The Eternal Tussle: Exploring the Role of Centralization in IPFS

Yiluo Wei[1] Dennis Trautwein[2] Yiannis Psaras[2] Ignacio Castro[3]
Will Scott[2] Aravindh Raman[4] Gareth Tyson[1]

[1]*Hong Kong University of Science and Technology (GZ)* [2]*Protocol Labs*
[3]*Queen Mary University of London* [4]*Brave Software*

Abstract

Web centralization and consolidation has created potential single points of failure, *e.g.,* in areas such as content hosting, name resolution, and certification. The "Decentralized Web", led by open-source software implementations, attempts to build decentralized alternatives. The InterPlanetary File System (IPFS) is part of this effort and attempts to provide a decentralized layer for object storage and retrieval. This comes with challenges, though: Decentralization can increase complexity, overhead, as well as compromise performance and scalability. As the core maintainers of IPFS, we have therefore begun to explore more hybrid approaches. This paper reports on our experiences building three centralized components within IPFS: (*i*) ***InterPlanetary Network Indexers***, which provides an alternative centralized method for content indexing; (*ii*) ***Hydra Boosters***, which are strategic DHT nodes that assist IPFS in content routing; and (*iii*) ***HTTP Gateways***, which are a public access point for users to retrieve IPFS-hosted content. Through this approach, we trade-off the level of decentralization within IPFS in an attempt to gain certain benefits of centralization. We evaluate the performance of these components and demonstrate their ability to successfully address the challenges that IPFS faces.

1 Introduction

Driven by powerful economies of scale, the centralization and consolidation of the web seems unstoppable [11, 19]. For example, website administrators will often choose to rely on large cloud platforms such as Amazon EC2, third-party DNS providers like GoDaddy, content delivery networks such as Akamai, and certificates issued by Let's Encrypt. These are well-engineered, performant services. Yet, they are also a potential single point of failure. Disruptions and outages in these centralized entities (*e.g.,* a cloud platform) can result in enormous financial losses, such as the reported loss of over $60,000 per minute by Amazon's eCommerce platform due to an outage [13, 15, 29, 33].

The "Decentralized Web" is a response to this growing concentration. The Decentralized Web refers to a group of technologies that aim to decentralize control away from major players. These technologies rely on open-source, community-led software implementations that decentralize traditional web functions, such as name lookup, object storage, and certification. As anyone can use and contribute to the software, the effort strives to reduce barriers to participation and reduce current trends towards web consolidation [19]. Note, we draw a clear distinction between *decentralized*, and *distributed*. Although many web system implementations are distributed, their control and operation remain centralized.

The *InterPlanetary File System (IPFS)* is part of this decentralization effort. IPFS is a storage layer for the Decentralized Web. It is a decentralized content-addressable object storage and retrieval platform. IPFS is a community-driven, open-source project, which covers 200 git repositories with 67809 stars and 12407 forks. In total, there are 83.5 K commits by 1352 code contributors, covering 400+ organizations including universities, start-ups and large corporations.

IPFS has seen widespread uptake with more than 1 B web client accesses and more than 250 k unique nodes participating in its peer-to-peer (P2P) network every week. Critically, IPFS 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) [2].

Although these figures point to the success of IPFS, our experience has shown that decentralization comes at a cost (§2.2). Decentralization can increase complexity and overhead due to the need to coordinate a large number of decentralized entities. Consequently, performance can be compromised, scalability can be challenging, and this can lead to a less practical system. Measuring, managing and debugging such systems is also more challenging. As part of the group of core maintainers, our operational experience has identified three key challenges faced by IPFS and similar systems:

1. **Massive content publication**: IPFS uses a distributed hash table (DHT) for content indexing and publication. However, our measurements show that publications can take over 1 minute, hindering the publication of large amounts of content. This is caused by the (*i*) decentralized routing process of node discovery in the DHT; and (*ii*) the need to replicate content indexing data across many nodes to mitigate the impact of churn.
2. **Content retrieval performance**: IPFS's overall retrieval delay is approximately 3x slower than HTTPS. By deploying a range of applications, we have found that IPFS's fully decentralized model is well suited to serving

delay-tolerant objects like file hosting, yet struggles with real-time applications such as live video streaming.

3. **Adoption**: The content-addressed IPFS protocols differ from traditional location-based protocols. This makes adoption difficult for both web developers and browser manufacturers, who mostly do not support IPFS retrieval. As IPFS retrievals tend to be slower than HTTPS, it also makes it difficult to integrate HTTP and IPFS objects within a single webpage. Further, full decentralization means that all clients should ideally install the IPFS node software. Yet, this requires technical skills, and IPFS software is currently too heavyweight for mobile devices. This presents barriers to adoption.

Our initial strategy was to tackle all these challenges in an entirely decentralized manner. However, practical constraints make this difficult, primarily driven by the inherent performance issues associated with DHTs, and the limited resources of (some) peers operating in the system. Thus, we have since been experimenting with the introduction of centralized components to complement the IPFS decentralized architecture. We argue that this can be an effective middle-ground, while decentralized technologies continue to be developed. The idea is to tradeoff the degree of decentralization of IPFS in order to attain potential benefits. Although this may result in a more efficient system, this also means that IPFS is partially centralized, raising certain issues which require exploration. For example, users may interact exclusively with centralized components during object retrieval, placing significant additional power in the hands of operators. This brings risks related to security, robustness, privacy, alongside raising philosophical questions related to the role of centralization in the Internet. Thus, we strive to design centralized components that are optional, allowing the system to continue operation even in the failure of centralized aspects (albeit with degraded system properties). With this tussle in mind, we report on our operational experiences at Protocol Labs in deploying a set of three complementary centralized components within the wider IPFS network.

First, we present the **InterPlanetary Network Indexers**. These provide an alternative centralized method for content indexing and lookup. Each Indexer acts as a server-based key store, complementing the role of the fully decentralized DHT. This avoids the overheads associated with peer-to-peer routing. Importantly, any entity can operate an Indexer with the ability (and expectation) to synchronize indexing data, thereby reducing the risks of consolidation. Second, we present the **Hydra Boosters**, a small set of high-performance DHT nodes that assist IPFS in content routing, content provision, and peer routing. The Hydra Boosters are strategically spread across the DHT key space and try to provide one-hop access to data records. Third, we present the **HTTP Gateways**, a set of proxy servers that offer an HTTP bridge into the IPFS network. These help reduce barriers to adoption by allowing clients to access IPFS content without installing the fully decentralized IPFS node software. Additionally, the centralized gateways benefit from aggregation in demand, allowing us to deploy caching of content in a way that was not possible when operating in a fully decentralized fashion. No privileges are required to establish these components, and anybody can contribute to the effort.

The contributions are as follows:

1. We present the design and implementation of three new centralized components deployed within the IPFS infrastructure. For each component, we explore the challenges it addresses and how we have tackled those challenges.
2. Based on real-world operational data, we present an evaluation of the three complementary components. We demonstrate that these centralized components do bring large benefits to IPFS, addressing the challenges discussed above.
3. We explore security, privacy, and other risks that could arise from the centralized nature of these components. We discuss the associated trade-offs that must be considered when designing hybrid solutions such as IPFS.

2 Background and Motivation

2.1 IPFS Fundamentals

We start by providing a brief overview of IPFS. We redirect interested readers to [41] for full details.

Content Addressing. IPFS is a decentralized object store. Much like prior information-centric networks, it uses self-certifying hash-based Content Identifiers (CIDs) to decouple content names from their storage location. When content is added to IPFS, it is split into chunks and each chunk is assigned its own CID, which is the result of hashing its content and adding metadata. These CIDs are then used to construct a Merkle Directed Acyclic Graph (DAG) of the file. The root node of the DAG combines all the CIDs of its descendant nodes and forms the final content CID, which allows for chunk de-duplication and eliminates the need to store or transmit the same content twice. Merkle DAGs are agnostic to where the content is stored, so they do not need to be updated when a file is replicated on or deleted from nodes in the network.

Peer Addressing. Each peer has a unique ID, generated as a hash of its public key, and represented as a Multihash [7]. A multiaddress is then associated with each PeerID. Multiaddresses are a simple data structure that allows multiple protocols and address types to be included for each peer.

Content Indexing. IPFS relies on a fully decentralized Kademlia DHT for content indexing. The DHT does not store content itself but, instead, hosts (*i*) *peer records* that map PeerIDs to the multiaddresses that can be used to contact the peer; and (*ii*) *provider records* that map the CIDs of content to the PeerIDs of the peers who provide the content. This indexing allows clients to map their desired content to a peer

that can provide it and to discover the multiaddresses that can be used to contact the peer.

Content Publication. To publish content, the host first generates a provider record that maps the CID to its own Peer ID. It then pushes it to the DHT. To ensure availability, this record is stored on the 20 closest peers in terms of their PeerIDs' XOR distance from the SHA256 hash of the CID. Note, peers retrieving the content can volunteer to become temporary or permanent content providers by publishing a provider record pointing to their own node on the DHT. By doing this, it avoids the original source becoming a single point of failure.

Content Retrieval. Content retrieval involves four steps: content discovery, peer discovery, peer routing, and content exchange. Content discovery is the process of looking up the provider record using the CID of the content. Before entering the DHT lookup, the requesting peer asks all peers it is already connected to for the desired content in an opportunistic fashion, using a protocol called BitSwap [3]. If this initial attempt is not successful, content discovery falls back to the DHT. After getting the PeerID of the provider from the provider record, peer discovery then involves querying the DHT and retrieving the PeerID's peer record. Recall, the peer record contains the peer's multiaddresses, listing the protocols and physical addresses that can be used to reach the node. To streamline this process, each IPFS node maintains a local address book of up to 900 recently seen peers. Once the PeerID is resolved to a peer record, the requesting node uses the list of Multiaddresses to connect to the desired peer, a process called peer routing. Finally, content exchange is carried out using the Bitswap protocol. Importantly, all the above steps take place across a fully decentralized infrastructure.

2.2 Challenges of Decentralization in IPFS

Our experience in operating the above decentralized setup has highlighted three key challenges that we discuss in this paper.

Massive Content Publication. IPFS utilizes a Kademlia distributed hash table (DHT) for publishing and locating content. Although Kademlia enables seamless distribution, this brings additional overhead and high delay when compared against simple centralized key stores (*e.g.,* SplinterDB). To highlight this, Figure 1 shows the content publication time as measured in our experiments (detailed in §4.2). The overall publication process across all regions takes 11.81s, 40.81s, and 66.73s at the 50th, 90th, and 95th percentiles, respectively. This is a clear cost of decentralization, which is not experienced by well-resourced centralized database lookups The delay is dominated by the DHT walk to find the nodes to publish provider records to (covering 83.37% of the overall delay), due to the need to distribute records to 20 different peers. Furthermore, the IPFS DHT requires re-publication of records every 24 hours. In practice, this takes an excessive amount of time, generates large traffic volumes, and consumes a substantial amount of storage space for peers to store provider records.

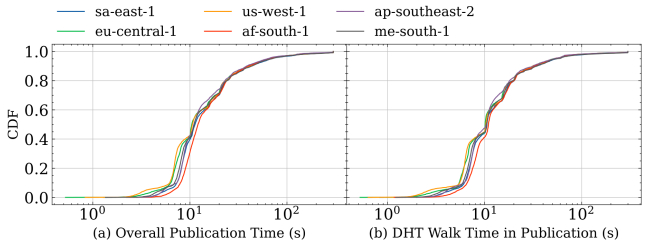

Figure 1: CDF for content publication for each AWS region: a) The overall publication duration, and b) the DHT walk.

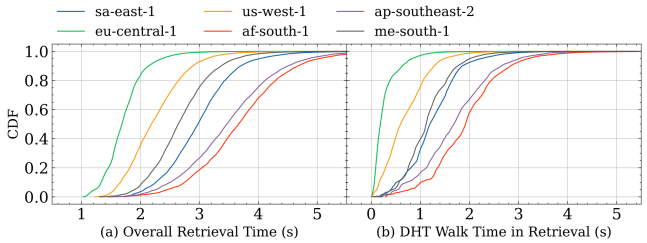

Figure 2: CDF for content retrieval for each AWS region: a) The overall retrieval duration, and b) the DHT walk.

Content Retrieval Performance. The decentralized nature of IPFS results in slower content retrieval speeds compared to well-resourced centralized systems. This is because all content must first be mapped to an appropriate source, via (several) routing hops in the DHT. Figure 2 shows the content retrieval time measured in our experiments (detailed in §4.2). The overall retrieval process across all regions takes 2.72 seconds, 4.03 seconds, and 4.42 seconds in the 50th, 90th, and 95th percentiles, respectively. This is slower than HTTP and is not suitable for certain delay-sensitive applications, such as live video streams. Moreover, the decentralization of storage across many independent nodes hinders the aggregation of demand, making it difficult to use techniques such as caching to improve the retrieval of frequently accessed content.

Adoption. To access IPFS-hosted content, users must run the IPFS node software, thereby participating in the storage and distribution of files. This is vital for the self-scaling properties of our decentralized setup, ensuring that no single node becomes overly powerful. However, setting up an IPFS node requires specific skills, which is a barrier for some users. Due to its footprint, the IPFS node software cannot yet run on mobile devices. This is problematic because 58% of website traffic comes from mobile devices, and approximately 92% of Internet users access the web using their smartphone [4]. There are also adoption challenges for web developers wishing to embed IPFS-host content within traditional HTML/HTTP websites. This is because IPFS retrievals tend to be slower than HTTP, creating usability issues with mixing HTTP and IPFS-hosted objects. Unfortunately, few browser implementations can retrieve IPFS content, further disincentivizing integration.

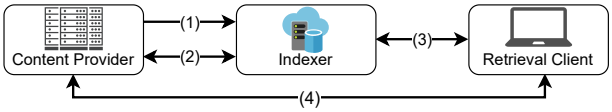

Figure 3: The publication and retrieval steps for Indexers. (1) Content provider publishes announce message. (2) Indexer synchronizes advertisements with the content provider. (3) Retrieval client queries Indexer using CID or multihashes and gets the provider record. (4) Retrieval client gets data from content providers according to the provider record.

3 Design and Implementation

To tackle the above challenges, we have experimented with deploying three new centralized elements to IPFS. These are all open-source, allowing anybody to set them up and contribute to IPFS. Through this, we strive to address the above challenges. These centralized components are:

1. ***InterPlanetary Network Indexers*** provide an alternative centralized approach to content indexing, complementing the DHT. It primarily targets the massive content publication challenge by making it faster to push new provider records. It also enhances content retrieval performance by making lookups faster.
2. ***Hydra Boosters*** introduce strategically placed reliable routing nodes in the DHT, to improve performance in a coordinated manner. This offloads work from the remaining nodes. Thus, the primary goal of Hydra Boosters is to enhance publication routing performance, as well as retrieval performance. It does this in two ways: (*i*) by hosting a large number of provider records in a stable fashion; and (*ii*) by providing stable routing nodes with the DHT.
3. ***HTTP Gateways*** provide an HTTP bridge into the IPFS network. Its primary focus is on improving adoption, by allowing access to IPFS without a full stack IPFS installation. Further, gateways also serve as a centralized point of request aggregation, enabling us to improve content retrieval performance via caching.

Although Protocol Labs has built and deployed these components, we emphasize that any stakeholder could adopt their usage. As such, whereas each individual component is centralized, there can be many instantiations by different operators. The rest of this section describes the components in detail.

3.1 InterPlanetary Network Indexers

Recall that the first challenge of decentralization in IPFS is massive content publication. We define this as the ability for providers to publish large numbers (*i.e.,* millions) of objects in an efficient fashion. This is currently difficult because IPFS's decentralized routing requires protocol exchange with *at least* 20 nodes to publish each object. Thus, in contrast to a centralized index, the overhead is dominated by the DHT hops that must be undertaken to publish provider records.

Overview of Indexers. To address this issue, we have built the InterPlanetary Network Indexers (aka "the Indexers") to complement the DHT. Put simply, an Indexer is a high-performance keystore server that indexes provider records. The provider records in the Indexers differ from those in the DHT. They comprise the identity of the content provider, its physical address, and the protocols required for retrieving the data. As a result, there is no need to resolve the PeerID to a physical address. Providers can push their provider records directly to the Indexer, and clients can directly retrieve the records from the Indexer using the appropriate CID. As this can be done within a single protocol exchange, it substantially reduces the overhead of storing provider records with the decentralized DHT. Importantly, the Indexer is optimized for bulk publication with certain requirements that publishers must adhere to. Hence, it is recommended for large content providers (and not for ordinary users) to publish via an Indexer. We emphasize that the Indexers do not replace the DHT but, instead, complement it by offering an additional content indexing approach. Figure 3 illustrates an overview of the publication and retrieval process, as detailed below.

Preparation of Advertisements. Each provider is required to record their own available content using a chain of immutable *advertisements*. An advertisement is a data structure asserting the publication or deletion of content by a given provider. An advertisement may contain multiple provider records. Importantly, by constructing an immutable chain of advertisements, it becomes possible for any Indexer to audit what content has been provided over time by each provider. Note, to achieve incremental verifiability, all advertisements are signed by the provider. To publish content, the provider adds a new advertisement to its chain and notifies the Indexers.

Sending Announcement Message. To notify the Indexers of newly published content, the provider sends an announcement message (Figure 3 (1)). An announcement message contains the CID of the advertisement, and the publisher's address (where to retrieve the advertisement from). By default, the announcement messages are sent via Gossipsub [42] (a P2P protocol to broadcast messages to nodes in a network). Through this, the providers gossip announcement messages among all Indexers. The Indexers can also receive such messages via an HTTP RESTful API call. The announcement messages can be shared among Indexers when configured to do so: they can re-publish HTTP announcements to other Indexers, and relay Gossipsub announcements.

Synchronize Advertisements. Once an Indexer receives an announcement message from a provider, it connects to the provider and traverses the advertisement chain to construct the up-to-date provider records (*i.e.,* advertisement synchronization, Figure 3 (2)). Since the advertisements are immutable,

the Indexers can recognize which portions of the chain are new and only traverse those sections. Once the provider records have been constructed, we use Pebble [8], a high-performance key-value store, to index all records.

Note, unlike the IPFS DHT, which only allows one file to be published at a time, the Indexers allow multiple provider records to be packed into a single advertisement. Additionally, content does not need to be repeatedly republished by providers, because there is no time-to-live for records in Indexers. Instead, only changes (additions or removals) are published, and the Indexers can reconstruct all provider records based on the advertisement chain. This makes the Indexer unsuitable for ephemeral publishers. Indexers must, therefore, configure their own policies for the removal of stale content. This design is based on our prior experiences with scalability and is optimized for bulk publication, directly addressing the previous massive publication challenge. Specifically, batching large amounts of provider records within a single advertisement makes it highly efficient to publish multiple objects at once and avoids republishing unless changes occur.

Retrieval of Content. Clients that wish to retrieve provider records interact with the Indexers via a RESTful HTTP query API. The identities of Indexers are currently public, with a default set distributed with client binaries. Clients can issue queries containing a single CID or several CIDs. In both cases, the Indexers return a list of provider records that match the query (Figure 3(3)). Retrieval clients can then retrieve the content directly from the providers based on the provider records (Figure 3(4)). To exploit the aggregation of queries at the Indexers, they utilize a frontend server to perform caching on the HTTP API calls. This caching stores frequently accessed provider records, allowing these requests to be directly served without querying the Indexer's Pebble database. This further reduces response latency and enhances overall performance.

3.2 Hydra Boosters

The second challenge of decentralization in IPFS pertains to content retrieval performance. This is because the DHT walk often needs to pass through multiple intermediate nodes, which can take a considerable amount of time.

Overview of Hydra Boosters. To address this issue, we have developed a set of Hydra Boosters. These are highly connected DHT nodes, which offer "shortcuts" through the routing space to provider records. They also help address the challenge of massive content publication, as they accelerate routing for publishing records.

The Hydra Boosters consist of: (*i*) Hydra Head DHT nodes, comprising thousands of virtual nodes in the DHT, such that every other peer has at least one Hydra Head in its routing table; and (*ii*) A shared Hydra Database that stores all provider records in the DHT and can be accessed by all Hydra Heads. This allows Hydra Boosters to serve other peers with the provider records immediately through the directly connected Hydra Heads, thereby improving content retrieval performance. Our key idea is that it does not require IPFS itself to adopt any infrastructure that is hard-coded into the protocol, or controlled by a few parties. Instead, Hydra Head nodes operate at the same level as any other node in the network, without any privileged status, making it a complement to the regular content routing operation.

Hydra Heads. The Hydra Boosters introduce the concept of Hydra Heads. Each head is perceived by other network participants as a conventional, distinct peer without special privileges. Each head has its own unique PeerID, routing table, and peer store. However, these heads are strategically positioned within the Kademlia XOR keyspace. The strategic positioning strives to make sure every peer has at least one Hydra Head within its 20-peer XOR proximity. If this goal is achieved, every provider record should be stored on at least one Hydra Head. Thus, giving the Hydra Boosters full vantage on all provider records. To further accelerate performance, records are stored in a distributed database shared across *all* Hydra Heads. As a result, any DHT walk to find a provider record will have a high chance of immediately getting the result from the Hydra database through the nearest Hydra Head in the routing table.

Peer ID Assignment. The main design challenge is to achieve this optimal placement. To address this, it is necessary to determine the total number of participating network peers and generate an appropriate set of peer identities to cover the entire keyspace. Initially, we considered relying on randomly generated peer identities, but in practice, this resulted in an uneven distribution. To mitigate this issue, we rely on the "power of two choices" [31]. This relies on a biased random algorithm and has been demonstrated to be effective at balancing loads when each load balancer has an incomplete or delayed view. First, all Hydra Head PeerIDs are tracked in an XOR trie. For each new head's PeerID, two choices are generated. The one that decreases the trie's depth the least is selected. This approach ensures an average depth of $\log N$, with a maximum depth of $\log N + \log\log N$. Importantly, the trie's depth correlates with proximity to existing nodes. If a new Hydra head node were inserted at depth D, then its closest node is at a distance 2^{-D}. Having all nodes at similar depths means they are implicitly equidistant in the XOR metric.

Hydra Database. The Hydra Booster system stores recent records in a shared Amazon DynamoDB key-value store. All heads can access this database, allowing every individual head to have full vantage on all provider records known by any head. Importantly, recall our earlier distinction between decentralized and distributed. Although DynamoDB is distributed, it sits under the control of Protocol Labs. Note, the Hydra database is not related to the peer routing functionality of Hydra heads. Thus, even if DynamoDB fails, the heads can continue to assist with peer routing similarly to any other node (although they cannot directly serve provider records).

3.3 HTTP Gateways

The third challenge of decentralization in IPFS pertains to adoption, particularly when compared to the simplicity of browser-based access using HTTP. Accessing IPFS content necessitates running an IPFS node, which requires technical expertise. Further, due to the computational cost, the implementation is currently not compatible with mobile systems. Additionally, due to the use of incompatible protocols, integrating IPFS with conventional (HTML/HTTP) websites is challenging. In part, this is also because IPFS experiences longer retrieval delays than traditional client-server HTTP, creating usability challenges for composing a webpage from both HTTP and IPFS-hosted objects.

Overview of Gateways. To address this issue, we have deployed a small set of gateway servers. A gateway works as a simple HTTP bridge into the IPFS network. We implement this as a Nginx web server front end, co-located with a full IPFS node installation. Clients interact with a gateway by issuing a `GET` request using the CID as a URL parameter. The gateway then retrieves the requested object from the IPFS network on behalf of the client before returning it via the HTTP connection. Note, as our software is open-source, any entity can deploy their own gateway. Indeed, other organizations such as Cloudflare have done precisely that.

Caching. To exploit the centralized aggregation of demand, and improve retrieval performance, we also enable caching on the gateways. Specifically, we use Nginx's default web cache with a Least Recently Used replacement strategy. Thus, any requested objects are cached within the default HTTP cache of the gateways. The gateways also allow authorized third parties to actively push content objects into its local object stores (*i.e.,* its node cache). To date, this is used by NFT.storage and Web3.storage. Web3.storage stores over 40M objects, while NFT.storage stores over 91M objects. This serves as a significant benefit of centralization: By aggregating user requests within the gateways, we can serve popular objects from the cache, avoiding repeatedly retrieving the same content from the IPFS network (as the gateway only forwards a request when it is not stored or cached locally). Alongside improving performance for the client, it also offloads unnecessary requests from the remainder of the network.

4 Evaluation Methodology & Data

To explore the efficacy of the above centralized components and their impact on IPFS's decentralization efforts, we rely on operational data collected from the components managed by Protocol Labs. See Appendix A for ethics discussion.

4.1 InterPlanetary Network Indexer Data

The below IPNI datasets are obtained from the Indexer managed by Protocol Labs (`https://cid.contact`).

Providers and Provider Records. Our first dataset covers the indexed records in the Indexer's data storage. As of `2023-04-24 UTC` , the dataset consists of 173,998,039,712 (approximately 0.17 trillion) provider records.

Index Ingestion Performance Data. Our second dataset covers the performance of our Indexer. We collect data from the operational logs of the Indexer. Because the Indexer processes approximately 5.3 billion provider records per day, we take a sample of advertisement synchronization operations covering 16,757,817 provider records on `2023-04-24`.

Index Query Performance Data. Our third dataset covers the Indexer query performance. The Indexer employs Amazon CloudFront. Our dataset comprises HTTP requests made to Amazon CloudFront, along with cache hit information and the request complete time. The dataset covers a period of one day, from `2023-04-18 09:05 UTC` to `2023-04-19 09:05 UTC`. During this time, 62,611,156 successful requests were made, and 246.77 GB of traffic was served.

4.2 Hydra Boosters Data

Protocol Labs has deployed 135 Hydra Booster nodes on Amazon ECS in the `us-east-1` region. Each of these Hydras has between 10 and 15 heads, resulting in a total of 2,015 Hydra heads.

Active Retrievals with Hydra Boosters. To evaluate the performance benefit of the Hydra Boosters, we perform a set of active measurements with the Hydra Boosters enabled. We utilize six machines that are located in different regions on Amazon AWS (see Table 1). On each machine, we run an IPFS DHT server node (Kubo implementation v0.16.0).

In each iteration, we randomly select a single node to publish a new 0.5 MB object (CID) to the network. Subsequently, all other nodes retrieve that object. Once all the nodes complete this process, they disconnect to prevent the next retrieval operation from being resolved through Bitswap. It is important to note that this is the closest one can get to a controlled test in the public IPFS network.

Active Retrievals without Hydra Boosters. To evaluate the benefit of Hydra Boosters, it is also necessary to perform the above measurements *without* the Hydra Boosters (as a baseline). Therefore, we repeat the above active measurements, whilst performing a *controlled deactivation* of Hydra Boosters. Specifically, we unplug the common database from the Hydra Boosters for a period of 7 days. We leave Hydra Heads in the network but configure them to behave differently:

- ADD-PROVIDER queries are ignored. Since Hydra Heads make about 10-15% of the network, this means we are artificially decreasing the k-replication to 19 [30].
- GET-PROVIDER queries are answered with only closer peers (*i.e.,* no provider records) because the common database is not there anymore.

The active retrieval experiment starts on `2022-11-25 15:27`

UTC to 2022-12-01 17:30 UTC. We have deployed the change of unplugging the database from the Hydra Boosters on 2022-12-01 17:30 UTC, thus the active retrieval experiment without Hydra starts on 2022-12-01 17:30 UTC to 2022-12-08 20:00 UTC. This gives us approximately 6 days of data with the Hydra Booster database, and 7 days without it. Table 1 summarizes the number of retrievals from each region before and after unplugging the database.

	Retrieval		Publication	
	with	without	with	without
af_south_1	9590	12084	2579	3284
ap_southeast_2	10375	13143	2579	3282
eu_central_1	10180	13233	2578	3289
me_south_1	10183	12814	2579	3280
sa_east_1	10192	12904	2578	3291
us_west_1	10162	12821	2579	3286

Table 1: The number of retrievals and publications from each region with and without the Hydra Boosters database.

4.3 HTTP Gateway Data

Finally, we gather a dataset from the public IPFS gateway (https://ipfs.io) managed by Protocol Labs. The dataset comprises of all HTTP GET requests made on 2023-03-15 UTC+0. Each record in the dataset represents the request and response details, covering remote IP address, request timestamp, user agent, HTTP referrer, request complete time, response size, response latency, and cache hit/miss information. In total, the dataset covers 69,339,954 successful requests from 1,645,611 IP addresses, with response sizes equalling 91,869 GB.

5 Evaluation

We now assess the effectiveness of our centralized components in addressing the challenges outlined in Section 2.2. We focus on (*i*) The effectiveness of the Indexers in supporting massive content publication; (*ii*) The effectiveness of the Indexers, the Hydra Boosters, and gateways in improving content retrieval performance; (*iii*) The effectiveness of the gateways in increasing the adoption of IPFS.

5.1 Challenge 1: Massive Content Publication

The first challenge of IPFS's decentralization pertains to the massive publication of objects. This is caused by the need to traverse many decentralized routing hops and replicate provider records to mitigate churn. To overcome this, we have introduced the Indexer, a centralized key store. Therefore, we start by evaluating the efficacy of our centralized components in improving the publication process.

Indexer Adoption. We first investigate the adoption of Indexers by content providers, by checking the number of provider records that have been uploaded to our Indexer server. Figure 4a illustrates the total number of provider records in the Indexer and IPFS DHT, alongside the number of provider records offered by the top 100 content providers in both sub-systems. We get the DHT data from our Hydra Booster database, as discussed in §3.2. The Indexer stores 173,998,039,71 provider records, over 100x more than the number stored in DHT. Additionally, we observe that the top providers in the Indexer offer 100-1000x more provider records than the top providers in the DHT, with most of them hosting around 10^9 provider records. Initially, one might assume that the Indexers have therefore eclipsed the DHT, and created a fully centralized environment. Closer inspection reveals a more nuanced situation though. Although the Indexers host many records, these are uploaded by just 604 major providers, with most of them running NFT or storage services. In contrast, 56k providers use the DHT, confirming the majority still choose to use the DHT. This confirms that large content providers *do* see benefit in using the Indexer system, while outside of these large providers, the remaining IPFS users continue to publish via the DHT.

We further examine the adoption of Indexers by retrieval clients. Our data shows that the Indexer receives 2.5k lookup requests per second, while the IPFS DHT receives around 5k (estimated using the number of GET-PROVIDER requests received by Hydra Boosters). These statistics confirm that, although the DHT plays a larger role, the Indexers successfully support massive content publication activities by this small number of major publishers. Overall, the results indicate that the Indexer is fulfilling its intended purpose of supporting massive content publications from large content providers and that clients do have interest in such content.

Publication Performance. The above confirms the adoption. We next assess if the Indexer indeed improves content publication performance. Figure 4b plots the CDF of advertisement synchronization time for the Indexer and compares it with the DHT publication time. We see that the advertisement synchronization time is faster than the DHT publication time at all percentiles. It is approximately twice as fast at the 50th percentile. Unsurprisingly, this confirms that the performance of the centralized Indexer is faster than the DHT publication.

However, an advertisement can contain multiple provider records (see §3.1). This implies that the time taken by the Indexer to index each provider record may be significantly shorter than what is depicted in Figure 4b. To explore this, Figure 4c presents the relationship between the synchronization time of advertisements and the number of provider records in the advertisement. Indeed, we find that the majority of advertisements contain 16K to 20K provider records (compared to 1 record for the DHT publications). Additionally, the synchronization time of advertisements is almost agnostic to the number of provider records, confirming that the Indexer is well suited for bulk publication.

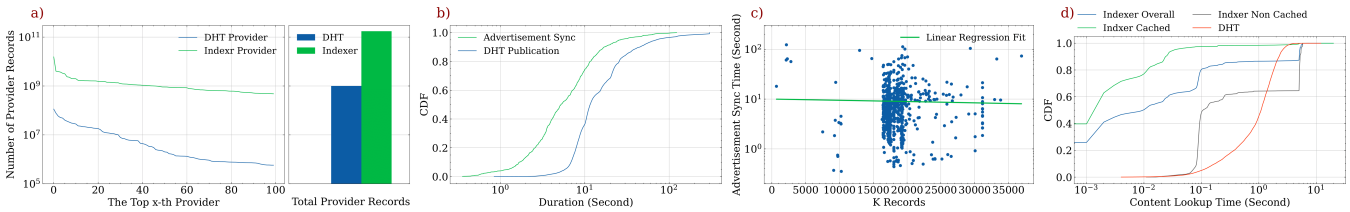

Figure 4: (a) The number of provider records offered by the top 100 content providers in the Indexer and IPFS DHT, alongside the total number of provider records in the Indexer and IPFS DHT; (b) CDF of the advertisement synchronization time for the Indexer, and the DHT publication time of an IPFS node for comparison; (c) Advertisement synchronization time vs. the number of provider records in the advertisement; (d) CDF of the content lookup time of the Indexer, and of the DHT for comparison.

Discussion & Takeaways. The above shows that Indexers effectively support massive content publication, with 604 large content providers hosting provider records 100x more than those in the DHT. Moreover, it offers a significantly faster indexing speed of approximately 500μs per provider record.

However, the Indexers also introduce centralization that brings certain trade-offs. Most noteworthy is the significant monetary cost of operating Indexers. In our experience, an Indexer server costs \$5–10K in capital expenditure, with approximately \$1K per month in operating expenditure. In contrast, IPFS's original publication mechanism (via the DHT) is designed such that the load is spread evenly (thereby removing the need for individuals to host expensive hardware). Moreover, the Indexer introduces a single point of failure as centralized failures in the Indexer can make significant volumes of provider records inaccessible. This is because some provider records are only stored within Indexers and is not replicated in IPFS DHT. We further discuss this in §6.

5.2 Challenge 2: Content Retrieval Performance

The second challenge of IPFS's decentralization pertains to content retrieval performance. This is driven by the need for clients to perform distributed lookups across (many) nodes. The Indexer, Hydra Boosters, and gateways are all geared towards addressing this, by complementing the distributed lookups and retrievals with centralized acceleration. We use the content lookup time as a metric to evaluate content retrieval performance. It refers to the duration between the point when the retrieval client launches a query and the point when the content is found. Across the three components, this metric corresponds to the time to get the first provider record in the DHT walk; the request completion time in an Indexer query; and the response latency in a gateway query.

Indexer Content Lookup Time. Figure 4d plots the CDF of the content lookup time of the Indexer. We also plot the content lookup time in a vanilla DHT lookup for comparison, as measured in our Hydra Booster experiments (§4.2). As expected, the Indexer's overall content lookup time is significantly shorter than the decentralized DHT lookup time. Our Indexer servers achieve approximately 10x better performance than the DHT lookup time at the 50th percentile.

One reason for the improvement is that the centralization allows us to exploit caching in the Indexer server (§3.1). On average, the cache hit rate is 65.22%. This means that the majority of queries hit the Indexer's cache, which is approximately 100x faster than the DHT lookup time in the 50th percentile. This is a significant benefit of centralization, as it allows us to aggregate demand into a small set of central points. Doing the same across our DHT is much less effective. We also note that, even for the 34.78% of requests that result in a cache miss, the content lookup times still outperform the DHT, as shown by the black line in Figure 4d. The performance around the 65th percentile is due to an optional cascading lookup which makes the Indexer query the IPFS DHT. Thus, even without the cache mechanism, the centralized Indexer still offers better content retrieval performance than the DHT.

Hydra Booster Lookup Time. The Hydra Boosters also improve content retrieval performance by shortening the DHT's provider record lookup. To evaluate this, recall that we performed a controlled experiment running IPFS both with and without the Hydra Booster active (see §4.2).

Figure 5a displays the content lookup time across all probed AWS regions, with and without the Hydra Booster database. We observe a clear performance improvement in `us-west-1`, `af-south-1`, `ap-southeast-2`, `sa-east-1`, and `me-south-1`, with the median improved by 36.5%, 25.1%, 16.4%, 11.7%, and 3.8% respectively. Although this is not as impactful as the Indexer speedups, it still constitutes a notable improvement. However, there is one outlier: the Hydra Booster database provides no performance improvements in `eu-central-1`. Our investigation suggests that this is because the retrieval performance in Europe is already optimized. As shown in Figure 5a, it is significantly better than in other regions, leaving limited room for further improvement. This suggests that the decentralization vs. centralization trade-off may differ based on region, and raises questions about whether such speed-ups warrant the high monetary cost.

We further investigate how these speedups are achieved. Recall, the Hydra Boosters improve performance by: (*i*) the strategic placement of stable Hydra Heads across the DHT

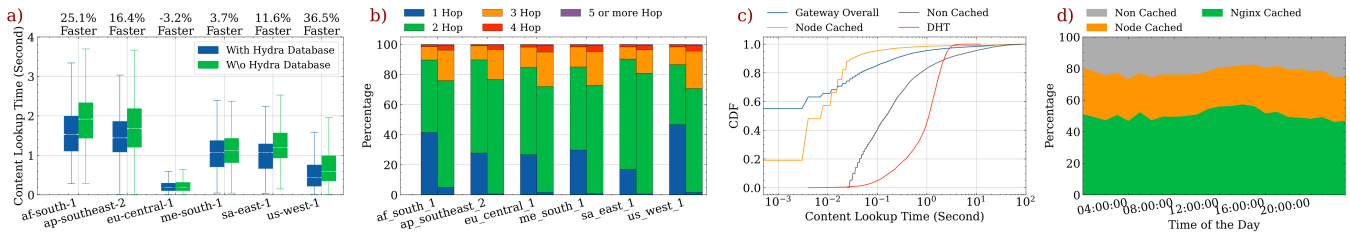

Figure 5: (a) The content lookup time across all probed AWS regions with and without the Hydra database, the values at the top refer to the medians; (b) Breakdown of the hop count per provider record lookup across all probed AWS regions with (left bar) and without the Hydra database (right bar); (c) CDF of the content lookup time of the Gateway and of the DHT for comparison; (d) Cache hit rate of the Gateway throughout the day, aggregated in 1-hour intervals.

hash space so that every peer has a Hydra Head in its 20-proximity; and (*ii*) the storage of provider records in the shared database, allowing many records to be retrieved within one hop. To confirm this, we examine the closest neighbors of each peer by taking a DHT routing table snapshot [40]. Confirming our expectation, we find a total of 16,208 peers in the network, with 96.6% having at least one Hydra Head in their 20-proximity. To quantify how this impacts path length, Figure 5b shows the hop count of DHT walks with and without the Hydra database. We separate the data into each AWS region. With the hydras, many DHT walks reach provider records within one hop. This is the case for 41.45% in `af-south-1`, 27.81% in `ap-southeast-2`, 26.68% in `eu-central-1`, 29.79% in `me-south-1`, 17.04% in `sa-east-1`, and 46.64% in `us-west-1`. In contrast, without the Hydra database, under 5% in `af-south-1` and 2% in other regions of queries achieve single hop routing, and about 20% of DHT walks require three or more hops to access the provider record in all regions.

Gateway Content Lookup Time. We finally inspect the gateway, which also improves retrieval performance by introducing centralized caching, circumventing the need to traverse the DHT or Indexers. Figure 5c displays a comparison between the gateway content lookup time for IPFS node-stored content and non-cached content. As a baseline, it also shows the DHT content lookup time when retrieving from an IPFS node, as obtained from our Hydra Booster data (§4.2). Note that Nginx cached content is not plotted separately because the latency is too short (near 0) to be plotted for comparison.

We see that introducing this centralized caching brings significant retrieval performance benefits. At the 50th percentile, the gateway achieves approximately 100x better performance than the DHT. This is because most requests hit either the Nginx cache or the node cache, as plotted in Figure 5d. On average, the Nginx cache hit rate is 50.3% and the average IPFS node storage hit rate is 27.6%. The gateway lookup time for cached objects is just 24 ms at the 95th percentile.

That said, there is little improvement for the remaining objects that are not cached. The non-cached content shows a similar trend to DHT lookup in Figure 5c. This is not surprising since, for non-cached objects, the gateway's co-located IPFS node also needs to perform a DHT lookup. On top of this, the co-located IPFS node needs to retrieve the content and then forward it to the retrieval client, thus introducing extra overhead (which causes the disparity between the non-cached line and the DHT line). This highlights the complexity of inter-linking these centralized components with the larger (slower) decentralized system.

Discussion & Takeaways. The centralized Indexer, Hydra Boosters, and gateways improve content retrieval performance from different perspectives. The Indexer offers better content lookup time in 90% of cases, while the gateway aggregates user demand to provide better content retrieval via caching, in 95% of cases compared to DHT lookup time. The Hydra Boosters reduce the number of hops during DHT lookups to achieve faster lookup times. Overall, deploying these components successfully improves content retrieval performance compared to our earlier fully decentralized design. That said, the introduction of these components leads to clear trade-offs, most notably between performance and privacy. As centralized instances, they allow operators to monitor and even censor usage. However, our experience and previous research [10, 37] has shown that these attacks are still possible in DHTs, regardless of whether Indexers or gateways are present. We further discuss this in §6.

5.3 Challenge 3: Adoption

Client Adoption. The gateways are our main mechanism to improve adoption, by providing direct integration with HTTP. Thus, we first inspect its success by measuring the number of gateway users as a metric of adoption. For this, we categorize all user-agents into three categories: desktop browsers, mobile browsers, and other net tools [9] or libraries such as *wget*. Access from both desktop browsers and mobile clients is a crucial metric for evaluating improvements in adoption because IPFS node is not yet implemented on mobile devices, and setting up an IPFS on a desktop can be complicated.

Figure 6a depicts the number of requests, number of users (we treat each IP address as an individual user), and total traf-

fic served for different user-agents. Figure 6b also presents a breakdown of the file types requested. Confirming uptake, 771,977 users access the gateway from mobile clients, 776,286 from desktop user-agents, and 195,854 using net tools. 31.8% of the total traffic and 18.8% of the total requests come from mobile devices, whereas 32.9% of the total traffic and 29.8% of the total requests originate from desktop browsers. This suggests that our gateway has successfully expanded access to IPFS.

For completeness, Figure 6b presents a breakdown of the file types requested. Note, `stream` refers to media streaming. We observe a wide range of object types. Excluding the unknown objects, for desktop agents, the most popular type of content is `stream` (20.47%) and `json` (12.99%). For mobile agents, the most accessed is `other` content (26.42%) followed by `web` content (14.57%). This confirms that the gateway has been adopted for various applications and content types.

Web Developer Adoption. To assess the adoption of the gateway by traditional websites, we look into the HTTP referer. This tells us from which website a request has been generated. We measure the number of requests with and without a referral, as well as the number of users and the total traffic served. The results show that 51.6% of the total traffic and 29.8% of the total requests come from a referral website, suggesting significant integration by web developers. This is similar to the percentage of traffic (48.3%) without a referral and lower than the requests (70.1%) without a referral. The requests with a referer come from 1,312,066 users, while the requests without a referer come from 405,087 users, which is more than 3x fewer, confirming that the majority of users use gateways due to integration with websites. This suggests that the gateway *has* supported web developer adoption, with the placement of IPFS-hosted objects in their pages.

We finally check the top 100 referral sites (based on the number of requests received) and manually label them with their website category. Figure 6c displays the request volumes among different referral types, the number of referral sites, total traffic served, and user count. Note, the IPFS type consists of IPFS official sites and tool sites such as the Gateway Checker [6]. We see that NFT and online video are the most common types of referral for the gateway, while other referral types only account for a small fraction. The former is likely because a large amount of NFT content is stored on IPFS, whereas the latter may simply be an effort to offload traffic from a developer's own server. Figure 6d also displays the top 100 referral sites, ranked by the number of requests and the number of users respectively. We observe a long-tail distribution, with 70.23% of the requests originating from just 23 websites. The findings arguably point to a level of additional centralization, with the majority of referred requests coming from a small number of top websites.

Discussion & Takeaways. The gateways enhance the adoption of IPFS for both clients and web developers, with a significant number of users accessing IPFS through our gateway on their mobile devices (which would not be possible without it). Additionally, over half of the traffic to our gateway is generated by referrals from existing HTML-based sites. The introduction of this centralization, therefore, has clear benefits. However, the use of such HTTP gateways does sacrifice the end-to-end cryptographic validation of IPFS content, which may enable man-in-the-middle attacks. Moreover, as centralized instances, gateways enable their operators to monitor usage, raising privacy concerns. We further discuss these concerns in §6. We also note that gateways may create other risks. For example, ideally, nodes who retrieve content also volunteer to serve it for others. This enables self-scaling, naturally creating more replicas of popular content. However, the use of the gateways prevents this and may risk an over-reliance on centralized elements. Thus, even though one can always circumvent any gateway, a lack of decentralized participants could hamper this in the longer term.

6 Discussion

The above solutions were introduced to mitigate challenges related to content publication, retrieval, and system adoption. This creates a trade-off, potentially undermining certain benefits of decentralization. We explore this below.

6.1 Security

Hydra Booster Security. One concern regarding the Hydra Boosters is that a centralized entity controlling multiple heads might theoretically gain excessive influence over IPFS routing, leading to issues such as result poisoning and Eclipse attacks [36]. However, we posit that these security concerns are unlikely to manifest. First, it is difficult for false content to be returned, as IPFS relies on self-validating CIDs. Additionally, while Hydra Boosters can supply the provider record, they do not serve the content itself. Thus, the most severe action Hydra Boosters could engage in is presenting a false provider record, forcing another query. This means that such an attack would only slow retrieval, rather than prevent it.

Second, as outlined in §3.2, the primary objective of the Hydra Boosters is to achieve a uniform distribution of Hydra heads. Our measurement (See Appendix B for details) shows that the mean number of hydra heads within a peer's 20-proximity is 2.36, and 99.5% peers have no more than five peers in the 20-proximity. Given the presence of just a few Hydra heads within 20-proximity, the ability to manipulate routing or execute eclipse attacks is constrained. This is because peers connect to not just the Hydra heads, but also and primarily to other peers. Thus, as long as the ordinary nodes in the IPFS network continue to function as intended, the routing of IPFS should remain resilient against potential disruption from Hydra Boosters. That said, the disruption would inevitably lead to a performance decline. The performance

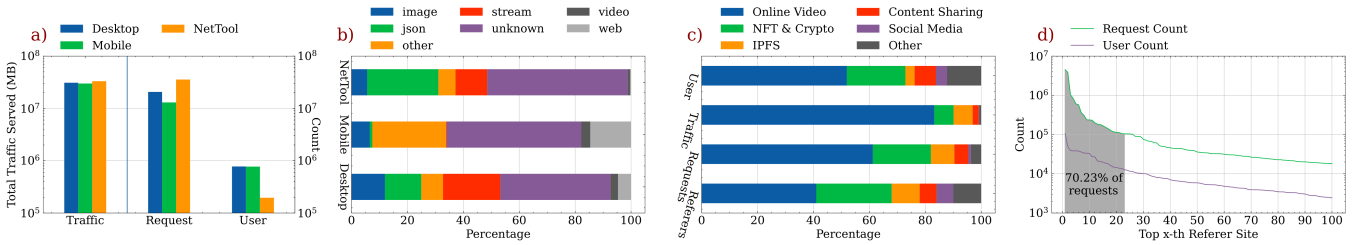

Figure 6: (a) Number of requests, number of unique users, and total traffic served for different user-agents; (b) Breakdown of requested file types for different user-agents; (c) Breakdown of referers for number of referral sites, the number of requests, total traffic served, and user count of the different referral types across the top 100 sites ranked by requests count; (d) The requests count and user count of the top 100 referer sites, ranked by request count and by user count, respectively.

impact, in this case, is similar to the situation where all Hydra Boosters are inactive, a topic we discuss further in §6.2.

Gateway Security. The HTTP gateway sacrifices the end-to-end cryptographic validation of content, which raises potential man-the-middle attacks that IPFS strives to prevent. To overcome this, retrieval client browsers could compute the hash of the retrieved file to verify whether they have received the correct content. However, this process entails the use of additional functionality in the browser, which may undermine the goal of improving adoption.

6.2 Robustness

Hydra Booster Robustness. The robustness of the Hydra Boosters could impact IPFS in two ways: (*i*) failing to return provider records (*e.g.,* due to the failure of the DynamoDB instance); or (*ii*) failing to correctly participate in IPFS routing (*e.g.,* due to failures on AWS EC2). Despite this, we emphasize that the failure of either component will result in IPFS peers falling back to the traditional decentralized model. Thus, although performance will suffer, availability will not. Confirming this, §5.2 demonstrated the scenario where Hydra Boosters stop returning provider records. This situation mirrors the conditions that arise when DynamoDB experiences a shutdown. Here, we observe a performance decrease by up to 36.5%, yet the content availability is not affected.

An additional risk is that the failure of the Hyrda Boosters will negatively impact routing. To test this, we repeat the methodology in §4.2, but prevent peers from utilizing any Hydra Boosters for routing (detailed in Appendix B). We do not observe any failed retrieval, indicating that the availability of provider records remains unaffected even when the peers do not interact with Hydra Boosters.

Indexer Robustness. When using the Indexers, the availability of provider records is entirely dependent on the availability of the Indexers. Unlike the DHT, which replicates provider records across 20 peers, centralized failures in the Indexer can make significant volumes of provider records inaccessible. This is mitigated by the ability of Indexers to gossip announcement messages among themselves. However, gossipsub (see §3.1) offers no hard guarantees on real-time synchronization. There are already 7 Indexers available for public use, and Protocol Labs is working on a protocol for automatic discovery of alternative Indexers. As more stakeholders operate Indexers, we hope the risk of Indexer unavailability decreases.

Briefly, it should be noted that the failure of an indexer does not necessarily result in a permanent loss of provider records. As explained in §3.1, Indexers obtain provider records from the advertisement chain of content providers. Therefore, provider records are inherently replicated by providers' advertisement chains. If an Indexer loses any provider records, it can restore them by re-synchronizing with the original providers, assuming such providers still wish to make their content available.

6.3 Privacy

Another concern is that, as centralized instances, these components may enable their operators to monitor usage. For instance, the Indexers and gateways can monitor the queried CIDs from the clients. Similarly, the Hydra Boosters can observe almost every DHT query in IPFS because most peers are directly connected to a Hydra Head. This asymmetry of power is a key concern and makes it necessary for clients to trust the operators with their data. We conjecture that this is one of the reasons why most IPFS clients use an instance of these components operated by Protocol Labs. Despite this, we acknowledge this is not a sustainable solution, with the risk of Protocol Labs becoming an overly central dependency. Although not discussed here, we also briefly note that the Indexers use a double-hashing technique to make this monitoring more difficult [5].

Our experiences within IPFS have also flagged that many of these attacks are possible in the decentralized DHT routing subsystem, agnostic to the presence of Indexers or gateways. For instance, a well-placed node in the DHT can learn which files are requested by a client during the routing process. By performing a Sybil attack, a well-resourced adversary could even gain global vantage on the routing process. Recent works

also show that privacy attacks [10] and censorship [37] are possible in the purely decentralized IPFS DHT. Despite that, it is clear that greater decentralization reduces the vantage of individual nodes, raising the barrier to such attacks.

6.4 Incentives

In previous subsections, we mention that issues with robustness can be mitigated by multiple stakeholders running these centralized components. This, however, requires a clear incentive model to encourage third-party operators to contribute their resources. To date, there are 7 Indexers and over 80 Gateways, and the stakeholders running them typically have three types of incentives: (*i*) Stakeholders that provide storage services utilizing IPFS can benefit from running Indexers and/or Gateways. These ease the discovery of their stored content, broaden adoption, and improve their service's performance. Such stakeholders include NFT.storage, Web3.storage, DSS, SXX, FilSwan, PikNik, Ken Labs, etc. (*ii*) Stakeholders that provide other applications built on top of IPFS benefit from running an Indexer and/or a Gateway to improve the stability and performance of their service. Such stakeholders include LeewayHertz, 4everland, Aragon, Pinata, etc. (*iii*) Stakeholders that have a broader interest in decentralized web applications have a motivation to contribute to the environment as it aligns with their long-term interests. Such stakeholders include CloudFlare and Infura.

We acknowledge that none of the aforementioned reasons provide formal incentives by design. It is yet to be determined how wide uptake is among third-party stakeholders without a formal incentive model in place. Our future plans involve implementing incentive mechanisms, and there are several potential options available [24].

7 Related Work

Decentralized Web. The IPFS network has grown alongside other Decentralized Web technologies, particularly the "fediverse". This is composed of server-based federated services, such as Mastodon [32], Pleroma [23], and Diaspora [22]. The service closest to IPFS is Nextcloud, which offers a federated file storage platform that integrates IPFS with server-local storage [1]. This functions similarly to IPFS gateways. However, IPFS can operate without gateways, while fediverse apps rely entirely on the uptime of federated servers [32].

P2P Networks. There have been numerous P2P overlay architectures, with dozens of DHT structures proposed, including Chord [38], Tapestry [44], Koorde [26], Pastry [34], and others [25]. Various applications have been built atop, such as large-scale content delivery platforms [14] and decentralized social networks [21], among others. Rather than creating an entirely new system, IPFS uses the Kademlia DHT for content indexing [30]. IPFS is built upon these technologies and is currently one of the largest "Decentralized Web" technologies deployed in the world. BitTorrent, which also uses Kademlia [14], is another example of a large-scale deployment. Note, as part of our decentralization efforts, IPFS aims to be resistant to censorship, much like other platforms such as Freenet [12] and Wuala [28]. These platforms achieve their goal by storing encrypted content across a random subset of peers. In contrast, IPFS follows a BitTorrent-like approach where nodes store only the content they are interested in.

DHT Optimization. Various attempts have been made to enhance the performance and usability of DHTs by employing caching [35], network-aware peer selection [27], and parallelizing lookups [39]. We also take inspiration from prior Information-Centric Networking designs that use DHTs for content indexing [17, 18]. We build on these prior works with our deployment of hydra boosters. These improve performance while offloading traffic from the DHT.

P2P System Evaluations. Closest to our evaluation are studies that measure operational IPFS [41]. Moreover, in [20], the authors measure DHT lookup latencies in the range of tens of seconds, but it was not a controlled experiment. There have been various performance evaluations of P2P systems. For example, [16] and [43] evaluate BitTorrent's implementation of Kademlia, to find a significant number of failed nodes that adversely affected the lookup times. Further, Stutzbach and Rejaie [39] model Kademlia's performance and propose several improvements. We borrow from these measurement studies to inform our methodologies in §4.

8 Conclusion

This paper has presented our experiences of deploying three centralized components within IPFS, namely Indexers, Hydra Boosters, and Gateways. Using real-world operational data, we have demonstrated their effectiveness in addressing the challenges faced by IPFS: massive content publication, content retrieval performance, and adoption. We have also discussed trade-offs, related to security, privacy, and other risks caused by the compromises of prior decentralization. Overall, our findings highlight the persistent challenges that exist in deploying operational decentralized systems with tight performance constraints. Equally, we have highlighted the challenges of overcoming traditional issues with centralized deployments (*e.g.,* single points of failure and privacy). Thus, we emphasize that developers must make judgment calls about how to balance such trade-offs. In the future, we plan to explore the remaining challenges discussed in §6. We will continue working on the research and development of decentralized protocols, and we hope that IPFS can eventually perform similarly even without these centralized parts.

References

[1] IPFS for Nextcloud. https://apps.nextcloud.com/, 2020.

[2] Ipfs ecosystem directory. https://ecosystem.ipfs.io/, 2022.

[3] Bitswap. https://docs.ipfs.tech/concepts/bitswap/, 2023.

[4] Internet traffic from mobile devices (apr 2023). https://explodingtopics.com/blog/mobile-internet-traffic, 2023.

[5] The ipfs dht reader privacy upgrade. https://discuss.ipfs.tech/t/the-ipfs-dht-reader-privacy-upgrade/16279, 2023.

[6] Ipfs gateway checker. https://ipfs.github.io/public-gateway-checker/, 2023.

[7] Multihash. https://github.com/multiformats/multihash, 2023.

[8] Pebble database. https://github.com/cockroachdb/pebble, 2023.

[9] User-agent net tool ua strings. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers, 2023.

[10] BALDUF, L., HENNINGSEN, S. A., FLORIAN, M., RUST, S., AND SCHEUERMANN, B. Monitoring data requests in decentralized data storage systems: A case study of IPFS. *CoRR abs/2104.09202* (2021).

[11] BOMMELAER DE LEUSSE, C., AND GAHNBERG, C. The global internet report: Consolidation in the internet economy. *Internet Society* (2019).

[12] CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T. W. Freenet: A distributed anonymous information storage and retrieval system. In *Designing privacy enhancing technologies* (2001), Springer.

[13] CLAY, K. Amazon.com Goes Down, Loses $66,240 Per Minute, 2013.

[14] COHEN, B. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems* (2003), vol. 6, Berkeley, CA, USA, pp. 68–72.

[15] COLDEWEY, D. Cloudflare dns goes down, taking a large piece of the internet with it, 7 2020.

[16] CROSBY, S. A., AND WALLACH, D. S. An analysis of bittorrent's two kademlia-based dhts. Tech. rep., Rice Technical Report, 2007.

[17] DANNEWITZ, C., D'AMBROSIO, M., AND VERCELLONE, V. Hierarchical dht-based name resolution for information-centric networks. *Computer Communications 36*, 7 (2013), 736–749.

[18] DANNEWITZ, C., KUTSCHER, D., OHLMAN, B., FARRELL, S., AHLGREN, B., AND KARL, H. Network of information (netinf)–an information-centric networking architecture. *Computer Communications 36*, 7 (2013), 721–735.

[19] DOAN, T. V., VAN RIJSWIJK-DEIJ, R., HOHLFELD, O., AND BAJPAI, V. An empirical view on consolidation of the web. *ACM Transactions on Internet Technology (TOIT) 22*, 3 (2022), 1–30.

[20] FALKNER, J., PIATEK, M., JOHN, J. P., KRISHNAMURTHY, A., AND ANDERSON, T. Profiling a million user dht. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement* (New York, NY, USA, 2007), IMC '07, Association for Computing Machinery, p. 129–134.

[21] GRAFFI, K., GROSS, C., STINGL, D., HARTUNG, D., KOVACEVIC, A., AND STEINMETZ, R. LifeSocial. KOM: A secure and P2P-based solution for online social networks. In *CCNC* (2011).

[22] GUIDI, B., CONTI, M., PASSARELLA, A., AND RICCI, L. Managing social contents in Decentralized Online Social Networks: A survey. *Online Social Networks and Media 7* (2018).

[23] HASSAN, A. I., RAMAN, A., CASTRO, I., ZIA, H. B., DE CRISTOFARO, E., SASTRY, N., AND TYSON, G. 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.

[24] IHLE, C., TRAUTWEIN, D., SCHUBOTZ, M., MEUSCHKE, N., AND GIPP, B. Incentive mechanisms in peer-to-peer networks – a systematic literature review. *ACM Computing Surveys* (Jan. 2023).

[25] ISDAL, T., PIATEK, M., KRISHNAMURTHY, A., AND ANDERSON, T. Privacy-preserving p2p data sharing with oneswarm. In *Proceedings of the ACM SIGCOMM 2010 Conference* (New York, NY, USA, 2010), SIGCOMM '10, Association for Computing Machinery, p. 111–122.

[26] KAASHOEK, M. F., AND KARGER, D. R. Koorde: A simple degree-optimal distributed hash table. In *International Workshop on Peer-to-Peer Systems* (2003), Springer, pp. 98–107.

[27] KAUNE, S., PUSSEP, K., LENG, C., KOVACEVIC, A., TYSON, G., AND STEINMETZ, R. Modelling the internet delay space based on geographical locations. In *2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing* (2009), IEEE, pp. 301–310.

[28] MAGER, T., BIERSACK, E., AND MICHIARDI, P. A measurement study of the wuala on-line storage service. In *2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P)* (2012), IEEE.

[29] MATHEWS, E. Amazon cloud outage hits major websites, streaming apps, 12 2021.

[30] MAYMOUNKOV, P., AND MAZIERES, D. Kademlia: A peer-to-peer information system based on the xor metric. In *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers*. Springer, 2002, pp. 53–65.

[31] MITZENMACHER, M. The power of two choices in randomized load balancing. *IEEE Trans. Parallel Distrib. Syst. 12*, 10 (oct 2001), 1094–1104.

[32] RAMAN, A., JOGLEKAR, S., CRISTOFARO, E. D., SASTRY, N., AND TYSON, G. Challenges in the decentralised web: The mastodon case. In *Proceedings of the Internet Measurement Conference* (2019), pp. 217–229.

[33] ROSEMAIN, M., AND SATTER, R. Millions of websites offline after fire at french cloud services firm, 3 2021.

[34] ROWSTRON, A., AND DRUSCHEL, P. 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* (2001), Springer, pp. 329–350.

[35] SALEH, O., AND HEFEEDA, M. Modeling and caching of peer-to-peer traffic. In *Proceedings of the 2006 IEEE International Conference on Network Protocols* (Nov. 2006), IEEE.

[36] SINGH, A., CASTRO, M., DRUSCHEL, P., AND ROWSTRON, A. Defending against eclipse attacks on overlay networks. p. 21.

[37] SRIDHAR, S., ASCIGIL, O., KEIZER, N., GENON, F., PIERRE, S., PSARAS, Y., RIVIÈRE, E., AND KRÓL, M. Content censorship in the interplanetary file system. *arXiv preprint arXiv:2307.12212* (2023).

[38] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review 31*, 4 (2001), 149–160.

[39] STUTZBACH, D., AND REJAIE, R. Improving lookup performance over a widely-deployed dht. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications* (2006), IEEE, pp. 1–12.

[40] TRAUTWEIN, D. Nebula – A crawler for networks based on the libp2p DHT implementation, 7 2021.

[41] TRAUTWEIN, D., RAMAN, A., TYSON, G., CASTRO, I., SCOTT, W., SCHUBOTZ, M., GIPP, B., AND PSARAS, Y. Design and evaluation of ipfs: A storage layer for the decentralized web. SIGCOMM '22, Association for Computing Machinery, p. 739–752.

[42] VYZOVITIS, D., NAPORA, Y., MCCORMICK, D., DIAS, D., AND PSARAS, Y. Gossipsub: Attack-resilient message propagation in the filecoin and eth2.0 networks, 2020.

[43] WOLCHOK, S., AND HALDERMAN, J. A. Crawling BitTorrent DHTs for fun and profit. In *4th USENIX Workshop on Offensive Technologies (WOOT 10)* (Washington, DC, Aug. 2010), USENIX Association.

[44] ZHAO, B. Y., HUANG, L., STRIBLING, J., RHEA, S. C., JOSEPH, A. D., AND KUBIATOWICZ, J. D. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications* 22, 1 (2004), 41–53.

A Ethics

All data used within this paper is collected as part of Protocol Lab's operational activities. Although we observe IP addresses, we do not attempt to map these back to personal identities, as such analysis is not within the scope of this study. The IPFS gateway data also contains user information, as it covers requests from web clients. In all cases, we anonymize IP addresses, and do not perform lookups on the CIDs to infer the nature of the content exchanged. We perform no per-user analysis, and focus only on overall system analysis. All information is collected as part of our routine operations, and in line with IPFS policies. This paper has not triggered additional data collection.

B Additional Experiments

B.1 Hydra Heads in 20-Proximity

We examine the closest neighbors of each peer by taking a DHT routing table snapshot [40]. Figure 7 shows the CDF of the number of hydra heads in a peer's 20-proximity.

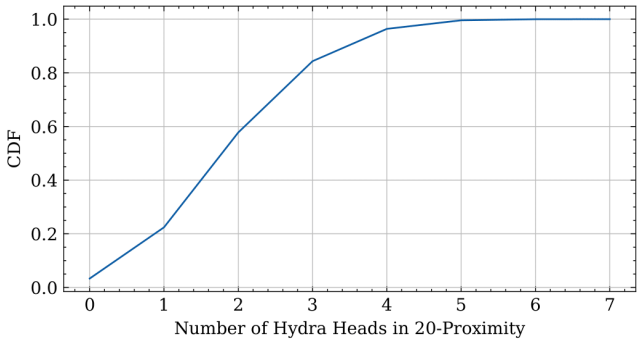

Figure 7: CDF of the number of hydra heads in a peer's 20-proximity.

B.2 Active Retrieval Ignoring Hydra Boosters

We conduct another experiment using the same methodology as described in §4.2. However, in this experiment, we utilize modified IPFS nodes that disregards all Hydra heads. This modification incurs minimal overhead since we possess a list of the PeerIDs of all Hydra heads. This simulation mirrors the conditions where a peer choose not to interact with any Hydra head. The experiment starts on `2022-11-28 16:17 UTC` to `2022-12-09 20:00 UTC`. Table 2 list the number of retrievals and publications from each region with modified IPFS nodes that ignore Hydra Boosters. Figure 8 shows the performance comparison by plotting the content lookup time across all probed AWS regions with and ignoring the Hydra database.

	Retrieval	Publication
af_south_1	7533	1674
ap_southeast_2	7800	1674
eu_central_1	8003	1677
me_south_1	7578	1674
sa_east_1	7529	1675
us_west_1	7886	1571

Table 2: The number of retrievals and publications from each region with modified IPFS nodes that ignore Hydra Boosters.

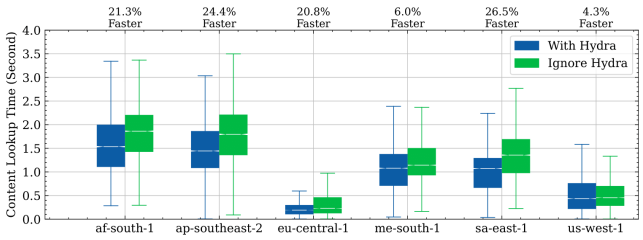

Figure 8: The content lookup time across all probed AWS regions with and ignoring the Hydra database, the values at the top refer to the medians.