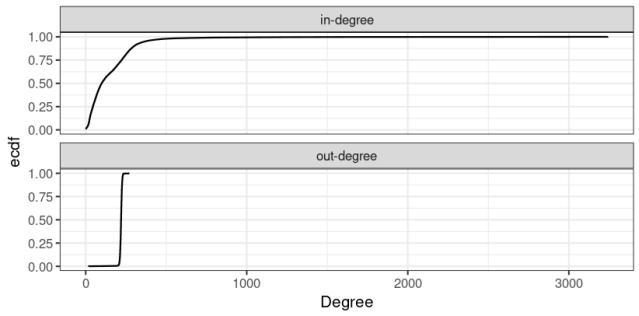


Figure 7: Degree Distribution of Nodes of the IPFS DHT Graph, Cumulative Density Function.

the random removal 10 times to be able to show a 95% confidence interval around the mean. The results of this are shown in Figure 8.

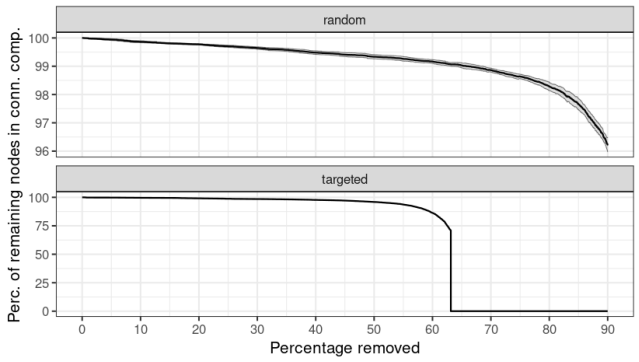


Figure 8: Resilience of the undirected IPFS DHT Graph to Random and Targeted Node Removals. Note the truncated vertical axis.

The network is very robust to random removals. The largest component spans 96% of remaining nodes even after randomly removing 90% of nodes. This has been attributed to networks being scale-free [1], *i.e.*, having a skewed distribution of node degrees. This is commonly found in structured P2P networks [66]. These results compare favourably to an earlier study done on IPFS in 2019 [31], which finds that only ≈ 50% of nodes remain connected at this point.

Targeted removal is more effective in disconnecting nodes from the system. The network ends up completely partitioned into components of size 1 after ≈ 60% of nodes were removed. This, still, points towards good resistance to node removals: A recent study on the resilience of the Mastodon social graph to targeted removal of users [64] found that removing even ≈ 10% of central user accounts leaves the majority of users outside the largest connected component. For the Twitter graph, this occurs at ≈ 30%. Our results compare favourably even to the earlier experiment on the IPFS DHT, which finds complete partitioning after ≈ 40% of nodes were removed.

It thus seems that the IPFS overlay, being very robust to random failures and only somewhat susceptible to targeted attacks, possesses properties both of structured [66] and unstructured [73] networks. This is in line with prior considerations on the robustness of the network [5, 32]. The network has even improved in robustness since the study performed in 2019 [31], which found that unconnectable leaf nodes can be removed easily with targeted attacks. These leaf nodes have been largely eliminated from the DHT in recent versions due to the differentiation between DHT servers and clients. Note, however, that we simplified the graph to be undirected, which allows Bitswap to be used on all edges, but ignores that Bitswap broadcasts only travel one hop. As such, even though the network stays mostly connected, it is not guaranteed that content stays available equally. A more nuanced analysis is left for future work.

5 THE TRAFFIC

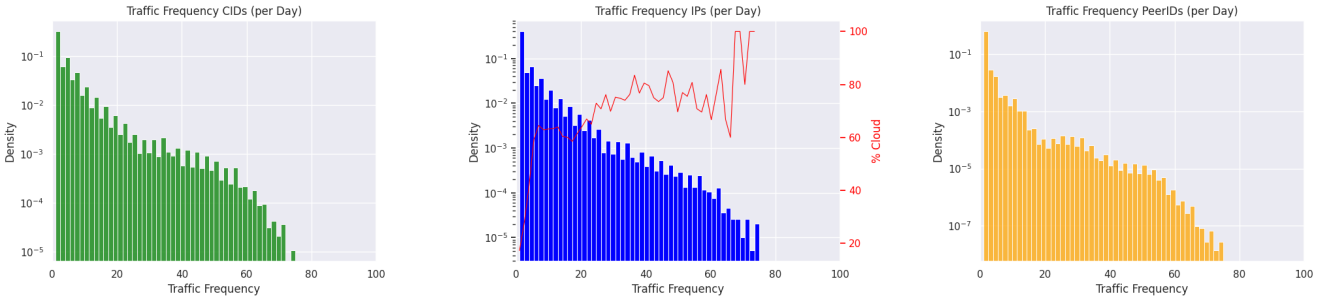
In this section, we investigate the traffic generated by the IPFS network. We use and compare our Bitswap and Hydra-booster datasets. We classify the DHT traffic into content-related *downloads* (*e.g.* requesting providers for a specific CID) and *advertisements* (*e.g.* announcing a new provider for a specific CID). We ignore all the other types of messages (*e.g.* nodes joining the network). We observe 290M messages where download-related traffic represents 57%, advertisement-related traffic 40% and other types of messages 3%. Importantly, our traffic traces include the NAT-ed peers (*i.e.* DHT clients) that were not visible in the previous section. Although Hydra-boosters receive traffic from DHT clients, their traffic is not distinguishable from the traffic by DHT servers. In Section 6, we present a better picture of how NAT-ed DHT clients impact content hosting in IPFS.

General metrics. First, we explore the temporal properties of the IPs, peerIDs and CIDs found in our Hydra-booster logs (Figure 9).[4] We define frequency as the number of different days where we observe the item. The vast majority of the CIDs are downloaded or advertised only for 1-3 days. This suggests that IPFS is mostly used for direct content transfer rather than persistent storage.

The majority of IPs and peerIDs are also short-lived. The result for the IPs suggests IP rotation and further explains result differences for two methodologies in Section 4. We observe a large portion of cloud nodes in general but their share increases for IPs seen over many days. The short-lived nature of the peerIDs is also surprising. By default, IPFS clients keep their peerIDs across multiple restarts. This suggests that many users use the network for a single interaction.

ID centralization. We then investigate the centralization of the traffic through the lens of the peerIDs (Figure 10) and distinguish between gateway and non-gateway nodes based on our *gateway dataset*. For both Bitswap and DHT, we observe high centralization of the traffic far beyond the Pareto 20%-80% principle. 5% of the most active peerIDs are responsible for almost 97% of the traffic. The gateway ratio differs significantly for both protocols (≈ 1% for DHT, ≈ 18% for bitswap). We suspect that this is caused by a large number of Bitswap connections maintained by each gateway and fixed links to the industrial content providers such as *pinata* or *nft.storage*.

[4]We exclude Bitswap logs from this analysis due to the sheer volume of data. Analyzing a random sample of the logs would not preserve the temporal properties.



(a) CID **(b) IP** **(c) PeerID**

Figure 9: Request frequency per identifier (in days seen). Note the y-axis log scale.

The gateways thus satisfy a vast majority of content requests via Bitswap and do not rely on the DHT for content resolution.

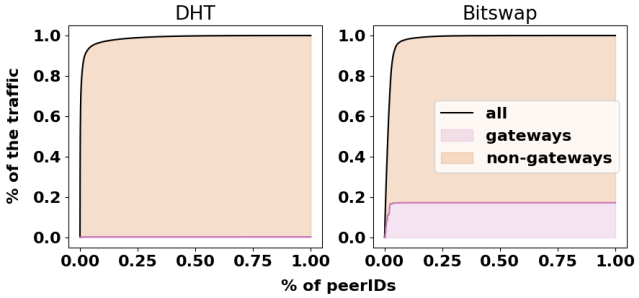


Figure 10: DHT/Bitswap peerID simplified Pareto chart.

IP centralization. We repeat the centralization experiments this time looking at IPs and distinguishing between cloud and non-cloud nodes (Figure 11). We observe a similar, high centralization of the traffic with 5% of the most active IP addresses responsible for almost 94% of messages. For the DHT, cloud nodes are the most active ones in the network generating a staggering ≈ 85% of the traffic. The non-cloud nodes, while similar in number, are much less active and are responsible for only ≈ 15% of the traffic. The distribution is similar for both download- and advertisement-related traffic (omitted on the graph). The Bitswap logs show a much smaller but still significant share of cloud nodes (≈ 42%). We explain this phenomenon in the paragraphs below.

Cloud providers. In Figure 12 (top graph), we analyze all IPs present in our logs distinguishing between traffic related to content downloading and advertising and indicating the most popular cloud providers. 35% overall ratio of cloud-based nodes. This is a smaller ratio than found during the network crawls (79% in Section 4). This is understandable, as the network crawls do not include the nodes behind NAT. We observe a similar division across cloud providers with *choopa*, *vultr* and *contabo* being the most popular ones, but an increased share of nodes hosted at Amazon AWS. Surprisingly, the cloud-based nodes are more present in the traffic related to downloading files (≈ 45%) than in the advertising traffic (≈ 34%).

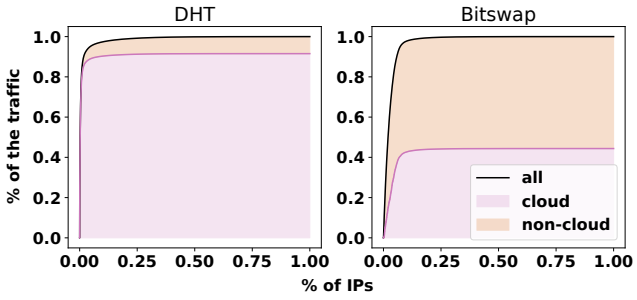


Figure 11: DHT/Bitswap IP simplified Pareto chart.

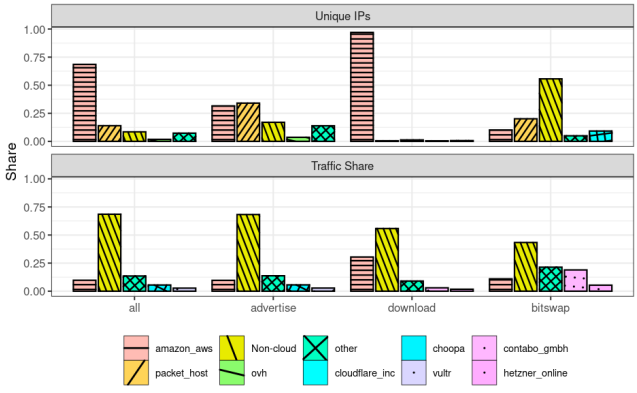


Figure 12: Cloud per traffic type

We repeat the analysis but this time we take into account the traffic generated by each IP address (Figure 12 bottom graph). Again, we observe that the cloud-based nodes are much more active and are responsible for ≈ 93% of the traffic. The ratio goes up to ≈ 98% for download-related traffic. The share of Amazon AWS raises to a staggering 68% followed by packet host being jointly responsible for 82% of the traffic.

IPFS-based platforms. To complete the picture of the IPFS traffic, we analyze the applications/platforms responsible for the traffic.

First, we obtain a set of peerIDs of Hydra-booster nodes operated by Protocol Labs and hosted on Amazon AWS. Those nodes were deployed in the network to speed up the DHT lookups. For all the non-Hydra IPs, we perform reverse DNS lookups (Figure 13).

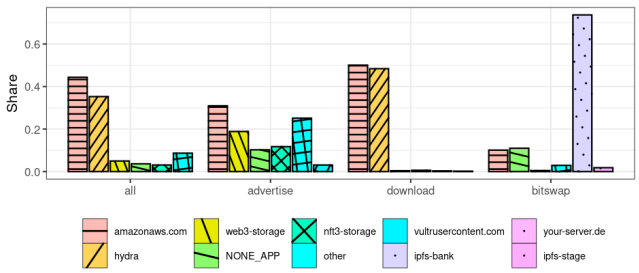


Figure 13: Platforms generating traffic based on reverse DNS lookups.

The Hydra-booster nodes are responsible for 35% of all the DHT traffic and 50% of the download traffic. However, it is not visible in the content advertising traffic. When a Hydra-booster receives a DHT content resolution request, it acts as a regular DHT node. First, it checks its cache for the relevant provider records. If those are unavailable, the Hydra-booster returns a list of peers closer to the target CID. However, the Hydra-booster also initiates its own lookups for all the requested CIDs that are not found in the cache, trying to proactively fill the cache for future requests. This behaviour exposes an easy Denial of Service vector. Asking a Hydra-booster for non-existing content generates significant amounts of traffic in the network. We are unable to confirm whether such intentional attacks happened during our measurement period or whether Hydra-booster nodes simply amplify regular requests for, potentially non-existing, content. This explains the higher share of cloud-based nodes in the DHT compared to the Bitswap traffic in Figure 11 and a higher share of Amazon AWS download-related traffic (Figure 12).

On the other hand, a few storage platforms using IPFS (*web3-storage* and *nft3-storage*) dominate the DHT advertise-related traffic. Those platforms offer practical persistent storage over IPFS using cloud infrastructure and thus periodically advertise all their CIDs in the network and explain the high cloud usage in the advertise-related DHT traffic (Figure 11, Figure 12). The Bitswap traffic is dominated by *ipfs-bank* which is an HTTP gateway platform. Unfortunately, we were not able to discover the purpose of the remaining traffic originating from Amazon AWS. While convenient for the end users and increasing accessibility, those platforms significantly contribute to the centralization of the network.

6 THE CONTENT PROVIDERS

In this section, we focus on the providers of content on IPFS. A provider record is a mapping of CID to *multiaddresses* — a self-describing address format, *e.g.*, /ip4/1.10.20.30/tcp/29087/ipfs/<peer ID>, that embeds provider's connectivity information and peer ID. We collect all the provider records of 5.6 million CIDs over 28 days. As provider records may be stale, *i.e.* a provider may have gone offline, we verified the reachability of the providers at the time of retrieving the provider record and ignored the unreachable ones.

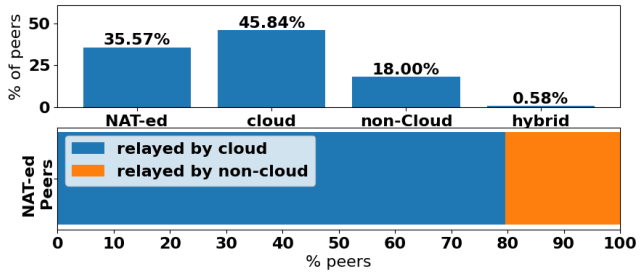


Figure 14: Classification of providers.

In IPFS, NAT-ed peers (*i.e.* DHT clients) can still provide content. This is done through a circuit relay protocol [59] where the NAT-ed peers keep a connection with a *relay* (*i.e.* a DHT server with a public IP) that reverse proxies the connection requests to make the NAT-ed peers reachable. When a NAT-ed peer advertises content, it provides a keyword *circuit* and the relay's IP address in its multiaddress.

Peer analysis. We categorize each unique provider peer ID based on its IP address(es) as one of: NAT-ed, cloud-based, non-cloud-based, and hybrid (Figure 14). The NAT-ed peers account for a significant portion (35.57%) of the providers. The cloud-based peers are the majority of the providers (45%), while 18% of the peers are non-cloud peers with public IPs. On the other hand, a very small percentage of the providers (*i.e.* 0.58%) had a mix of cloud and non-cloud IP addresses (*hybrid* in Figure 14). Those peers either have two instances in both cloud and non-cloud nodes or have moved during the provider record collection as we take a snapshot of a content's providers only once.

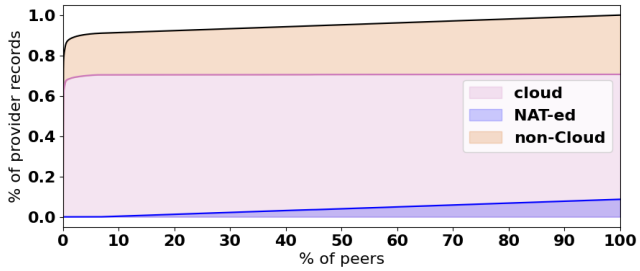


Figure 15: A simplified Pareto chart of peer IDs.

In the bottom plot of Figure 14, we present the distribution of relay nodes used by the NAT-ed peers. We observe that around 80% of the NAT-ed peers use a cloud-based peer as a relay node. This means that a large portion of NAT-ed peers makes use of cloud-based nodes to provide content.

Provider popularity. We look into the popularity of each provider in terms of the number of times each appears in the collected provider records (Figure 15). A small percentage of peers appear in a large percentage of provider records. Around 1% of the peers

appear as *one of the* providers in approximately 90% of the records. A large portion (*i.e.* 70%) of these popular providers are cloud-based, while NAT-ed peers appear in less than 8% of the records. On the other hand, the non-cloud providers (with public IP addresses) appear in around 22% of the provider records.

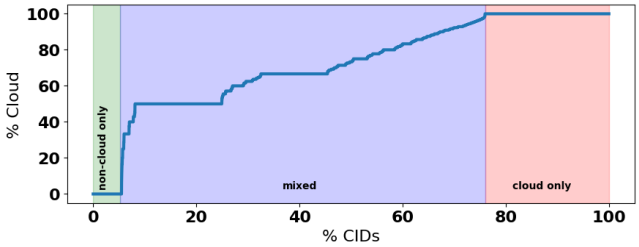


Figure 16: CIDs classified based on their providers.

Content-level analysis. A single CID can be hosted by multiple providers. We explore the properties of the content in terms of its reliance on cloud-based peers. For each content, we calculate the percentage of its cloud-based providers among all of its providers (shown as "% Cloud" in Figure 16). In this calculation, we classified NAT-ed providers as non-cloud peers. Nearly 95% of the content is provided by at least one cloud-based provider. At the same time, for 91% of the content, at least half of the providers are cloud-based and 23% of the content is provided only by cloud-based peers. On the other hand, an alternate interpretation posits that around 77% of the CIDs have at least one non-cloud provider.

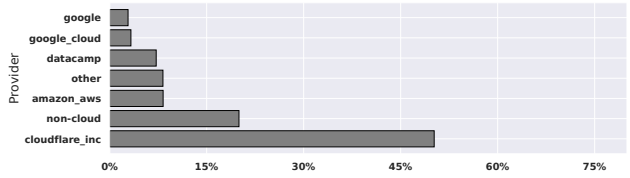
To sum it up, content hosting in IPFS is heavily reliant on cloud-based infrastructure. Cloud nodes are not only dominant in content hosting, but they also serve as relays for a large portion of NAT-ed content providers.

7 THE ENTRY POINTS

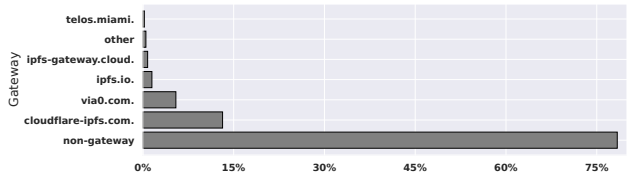
In this section, we investigate entry points to IPFS. Namely, HTTP Gateways and two systems mapping human-readable domain names to IPFS CIDs.[5]

DNSLink. DNSLink allows resolving domain names to CIDs using the DNS. However, each correct record must contain an HTTP gateway through which the CID can be fetched (Section 2). Figure 17 presents the distribution of these IP addresses across cloud providers. Similarly to our previous measurements, only 20% of the gateways are non-cloud nodes. This is understandable, as gateways require a public IP and high availability to operate efficiently. However, we observe different popularity of specific cloud providers. 50% of the IP addresses are hosted by Cloudflare alone. Cloudflare actively supports IPFS, for instance, by operating one of the most popular public HTTP gateways that can be easily used for DNSLink. Surprisingly, we observe only 21% of the IPs belonging to public gateway domains [40].

[5]IPFS also provides IPNS - one more way of mapping human-readable names to CIDs. We skip this mechanism as it is internal for IPFS and is equivalent to regular CID fetching already covered in Section 5.



(a) Cloud providers



(b) Gateways

Figure 17: DNSLink statistics for records pointing to IPFS content providers

Gateways. We investigate IPFS gateways themselves using the *gateway ID* and *passive DNS datasets*. We collect the domain names of public gateways and those referenced by DNSLink records. We distinguish between HTTP-facing (*i.e.* accepting HTTP requests) and overlay-facing (*i.e.* issuing requests to the IPFS network) IP addresses.

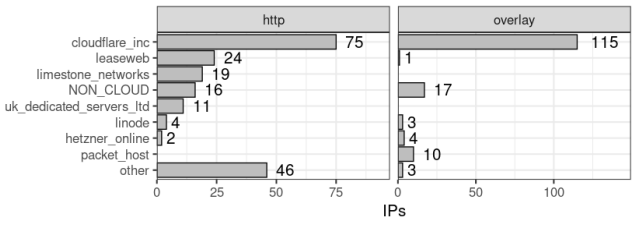


Figure 18: Unique IP Addresses of Gateway Frontends and Overlay Nodes by Cloud Provider.

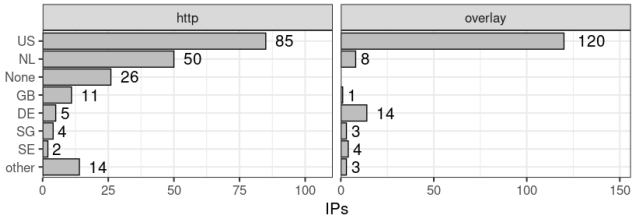


Figure 19: Unique IP Addresses of Gateway Frontends and Overlay Nodes by Geolocation.

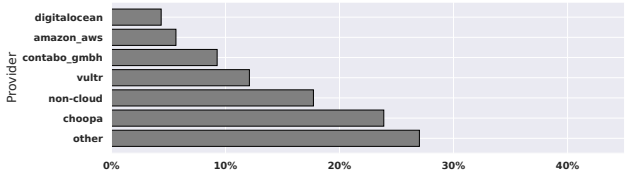
Similar to the DNSLink data, we find heavy reliance on Cloudflare (Figure 18). This is unsurprising for the HTTP frontends, as Cloudflare is commonly employed as a reverse proxy and protection service, a practice equally often criticized for creating central points

of failure [10]. Conversely, some of these gateways may actually be located in other autonomous systems, or hosted on different providers. However, we also find a large number of IPs provided by Cloudflare on the overlay side. It appears that these IPs are internal to Cloudflare, utilized to reverse proxy the *overlay* connections of their Gateways.[6] A notable number of gateways are running on non-cloud systems. This is commendable and potentially due to the open nature of the gateway ecosystem: In principle, anyone can operate a gateway and add themselves to the public register [40]. Note that, with the exception of Cloudflare, gateway providers generally do not operate their own AS.

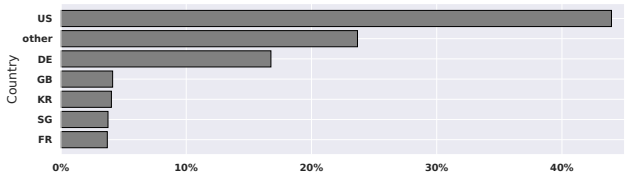
Geolocating both the HTTP frontend IP addresses as well as the overlay addresses using the MaxMind GeoLite2 [50] database, we find that the majority of nodes are situated in the United States and Germany (Figure 19), similar to the overall trends observed in the DHT (cf. Section 4). The large number of frontend IP addresses situated in the Netherlands could be a result of our vantage point in Germany. The geolocation of overlay addresses is unaffected by our vantage point.

Ethereum Name Service. Combining IPFS with ENS is a popular approach to scale content delivery in Web3. ENS records point to CIDs that can be later downloaded from IPFS. We extract a list of IPFS providers for each CID referenced by ENS and analyze them in the context of cloud providers (Figure 20.a). The results are consistent with the previous measurements: 82% of the content is hosted on cloud nodes with the main providers being *choopa*, *vultr* and *contabo*.

We then investigate the geolocation of the providers (Figure 20.b) and observe that the majority of content (60%) is concentrated in the US and Germany alone. Even with the ENS records being held on a blockchain, the actual retrieval of the referenced content is centralized and heavily dependent on a few cloud providers.



(a) Cloud providers



(b) Geolocation

Figure 20: Content provider statistics for IPFS content on ENS records (taking unique IPs)

[6]We are currently confirming this with Cloudflare.

8 RELATED WORK

DHT Measurements. Numerous studies investigated the state and performance of various DHTs implementation such as KAD [66, 70, 71] or BitTorrent DHT [11, 22, 36, 83]. In contrast, we focus on the multimodal analysis of multiple protocols used in IPFS.

IPFS. We add to a growing body of research on IPFS [7]. Henningsen *et al.* [31, 32] develop a crawler for the IPFS network we use for this work. Their DHT analysis from December 2019 finds lower overall robustness to node removal than our study. We attribute this to improvements in the DHT since 2019, most notably the removal of unconnectable leaf nodes. Balduf *et al.* [5] investigate privacy issues relating to unstructured Bitswap broadcasts. [13] focuses on network participants and their churn. Daniel *et al.* [12] provide a comprehensive overview of the IPFS ecosystem and its components but without performing any measurements. Trautwein *et al.* [78] gives an overview of the functionality of IPFS and measures its client population over an extended period of time. Other studies describe security vulnerabilities leading to eclipse [61] or censorship [28] attacks or investigate how IPFS can be used to spread malware [55].

Assessing Decentralization. Networks, both in nature and communications, have been analyzed for their properties and robustness before. In their seminal work, Albert *et al.* showcased how many complex networks, including the internet and social networks, are resistant to random failures due to being scale-free [1]. More recently, such analyses have been applied to Diaspora [8], and Mastodon [64] as well as the Bitcoin and Ethereum blockchains [27, 29]. Sadly, and similar to our study, those papers usually report a higher level of centralization than expected for their respective platforms. To the best of our knowledge, we are the first to analyze and assess the decentralization of IPFS in a comprehensive way.

9 DISCUSSION AND CONCLUSION

We find evidence of a heavy IPFS reliance on cloud infrastructure that is visible in the network topology, generated traffic, content provider records and its entry points. This dependency threatens the core design goals of the IPFS such as censorship resistance, robustness and openness.

One of the main challenges of IPFS remains the inability of NAT-protected nodes to fully participate in the network. It limits the number of full DHT participants, putting more pressure on the public-IP nodes. Furthermore, hosting content using proxies increases centralization and reliance on cloud nodes. IPFS supports a NAT-traversal protocol that requires assistance from a public-IP node only during the connection setup [60, 67], but NAT-punching clients still function as DHT clients only. However, in the long run, the wider deployment of IPv6, and thus the removal of IPv4 NAT, seems like a more sustainable solution.

The Hydra-booster nodes deployed by Protocol Labs were supposed to speed up the content resolution by acting as a large provider records cache and DHT query speed amplifier. They are thus cloud-based and operated by a single entity. On the other hand, the presence of Hydra-boosters is not an entirely bad thing as the DHT can still function properly, should the Hydra-booster nodes disappear.

A more concerning idea is the recent introduction of network indexers that are entirely hosted in the cloud [44]. The indexers gather information about all the content stored on IPFS and can resolve it much faster than the current DHT lookups. Content resolution is a core functionality of the platform and its control by a single entity gives its operator the power to block content (*e.g.* when pressured by the government).

In general, cloud-based resolution is always faster than decentralised lookup. Its deployment may be thus important for multiple latency-sensitive applications. At the same time, we strongly advise keeping the DHT as a fallback resolution mechanism to maintain the decentralization of the network. More research on more efficient DHTs (or similar resolution networks) is also needed to close the performance gap between the two solution classes.

Protocol Labs introduced multiple commendable solutions (*e.g.* Brave integration, HTTP Gateways, DNSLink) making the network easy to use for non-tech-savvy users and boosting its adoption. However, if done incorrectly, they can also introduce centralization. For instance, Brave users can currently choose between a self-hosted IPFS node and a default, cloud-based gateway. Changing the default gateway to a random one supported by a dynamic, permissionless discovery system could maintain simplicity while avoiding reliance on cloud infrastructure.

Currently, the IPFS network is used as a file transfer system rather than decentralized storage, based on the short lifetime of data items. At the same time, storage persistence is provided by a handful of centralized, cloud-based providers. Mechanisms such as Filecoin [62] could help to solve this problem by providing incentives for storing content. However, it is unclear whether decentralized storage nodes can compete with the service quality and prices of dedicated cloud providers. Furthermore, a decentralized swarm of peers cannot provide reliable storage guarantees without a reliable replication mechanism allowing some nodes to go offline while guaranteeing that the data is always available.

Overall, the IPFS design creates a solid foundation for a sustainable, efficient and decentralized storage network. We hope that by highlighting the current challenges, we can help to address them in future releases of the protocol.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions. The authors express their gratitude to the contributors of passive DNS data to the European Data Sharing Collective — SIE Europe. This work was supported by Protocol Labs under Grant No. PL-RGP1-2021-054. This work was also supported by the German Research Foundation (DFG) within the Collaborative Research Center (CRC) SFB 1053: MAKI (https://gepris.dfg.de/gepris/projekt/210487104). This work has been partially supported by the Cisco grant number 2020-216508 Hybrid-ICN Interoperability with IPFS.

REFERENCES

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 2000. Error and attack tolerance of complex networks. *nature* 406, 6794 (2000), 378–382.
[2] Inc. Audius. 2022. *Audius.* Retrieved May 26, 2023 from https://audius.org
[3] The DTube Authors. 2018. *DTube.* Retrieved May 26, 2023 from https://d.tube
[4] Leonhard Balduf, Martin Florian, and Björn Scheuermann. 2022. Dude, Where's My NFT: Distributed Infrastructures for Digital Art. In *Proceedings of the 3rd International Workshop on Distributed Infrastructure for the Common Good* (Quebec, Quebec City, Canada) *(DICG '22)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3565383.3566106
[5] Leonhard Balduf, Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. 2022. Monitoring Data Requests in Decentralized Data Storage Systems: A Case Study of IPFS. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. 658–668. https://doi.org/10.1109/ICDCS54860.2022.00069
[6] Balduf, Leonhard and Ascigil, Onur and Keizer, Navin V. and Pavlou, George and Scheuermann, Björn and Korczyński, Maciej. 2023. *Code for the IMC23 paper The Cloud Strikes Back: Investigating the Decentralization of IPFS.* https://github.com/mrd0ll4r/imc23-tcsb
[7] Juan Benet. 2014. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561* (2014).
[8] Ames Bielenberg, Lara Helm, Anthony Gentilucci, Dan Stefanescu, and Honggang Zhang. 2012. The growth of diaspora-a decentralized online social network in the wild. In *2012 proceedings IEEE INFOCOM workshops*. IEEE, 13–18.
[9] Cali Dog Security. 2022. *Certstream.* https://calidog.io
[10] Devin Coldeway. 2020. *Cloudflare DNS goes down, taking a large piece of the internet with it.* Retrieved May 26, 2023 from https://techcrunch.com/2020/07/17/cloudflare-dns-goes-down-taking-a-large-piece-of-the-internet-with-it/
[11] Scott A Crosby and Dan S Wallach. 2007. *An analysis of bittorrent's two kademlia-based dhts.* Technical Report.
[12] Erik Daniel and Florian Tschorsch. 2022. Ipfs and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Communications Surveys & Tutorials* 24, 1 (2022), 31–52.
[13] Erik Daniel and Florian Tschorsch. 2022. Passively Measuring IPFS Churn and Network Size. In *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 60–65.
[14] Dipanjan Das, Priyanka Bose, Nicola Ruaro, Christopher Kruegel, and Giovanni Vigna. 2022. Understanding Security Issues in the NFT Ecosystem. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 667–681. https://doi.org/10.1145/3548606.3559342
[15] dClimate Inc. 2021. *dClimate.* Retrieved May 26, 2023 from https://www.dclimate.net
[16] Alfonso De la Rocha, David Dias, and Yiannis Psaras. 2021. Accelerating Content Routing with Bitswap: A multi-path file transfer protocol in IPFS and Filecoin.
[17] district0x. 2018. *Ethlance.* Retrieved May 26, 2023 from https://github.com/district0x/ethlance
[18] D Dittrich and E Kenneally. 2012. *The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research.* https://catalog.caida.org/paper/2012_menlo_report_actual_formatted
[19] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. 2013. ZMap: Fast Internet-Wide Scanning and Its Security Applications. In *USENIX Security*. USENIX Association, 605–620.
[20] Dean Eigenmann and Nick Johnson. 2018. *ERC-1577: contenthash field for ENS.* https://eips.ethereum.org/EIPS/eip-1577
[21] EtherScan. 2023. *Ethereum Blockchain Explorer.* https://etherscan.io
[22] Jarret Falkner, Michael Piatek, John P John, Arvind Krishnamurthy, and Thomas Anderson. 2007. Profiling a million user DHT. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. 129–134.
[23] Filebase Inc. 2023. *Filebase - IPFS Made Easy.* Retrieved Sep 14, 2023 from https://filebase.com/
[24] Mozilla Foundation. 2007. *Public Suffix List.* Retrieved May 26, 2023 from https://publicsuffix.org
[25] The Swedish Internet Foundation. 2019. *Zone Data of .se and .nu TLDs.* Retrieved May 26, 2023 from https://internetstiftelsen.se/en/zone-data/
[26] Gala Games. 2020. *Gala Games.* Retrieved May 26, 2023 from https://app.gala.games/games
[27] Adem Efe Gencer, Soumya Basu, Ittay Eyal, Robbert Van Renesse, and Emin Gün Sirer. 2018. Decentralization in bitcoin and ethereum networks. In *Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22*. Springer, 439–457.
[28] François Genon, Sébastien Pierre, Etienne Riviere, and Michał Król. [n. d.]. An Eclipse attack on content availability in IPFS: a large-scale decentralized storage service. ([n. d.]).
[29] Arthur Gervais, Ghassan O Karame, Vedran Capkun, and Srdjan Capkun. 2014. Is bitcoin a decentralized currency? *IEEE security & privacy* 12, 3 (2014), 54–60.
[30] Ragib Hasan, Zahid Anwar, William Yurcik, Larry Brumbaugh, and Roy Campbell. 2005. A survey of peer-to-peer storage techniques for distributed file systems. In *International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II*, Vol. 2. IEEE, 205–213.
[31] Sebastian Henningsen. 2022. *Empirical and Analytical Perspectives on the Robustness of Blockchain-related Peer-to-Peer Networks.* Ph. D. Dissertation.

Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät. https://doi.org/10.18452/24401
[32] Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. 2020. Mapping the Interplanetary Filesystem. In *2020 IFIP Networking Conference (Networking)*. 289–297.
[33] ICANN. 2022. *Centralized Zone Data Service.* https://czdns.icann.org
[34] Liz Izhikevich, Gautam Akiwate, Briana Berger, Spencer Drakontaidis, Anna Ascheman, Paul Pearce, David Adrian, and Zakir Durumeric. 2022. ZDNS: A Fast DNS Toolkit for Internet Measurement *(IMC '22)*.
[35] Sagar Joglekar, Nishanth Sastry, Neil S Coulson, Stephanie JC Taylor, Anita Patel, Robbie Duschinsky, Amrutha Anand, Matt Jameson Evans, Chris J Griffiths, Aziz Sheikh, et al. 2018. How online communities of people with long-term conditions function and evolve: network analysis of the structure and dynamics of the asthma UK and British lung foundation online communities. *Journal of medical Internet research* 20, 7 (2018), e238.
[36] Sebastian Kaune, Ruben Cuevas Rumin, Gareth Tyson, Andreas Mauthe, Carmen Guerrero, and Ralf Steinmetz. 2010. Unraveling bittorrent's file unavailability: Measurements and analysis. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 1–9.
[37] Navin Keizer. 2021. *Deece.* Retrieved May 26, 2023 from https://github.com/navinkeizer/Deece
[38] Matters Lab. 2020. *Matters News.* Retrieved May 26, 2023 from https://matters.news
[39] Matters Lab. 2022. *Splinterlands.* Retrieved May 26, 2023 from https://splinterlands.com/
[40] Protocol Labs. 2017. *IPFS Public Gateway Checker.* Retrieved May 26, 2023 from https://ipfs.github.io/public-gateway-checker/
[41] Protocol Labs. 2018. *Discussify.* Retrieved May 26, 2023 from https://github.com/ipfs-shipyard/discussify-browser-extension
[42] Protocol Labs. 2019. *DNSLink.* Retrieved May 26, 2023 from https://docs.ipfs.tech/concepts/dnslink/
[43] Protocol Labs. 2020. *IPFS Ecosystem directory.* Retrieved May 26, 2023 from https://ecosystem.ipfs.tech/
[44] Protocol Labs. 2022. *Introducing the network indexer.* https://filecoin.io/blog/posts/introducing-the-network-indexer/
[45] Protocol Labs. 2022. *Space.* Retrieved May 26, 2023 from https://github.com/ipfs-shipyard/space
[46] Protocol Labs. 2023. *IPFS powers the Distributed Web.* https://ipfs.tech/
[47] Leonhard Balduf. 2023. *Privacy Policy for IPFS Monitoring.* https://monitoring.ipfs.trudi.group/privacy_policy.html
[48] libp2p. 2017. *Hydra Booster.* https://github.com/libp2p/hydra-booster
[49] RTrade Technologies Ltd. 2020. *Temporal.* Retrieved May 26, 2023 from https://temporal.cloud
[50] Maxmind. 2023. *GeoLite2 Database.* https://www.maxmind.com https://www.maxmind.com.
[51] Petar Maymounkov and David Mazieres. 2002. Kademlia: A peer-to-peer information system based on the XOR metric. In *International Workshop on Peer-to-Peer Systems (IPTPS)*. Springer.
[52] Danny R. McPherson, David R. Oran, Dave Thaler, and Eric Osterweil. 2014. Architectural Considerations of IP Anycast. RFC 7094. https://www.rfc-editor.org/info/rfc7094
[53] Almonit Organization. 2021. *Almonit.* Retrieved May 26, 2023 from https://almonit.eth.link/
[54] Craig Partridge and Mark Allman. 2016. Ethical Considerations in Network Measurement Papers. *Commun. ACM* 59, 10 (sep 2016), 58–64.
[55] Constantinos Patsakis and Fran Casino. 2019. Hydras and IPFS: a decentralised playground for malware. *International Journal of Information Security* 18 (2019), 787–799.
[56] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings 2019 Network and Distributed System Security Symposium*.
[57] Johan Pouwelse, Paweł Garbacki, Dick Epema, and Henk Sips. 2005. The bittorrent p2p file-sharing system: Measurements and analysis. In *Peer-to-Peer Systems IV: 4th International Workshop, IPTPS 2005, Ithaca, NY, USA, February 24-25, 2005. Revised Selected Papers 4*. Springer, 205–216.
[58] Ian Preston. [n. d.]. *Peergos.* Retrieved May 26, 2023 from https://peergos.org
[59] Protocol Labs. 2022. *Circuit Relay Protocol.* Retrieved May 26, 2023 from https://docs.libp2p.io/concepts/nat/circuit-relay/
[60] Protocol Labs. 2022. *Hole punching in libp2p - Overcoming Firewalls.* Retrieved September 12, 2023 from https://blog.ipfs.tech/2022-01-20-libp2p-hole-punching/
[61] Bernd Prünster, Alexander Marsalek, and Thomas Zefferer. 2022. Total Eclipse of the Heart–Disrupting the {InterPlanetary} File System. In *31st USENIX Security Symposium (USENIX Security 22)*. 3735–3752.
[62] Yiannis Psaras and David Dias. 2020. The interplanetary file system and the filecoin network. In *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 80–80.
[63] Manoharan Ramachandran, Niaz Chowdhury, Allan Third, John Domingue, Kevin Quick, and Michelle Bachler. 2020. Towards complete decentralised verification of data with confidentiality: different ways to connect solid pods and blockchain. In *Companion Proceedings of the Web Conference 2020*. 645–649.
[64] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. 2019. Challenges in the decentralised web: The mastodon case. In *Proceedings of the internet measurement conference*. 217–229.
[65] Rich Rosenbaum. 1993. Using the Domain Name System To Store Arbitrary String Attributes. RFC 1464. https://www.rfc-editor.org/info/rfc1464
[66] Hani Salah, Stefanie Roos, and Thorsten Strufe. 2014. Characterizing graph-theoretic properties of a large-scale DHT: Measurements vs. simulations. In *2014 IEEE Symposium on Computers and Communications (ISCC)*. 1–7. https://doi.org/10.1109/ISCC.2014.6912540
[67] Marten Seemann, Max Inden, and Dimitris Vyzovitis. 2022. Decentralized Hole Punching. In *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, 96–98.
[68] Ethereum Name Service. 2021. *ENS Documentation.* https://docs.ens.domains
[69] SIE Europe. 2022. *Passive DNS Data Sharing.* https://www.sie-europe.net
[70] Moritz Steiner, Ernst W Biersack, and Taoufik En-Najjary. 2007. Actively Monitoring Peers in KAD.. In *IPTPS*, Vol. 7. Citeseer, 26–27.
[71] Moritz Steiner, Taoufik En-Najjary, and Ernst W Biersack. 2007. A global view of kad. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. 117–122.
[72] George J Stigler. 1958. The economies of scale. *The Journal of Law and Economics* 1 (1958), 54–71.
[73] Daniel Stutzbach and Reza Rejaie. 2005. Evaluating the Accuracy of Captured Snapshots by Peer-to-Peer Crawlers.. In *PAM*. Springer, 353–357.
[74] Daniel Stutzbach and Reza Rejaie. 2006. Capturing accurate snapshots of the gnutella network. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*. IEEE, 1–6.
[75] SWITCH. 2017. *Zone Data of .ch TLD.* Retrieved May 26, 2023 from https://www.switch.ch/de/open-data/
[76] Berty Technologies. 2018. *Berty.* Retrieved May 26, 2023 from https://berty.tech
[77] The Invisible Internet Project. 2022. *Secure Semireliable UDP.* Retrieved September 12, 2023 from https://geti2p.net/en/docs/transport/ssu
[78] 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 *Proceedings of the ACM SIGCOMM 2022 Conference*. 739–752.
[79] trudi group. 2019. *IPFS Crawler.* https://github.com/trudi-group/ipfs-crawler https://github.com/trudi-group/ipfs-crawler.
[80] trudi group. 2022. *IPFS Network Size Estimates.* Retrieved May 26, 2023 from https://grafana.monitoring.ipfs.trudi.group/
[81] Udger. 2022. *Udger Data v3 – 20220606-01.* https://udger.com/ Retrieved 10 February 2023 from https://udger.com/.
[82] Christo Wilson, Bryce Boe, Alessandra Sala, Krishna PN Puttaswamy, and Ben Y Zhao. 2009. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*. 205–218.
[83] Scott Wolchok and J Alex Halderman. 2010. Crawling BitTorrent DHTs for Fun and Profit.. In *WOOT*.
[84] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.
[85] Pengcheng Xia, Haoyu Wang, Zhou Yu, Xinyu Liu, Xiapu Luo, Guoai Xu, and Gareth Tyson. 2022. Challenges in decentralized name management: the case of ENS. In *Proceedings of the 22nd ACM Internet Measurement Conference*. 65–82.
[86] Osman Yagan, Dajun Qian, Junshan Zhang, and Douglas Cochran. 2013. Conjoining speeds up information diffusion in overlaying social-physical networks. *IEEE Journal on Selected Areas in Communications* 31, 6 (2013), 1038–1048.
[87] ZorrillosDev. 2022. *Watchit.* Retrieved May 26, 2023 from https://watchit.movie/

A ETHICS

The topology and traffic 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. The Bitswap and HydraBooster datasets additionally contain personal information, as they cover CID requests and advertisements from the clients. However, 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.

We provide a user-facing website with our data management policy [47]. We use collected data only for purposes that are permissible with respect to Art. 6 (4) GDPR. We operate within the bounds of European data protection law, specifically Art. 5 (1) (e) GDPR. We are storing the data for up to 24 months or until we do not need it anymore to complete our research, whichever comes sooner. We are constantly reflecting on our data storage practices. If we conclude that some data fields or whole data sets are no longer needed for our intended purpose, we will delete them accordingly.

Part of our research is based on active DNS scans and follows industry best practices in network measurements [18, 19, 54]. To distribute the load across different authoritative nameservers, we have implemented randomization in our input list of domain names, avoiding simultaneous scanning of all domains under the same TLD. We expect that a portion of our DNS requests has been resolved from the internal cache of Cloudflare Public DNS resolver.

Furthermore, we tried to minimize the load caused by our research on the network. In IPFS, the provider records are stored on the 20 closest nodes to the CID hash. The original FINDPROVIDERS(CID) query performs a single DHT walk towards the CID hash and asks all the encountered nodes on the path about all the providers for the target CID. By default, this walk ends when either enough provider records have been found or when all 20 closest peers to the CID were queried. Our version of the FINDPROVIDERS(CID) call terminates once all 20 closest peers to the target CID are queried. In our analysis, we found that for 99.55% of the CIDs, our modified FINDPROVIDERS(CID) call discovered fewer than 20 providers and thus behaved exactly as the original FINDPROVIDERS(CID) call. Consequently, for the vast majority of CIDs, our modification incurred no additional network overhead. The additional overhead incurred by the remaining 0.45% of CIDs was limited to obtaining providers from at most 19 additional peers. Therefore, our modifications to the FINDPROVIDERS function were implemented to minimize any potential overhead and ensured that the vast majority of CIDs experienced no additional network burden.

For the IP geolocation, we used the MaxMind GeoIP database [50]. We downloaded and later locally queried their GeoLite IP database, which operates entirely offline on our machines. No IP information left our local machine.