Characterization of the IPFS Public Network from DHT Requests

Bastien Confais[1(✉)], Benoît Parrein[2], Jérôme Lacan[3], and François Marques[1]

[1] Inatysco, 30 rue de l'Aiguillerie, 34000 Montpellier, France
{bastien.confais,francois.marques}@inatysco.fr

[2] Nantes Université, Polytech Nantes, rue Christian Pauc, BP50609, 44306 Nantes, France
benoit.parrein@univ-nantes.fr

[3] ISAE Supaero, 10, Avenue Édouard-Belin, BP 54032, 31055 Toulouse, France
jerome.lacan@isae-supaero.fr

Abstract. Interplanetary File System (IPFS) is a file sharing network relying on a Distributed Hash Table (DHT) to locate data and a BitTorrent-like protocol to exchange blocks between the peers. However, in such a network, all nodes can access information about the files stored or accessed by others. In this article, we use these public pieces of information to try to characterize the IPFS network both in terms of nodes composing it and on the files stored in it. To that end, we set up an IPFS node connected to the public IPFS network and saved all the DHT requests forwarded through it. We show that nodes are mostly located in datacenters and not on the end-users' computers and therefore that files are often accessed through public gateways. We also show that most files are not replicated and are not accessed frequently (cold data) which can question us about the relevance of using IPFS in such use case scenarios.

Keywords: IPFS · Network monitoring · Kademlia DHT · Peer-to-Peer storage

1 Introduction

Distributed Storage solutions such as Google File System [10], often rely on a centralized metadata server and require to trust in storage nodes. Therefore, they can only be used within a single network provider or datacenter. To solve these major drawbacks, new alternatives such as Interplanetary File System (IPFS) [5], Sia [26] or Arweave [27] were developed in the last years. Some solutions are even built as an overlay of IPFS like Filecoin [6]. These solutions enable anyone, including non-trusted nodes to join the network. This is made possible thanks to replication of data and protocols checking the integrity of the returned data, preventing nodes from misbehaving. They also replace the centralized metadata server with a distributed structure such as a Distributed Hash Table (DHT) or

A. Hameurlain and A. M. Tjoa (Eds.): *Transactions on Large-Scale Data-and Knowledge-Centered Systems LV*, LNCS 14280, pp. 87–108, 2023.
https://doi.org/10.1007/978-3-662-68100-8_4

a blockchain. IPFS was developed in 2014 [5] and relies on a BitTorrent-like protocol [17] to exchange data between peers and on a Kademlia Distributed Hash Table [18] to locate the replicas.

In this paper, we propose a deep analysis of the public IPFS network, analyzing the nodes composing it and the pieces of data that are stored. We show that despite the need for the user to know the file identifier to retrieve the corresponding file, any node on the network can discover which files are stored and where, questioning the lack of privacy of using such networks. The remaining of the paper is organized as follows. Section 2, presents how IPFS works and lists some use cases of the use of IPFS that we can find in the literature. Section 3 presents the motivations of this work. Section 4 details how data was collected and Sect. 5 analyzes the results. Section 6 tries to take a step back and analyzes the consequences of the results before concluding in Sect. 7.

2 Interplanetary File System

IPFS [5] is a distributed storage solution relying on a BitTorrent-like protocol [17] to exchange data between the peers and on a Kademlia DHT [18] to locate the replicas. In the following, we explain how IPFS is working.

First, to join the network, a node connects to a bootstrap node in the DHT. The address of the bootstrap node is embedded in the source code of the software program as a default value for the node configuration. Once connected to the bootstrap node, the node sends DHT requests to build its routing table and determine which nodes it has to connect directly to. The neighboring nodes also send a replica of the DHT records to the new node, so it can be stored and served to other users. This is summarised in Fig. 1.

2.1 Writing Operation

When a user wants to write a new file, IPFS software program splits the file into different blocks of 256 KB that are stored on the local node. The block identifiers are the values of the hashes of the block contents. A Merkle tree is then built from these different blocks. The root hash becomes the file identifier (CID) and will be used to guarantee the integrity of the file (preventing the storage node to act maliciously) when retrieved. For each data block, the node creates a DHT record to indicate that it stores these blocks. This record is stored on the node in charge of managing the key of the block identifier. Therefore, the different records are spread within the IPFS network. Finally, the Merkle tree is also stored in the DHT. Figure 2 shows the full process of file creation.

2.2 Read Operation

To access a file, the user needs to know the value of the root hash of the Merkle tree corresponding to the file. A request is sent to the DHT to retrieve the Merkle tree. Then, a DHT request is sent for each block. This enables the node to get the

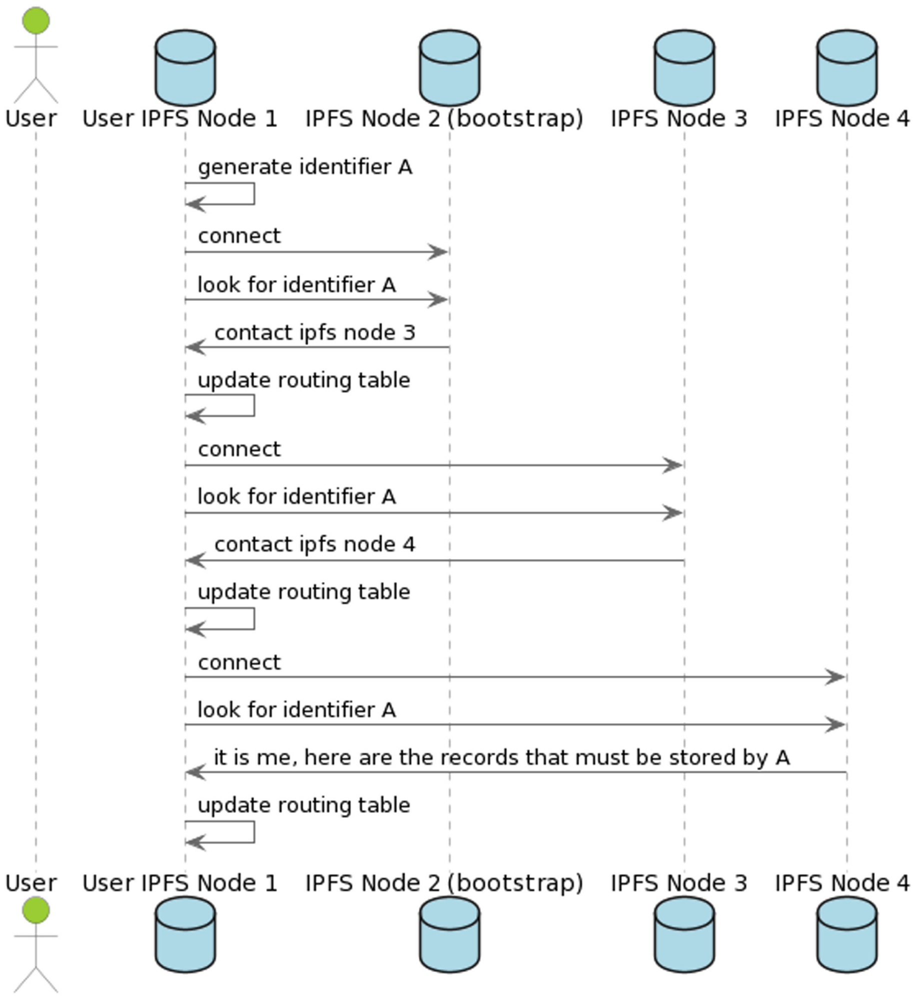

Fig. 1. Sequence diagram for the connection of a new node.

list of the available replicas for the block. Then, it can contact the nodes storing a replica to retrieve it. Finally, the copy of the retrieved blocks are stored on the local node, and the DHT is updated accordingly. This process is illustrated in Fig. 3.

Because of the DHT, many nodes are informed of the existence of the file, which is so not private. From a user's point of view, this can be counter-intuitive because the user can think that when the file identifier is not divulged, nobody except him can access the file.

2.3 Main Uses of IPFS

IPFS is currently used in different situations. A first use case is file sharing, that can benefit from the efficiency of the BitTorrent protocol and especially in video

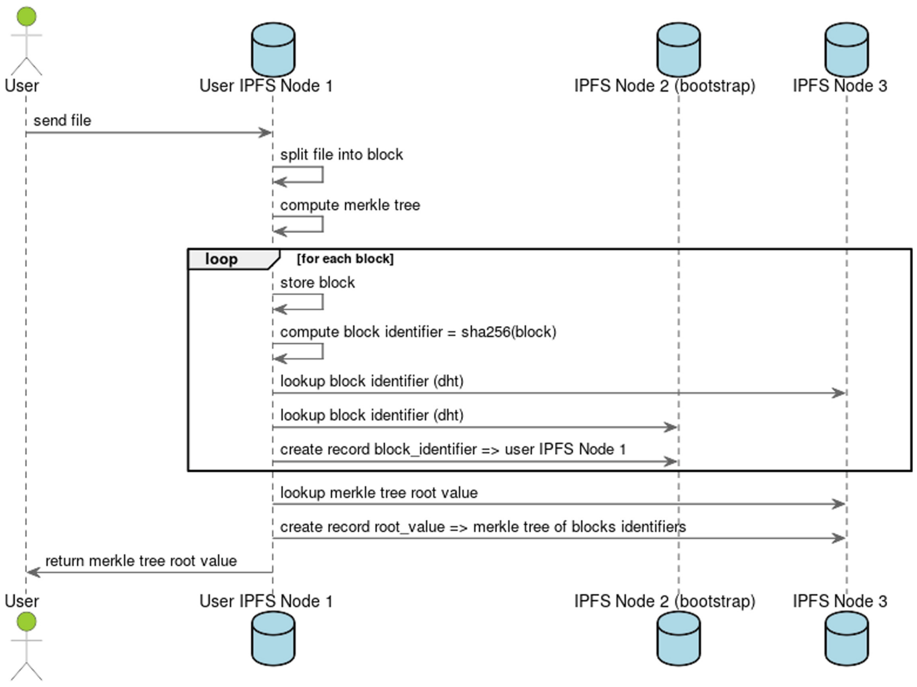
User
User IPFS Node 1
IPFS Node 2 (bootstrap)
IPFS Node 3
send file
split file into block
compute merkle tree
loop
[for each block]
store block
compute block identifier = sha256(block)
lookup block identifier (dht)
lookup block identifier (dht)
create record block_identifier => user IPFS Node 1
lookup merkle tree root value
create record root_value => merkle tree of blocks identifiers
return merkle tree root value

Fig. 2. Sequence diagram of the writing operation: the creation of the file on the storage node and the update of the DHT spreads among all nodes of the network

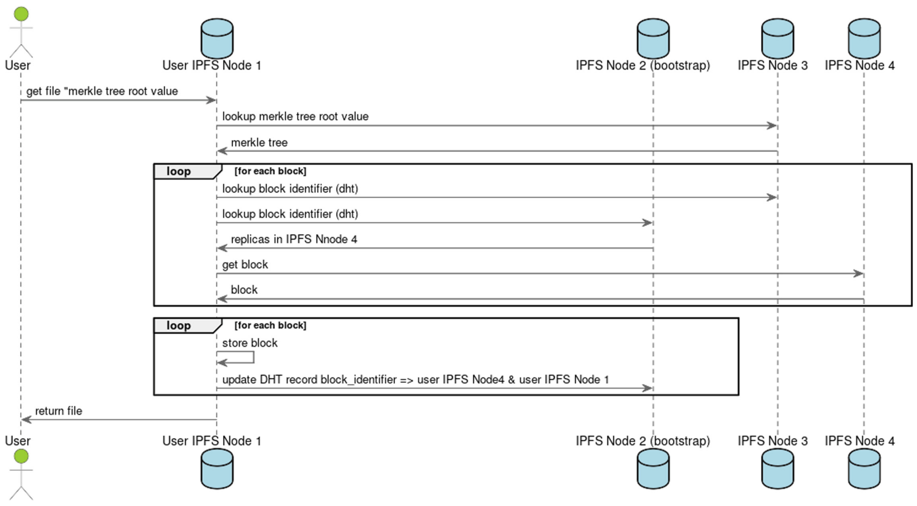
User
User IPFS Node 1
IPFS Node 2 (bootstrap)
IPFS Node 3
IPFS Node 4
get file "merkle tree root value
lookup merkle tree root value
merkle tree
loop
[for each block]
lookup block identifier (dht)
lookup block identifier (dht)
replicas in IPFS Nnode 4
get block
block
loop
[for each block]
store block
update DHT record block_identifier => user IPFS Node4 & user IPFS Node 1
return file

Fig. 3. Sequence diagram of the reading operation implying....

streaming application [9] or in the Internet of Things (IoT) to collect and store measurements [2,11]. Hosting a website [19], managing file revisions [14,20] or storing large container files are other use cases mentioned on the IPFS website[1]. However, coupled with a blockchain, IPFS can also be used for non-repudiation storage [24] providing a proof that a certain file exists and has been stored by a specific user. In practice, IPFS is used for traceability in supply chains [3] or for pieces of data produced by the devices of the IoT [16].

Monitoring IPFS software program has been done in many works. Shen *et al.* [22] or Abdullah *et al.* [1] evaluated the I/O performance of the program, but they deployed their own IPFS network and have the control over all nodes composing it. Public IPFS network has also been studied. For instance, it has been done in 2021 by Balduf *et al.* [4], Daniel *et al.* [8], Henningsen *et al.* [12] or even more recently by the organisation that develops the IPFS software program in Trautwein *et al.* [25]. We will present at the end of Sect. 4 the difference between their methodology and ours.

3 Motivations and Objectives

Before launching this study, we were interested in developing a software program that uses IPFS as a back-end for file exchanges between users. However, we quickly realised that the DHT could leak privacy information about the partners that are exchanging files but also that everybody could access the content of the file if it was not encrypted. The first goal of the paper is to demonstrate by experimentation that IPFS does not preserve the privacy of users. For that purpose, we would like to know if we can identify some peers that work together, that access the same files, indicating the movement of the user or two users that collaborate on a regular basis. The second goal of the paper is to gather as much information as possible about the nodes that are present in the network: what nodes are composing the network, when are they exchanging, but also on the files that are exchanged: their types, the number of replicas in order to characterize the network. The idea behind this is to try to determine the standard usage of IPFS, if people use it for its BitTorrent properties, or a storage like an alternative to Google Drive. We also want to make stable observations that are valid and reproducible. Therefore, we made the data collection from different nodes hosted on different providers.

4 Material and Methods

This study was made on the IPFS public networks that anybody can join and where all files are publicly available. We connected a IPFS node to the Internet for an entire year, from mid-December 2021 to January 2023 and saved the DHT requests that were forwarded by our node. The monitoring node was deployed on

[1] https://docs.ipfs.tech/concepts/usage-ideas-examples/.

the Google Kubernetes platform[2] [7] that could be reached using a public IPv4 address (Internet Protocol address) and by installing a full IPFS node (using the implementation developed in Golang[3], version 0.9.1). The node was deployed in region “europe-west3” located in Frankfurt, Germany. To validate the results and be assured that they are reproducible, we set up a second node on Amazon AWS. This node was located in Europe, Ireland.

We simply collected the DHT requests forwarded by our node by enabling the verbosity of the IPFS process.

Each saved record contains the date when the request was made, the file identifier (Content Identifier also known as “CID”) and the identifier of the node which originally sent it (“PeerID”). In other words, a user using the node with the identifier “PeerID” is trying to locate the replicas of the file with the identifier “CID”. We add that the DHT requests that were not resolved have been ignored. We used a Python script to extract more information from these records. For each line of log, we sent our own DHT request to the file identifier in order to retrieve all the peer identifiers that stores a replica. Then, we downloaded one of the replicas in order to identify the file type (MIME type). However, in order to preserve the privacy of users, we only stopped the download after the first few bytes (1000 bytes). For most of the file types can be determined with only the 24 first bytes. Nevertheless, some file types like Microsoft Office document require at least 512 bytes to be correctly identified[4]. Finally, for each peer identifier that stores a replica of the file, we resolved the identifier using the DHT in order to determine the corresponding IP address. With this piece of information, we could determine the network operator of the node and its geographical region thanks to the public databases published by the different Local Internet Registry (LIR) and aggregated in search trees by specific companies like MaxMind[5]. This whole process (capture of DHT requests and analysis of observed records) is summarised in Fig. 4. The source code of the developed scripts can be found at the following URL: https://github.com/Inatysco/IPFS-network-analysis. We also made our dataset available alongside with the source code but, we replaced all files identifiers, node identifiers and IP addresses with random strings.

4.1 Difference with Previous Studies

The global approach of exploiting the requests that we received from the DHT is not original and is similar to Balduf *et al.* [4], Henningsen *et al.* [12], Daniel *et al.* [8] Trautwein *et al.* [25]. For instance, Balduf *et al.* [4] showed that by intercepting the DHT requests, it is possible to determine the size of the network (the number of connected nodes), the activity levels, the structure and the content popularity distribution. Daniel *et al.* [8] made a similar work in 2022 by collecting passively the DHT requests to determine the network size and churn.

[2] https://cloud.google.com/.
[3] https://github.com/ipfs/kubo.
[4] https://www.garykessler.net/library/file_sigs.html.
[5] https://dev.maxmind.com/geoip/geolite2-free-geolocation-data?lang=en.

Henningsen *et al.* [12] also proposed to exploit the Kademlia DHT used by IPFS to crawl the network. The major difference with our work is that they use a software program called "Hydra" which is a DHT indexer that manages multiple "PeerID" in order to provide them a more global vision of what the DHT contains. Finally, Trautwein *et al.* [25] published an article in collaboration with the organisation that develops the IPFS software to analyze the distribution of requests made in the network. This last study has perhaps the approach which is the most similar to ours, by analysing nodes using their IP addresses and determining the number of requests received on a daily basis. However, all these studies focused on the nodes and none of them considered the files that were exchanged. The first originality of our approach is to send extra requests to complete the pieces of information that are collected directly from the DHT logs. For instance, we send the same request ourselves in order to receive the response and determine the place of all the replicas of the requested file. This led us to analyze the number of replicas or the position of the different replicas of a file in the network. Secondly, our study was made on a longer period of time (1 year) while Balduf *et al.* [4] was made on only 2 months and Daniel *et al.* [8] presents their results on only 10 days and Trautwein *et al.* [25] have a 7-months long analysis. This leads to detect much more nodes (three times more actually) in the public IPFS network than in these previous studies as we will see in the next section of results. Finally, our approach did not require using multiple "PeerID" and different positions in the DHT like in Henningsen *et al.* [12]. The long period of capture enabled us to see the environment changes and gave more opportunities for every node to contact ours. And secondly, the extra requests sent to complete the analysis were sent uniformly to the different peers, so that we could have contacts to the different zones of the DHT.

4.2 Ethical Question on Data Collection

Collecting such metrics is not uncommon for an analysis of a public network. It was already done to test peer-to-peer networks like BitTorrent [28] or Bitcoin [15].

Ethical concerns of such data collection from public networks has been studied by Small *et al.* [23] and by Partridge *et al.* [21]. Both studies show that the risk to identify the real person behind a node is small and that the possibility to exploit the collected pieces of information to cause harm to users is limited. They also distinguish active from passive data collection. When data is obtained passively and that by using the public network, users consent to see their pieces of information accessed by other nodes and they are free to leave the network. Generally speaking, the explicit consent of users of large-scale public networks as public IP network or P2P network is very difficult to obtain.

Our approach is semi-passive as we collect passively requests from the DHT but, we also send request to determine the type of stored files and the location of the replicas. However, these pieces of information are insufficient to identify the users (it would require us to know that a specific user stores a certain set of files).

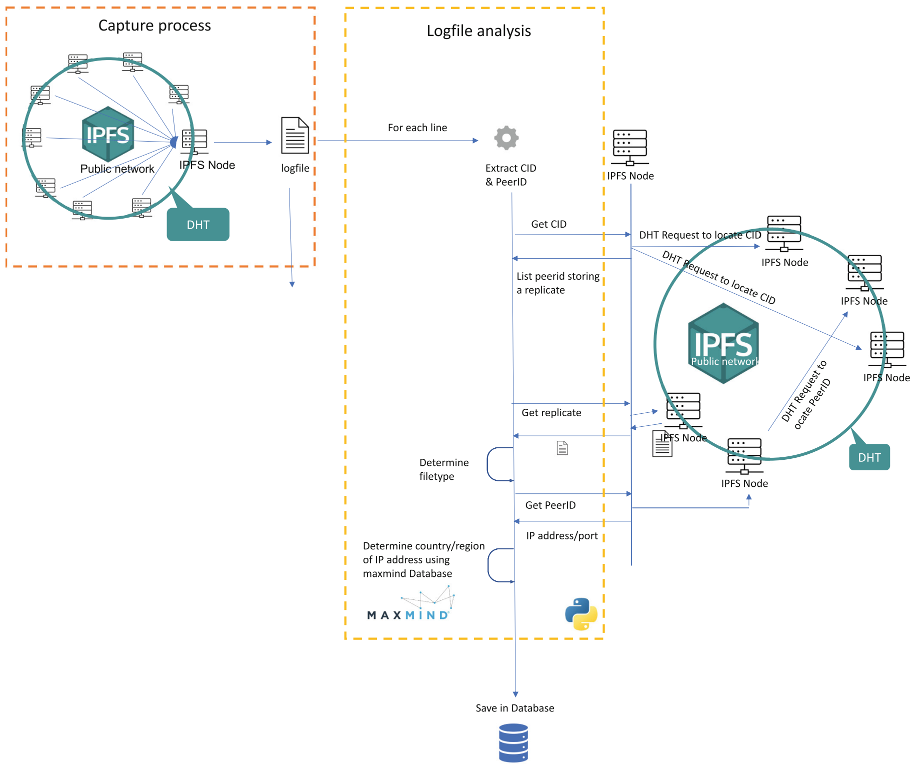

Fig. 4. Analysis process of the captured DHT requests.

If we compare our approach to previous papers, Badulf *et al.* [4] published a privacy policy on the website of their organisation to inform the users about the data collection. But there is no information about how IPFS users are concretely informed about their policy.

Trautwein *et al.* [25] added a paragraph in their paper to indicate that they have not tried to identify the real users behind the collected IP addresses. All the other works on IPFS mentioned in the previous section have not mentioned any information on ethics as recommended in [21].

5 Results

We collected 9 278 089 DHT requests that have been forwarded by our node (63 548 requests per day on average). In each request, we can see the identifier of the node sending it ("PeerID") and the "CID" (Content Identifier) of the file requested.

We also found that the node rebooted several times during the period, leading to observe few minutes without collecting any requests, but it also unexpectedly enabled it to change its position on the DHT because at each reboot, the node identifier of the node was modified.

The first general observation is that 13,5% of file exchanges are requested by only 10 nodes. These nodes probably correspond to public gateways, allowing any user to access files stored in IPFS, even if the user has not installed IPFS in his computer and does not actively participate in the network.

By observing the IP addresses associated to the node identifiers, we discovered in a second general observation that the network is composed of more than 147 000 nodes, that are mostly connected through providers operating in France, United States and Germany (detailed in Fig. 5). As a comparison, Balduf *et al.* [4] detect 48 000 different nodes in the period of their study. In the following, we will focus on the types of files that are stored in the network and their distribution among the nodes.

5.1 Type of the Requested Files

We collected approximately 230 000 file identifiers (CID) for which we could reach one of the replicas. After having downloaded the first 1000 bytes of each file, we could access to 7 074 files in plain text (TXT, JSON, ...), 5 199 PNG images and 3 953 PDF documents. All the types of file are summarised in Table 1.

Table 1. Types of files observed in the IPFS network (top 20)

MIME type	Number of occurrences	percentage
application/json	180 997	76%
application/octet-stream	35 012	14%
text/plain	7 074	2.9%
image/png	5 199	2.2%
application/pdf	3 953	1.7%
text/html	1 614	0.68%
image/jpeg	1 396	0.59%
application/gzip	812	0.34%
application/x-dosexec	366	0.15%
image/svg+xml	274	0.11%
image/gif	231	0.097%
video/mp4	227	0.095%
application/zip	162	0.06%
image/webp	116	0.049%
application/zlib	94	0.039%
text/x-java	78	0.033%
application/wasm	50	0.021%
audio/midi	39	0.016%
application/x-xz	39	0.016%
text/xml	39	0.016%

Three quarters of requests (76% exactly) are about JSON files. These mostly correspond to indexes: IPFS lets users send an entire hierarchy of files. All the identifiers of the files that are parts of the directory are added to a JSON structure and the JSON document is then stored in IPFS with its own file identifier. We have not specifically analysed the content of the directories (number of files, depth of the hierarchy, ...) but if a user accesses a file within a directory, a new DHT request is sent that could be captured and analyzed separately if our node receives the request for this resolution. In other words, when a user accesses a file like "QmXXXX/file.txt", a first request is sent to the DHT to resolve the CID of the directory: "QmXXXX". This returns a JSON object containing the filenames present in the directory, as well as the corresponding CID of the files. Then a second request is sent to the CID of the file to locate the replicas of it. In the worst case, if every file is accessed through a directory, we should see the directories representing 50% of all accesses. Because we see 76%, it means that either the position of our node led us to not capture the request for more than 2/3 of files access, or some users load the content of a directory without accessing any file. A third explanation and perhaps the most probable is that some files are under a deep hierarchy of folder (folderA/folderB/folderC/file.txt) leading to locate several JSON files before accessing the real file.

The 14% of total files that are binary files (application/octet-stream) correspond to files without recognisable header. It can be portions of a bigger file that have been split into several parts in order to enable users to perform random accesses, or it can be files that have been encrypted because the owner does not want anybody to read it. But most probably, these are tied to the fact that we only downloaded the first bytes of the files to preserve the privacy of users and we did not get sufficient information to determine the real type of the file. Therefore, it is difficult for us to understand what these files are about.

Although PDF or ZIP files can have an encrypted payload, most files with an identified type can be read by everyone. This illustrates and justifies the need of managing access permissions in such a network. Regarding the PDF files, we downloaded some of them and found that the immutability property of IPFS led some people to use it to store administrative documents such as proofs of delivery, invoices, etc. These private documents are made publicly available but probably without the consent of their owner. Medias like images (PNG, JPEG, WEBP) or videos (MP4) are stored on IPFS probably for optimising the distribution to several receivers[6]. In this case, the BitTorrent aspect of IPFS is probably what was looking for. We also note that many websites seem hosted using IPFS because we observe a lot of HTML files, but also Web assembly (WASM). These websites are usually reached through an IPFS gateway that can either be private or publicly accessible, like "ipfs.io".

The conclusion of this, is that most IPFS files are accessed through folders and that binary files from which we cannot determine the content, is the most common file type on the network. Then, we have a long tail with different kinds of files corresponding to different usages (PDF, images or websites).

[6] https://ipfs.video/.

5.2 Geolocalisation of the Nodes

We identified 147 003 node identifiers in the network but all the nodes have not the same importance. For instance, we identified 3 nodes announcing more than 2 000 IP addresses (mostly IPv6 addresses) that belong to a unique 64-bits long prefix network for each host. These nodes probably reconnect very often and announce a different address/port each time they reconnect.

In the same way, some IP addresses host different IPFS nodes, with different "peerID"s. This can correspond to nodes connected through NAT devices or IPFS nodes that reinstall their configuration on a regular basis. It can even be several IPFS daemons running on the same computer. A reason for this last case could be to improve the performance of IPFS as the daemon is only able to deal with one request at a time. Approximately 5 000 IP addresses host more than a single IPFS node.

From the node IP address, we determine a country to each node. As said in Sect. 4, countries are deduced from the Local Internet Registry (LIR) databases which can only reflect the administrative country of the network operator and not the real location of IPFS node. We luckily did not find any node with different IP addresses associated with different countries, therefore, if a node has several IP addresses, it is counted only once. However, if a node is restarted and has its identifier ("peerID") modified, it is counted as a new node.

Figure 5 shows the number of IPFS nodes that have been identified in each country. The figure shows that most IPFS nodes are located in the biggest economic zones of the world, which is not surprising. In Europe, most nodes are located in France (and Germany) and on the American continent, most nodes are located in the United States of America. The situation is different in Asia where nodes are more evenly distributed among the countries. But this can come from the fact that our node was located in Europe, prioritized connections to

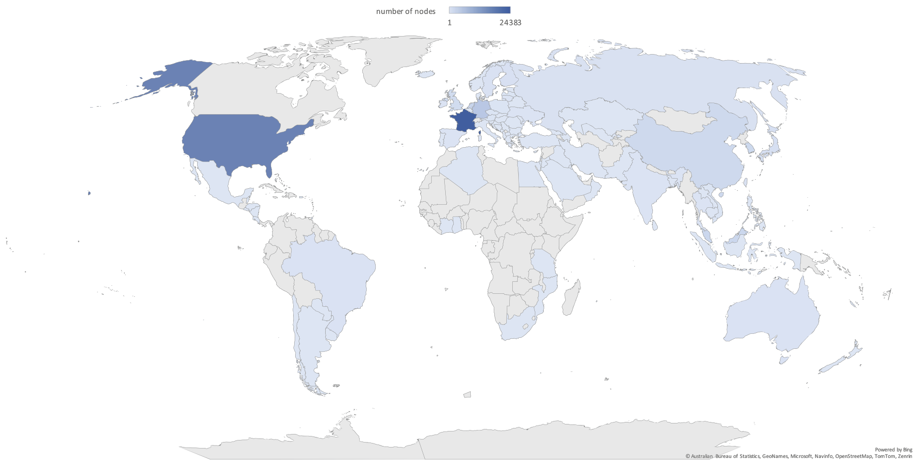
number of nodes
1
24383
Powered by Bing
© Australian Bureau of Statistics, GeoNames, Microsoft, Navinfo, OpenStreetMap, TomTom, Zenrin

Fig. 5. Number of IPFS nodes per country.

nodes with low latency. Therefore, connection to nodes in Asia were made only when no other choice was possible leading to this map.

Table 2 and Table 3 show the distribution of nodes according to network operators. Like countries, the network operator of a node was determined from its IP address. Table 2 shows surprisingly that only 11% of the IPFS nodes are hosted on Google cloud, Microsoft Azure or Amazon AWS. However, Table 3 shows that the vast majority of the nodes are hosted in traditional datacenters and not in home networks. Providers like OVH, Packet, DigitalOcean hosts more than 45% of the IPFS instances. Only the last provider, T-Mobile is a provider for home networks. This questions the true decentralization of the IPFS network.

Table 2. Number of IPFS nodes hosted in famous cloud providers.

cloud provider	number of nodes identifiers	percentage
Amazon AWS	5 274	9.8%
Google	597	1.1%
Microsoft Azure	314	0.58%

In conclusion, the nodes are located in the most developed regions of the world which is not surprising. However, the more surprising result is that most of the node of the IPFS network are located in datacenters and not on the personal computers of the users.

Table 3. Number of IPFS nodes hosted in each Autonomous System (top 10).

network provider	number of nodes identifiers	percentage
OVH SAS (FR)	20 227	37%
PACKET (US)	5 737	10%
AMAZON-02 (US)	5 274	9.7%
AS-CHOOPA (US)	4 997	9.2%
DIGITALOCEAN-ASN (US)	3 218	5.9%
Contabo GmbH (DE)	2 995	5.5%
Verdina Ltd. (BLZ)	2 083	3.8%
Hetzner Online GmbH (DE)	1 950	3.6%
Hostkey B.v. (NLD)	1 895	3.5%
T-MOBILE-AS21928 (DE)	1 692	3.1%

5.3 Popularity of the Files

There are two ways to define the popularity of files:

– by looking at the replicas of each file - the file with the highest number of replicas is the most popular;
– by looking at the files that are requested the most in the captured DHT requests.

If we look at the number of replicas available for each file, we can build the Fig. 6. The figure shows a graph bar where each bar corresponds to a different file and the y-axis represents the number of replicas found for each file. We sorted the bars by the number of replicas.

It unsurprisingly shows a Zipf distribution with a very small number of popular files with a high number of replicas and a extra large number of files that are not popular and have fewer than 10 replicas. We can even see that 100 612 files (100612 files over 330638, approximately 30%) have less than 3 replicas.

Contrary to Balduf *et al.* [4] and as mentioned in Sect. 4, the requests that are not resolved are ignored. Therefore, we can observe a power-law distribution.

We observe that the file we found with the highest number of replicas has 11 515 replicas, and the second highest has 4 597 replicas. These files correspond to the "README" files of IPFS that are present by default on the nodes.

If we use the second criterion: by looking at the files requested in the intercepted requests, we can build Fig. 7. Like in the previous graph, each bar represents a file, and the height of the bar is the number of requests observed for the considered file. The bars are sorted according to their height.

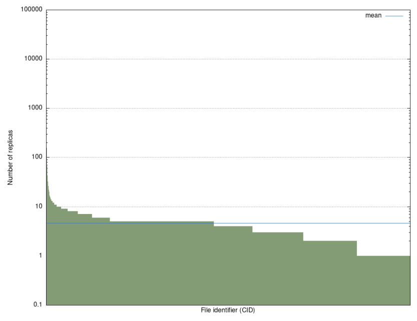

Fig. 6. Graph bar showing the number of replicas for each file (log scale in Y). Blue line represents the average number of replicas (average of 4.61). (Color figure online)

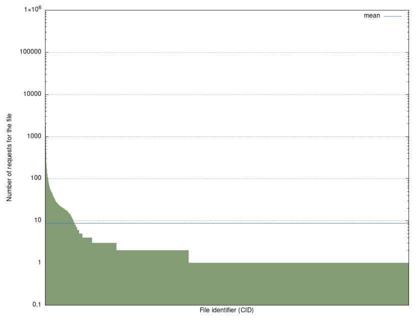

Fig. 7. Graph bar showing the number of requests for each file (log scale in Y). Blue line represents the average number of requests (average of 8.89). (Color figure online)

The first thing that we observe is that the average number of requests for each file is low (8.89 on average). If we look at the CID in the X-axis (not represented for readability), we observe that most queries are not about the files with most of the replicas (we do not observe any overlap in the 20 first CID of the two graphs in Fig. 6 and 7). In other words, the files with most of the replicas are not the most requested ones. This result seems strange because in IPFS each request on a new node leads to the creation of a new replica. This supposes that the popularity of files is changing over time, that many replicas are created when the file is popular, but then, the number of requests significantly reduces. This can explain why the files with the highest number of replicas are not the files with the most requests. We also note a bias in this figure. When a client requests a file to a node, the node looks for file replicas in the IPFS network by sending a DHT request. Then, a replica is downloaded, and a copy is created on the node that has received the request. Therefore, when a second client requests the same file to the node, no DHT request is sent in the network and can be captured. As a consequence, popular files that are requested by only a small set of IPFS nodes do not appear in the figure because DHT requests are not generated for each file access.

To conclude, the popularity of files shows a power-law distribution.

5.4 Activity in the Network

The fourth part of our analysis is about the use of IPFS network. We can show in Fig. 8 the number of requests intercepted for each day. We filtered the fact that nodes can send the same request several times within an interval of few

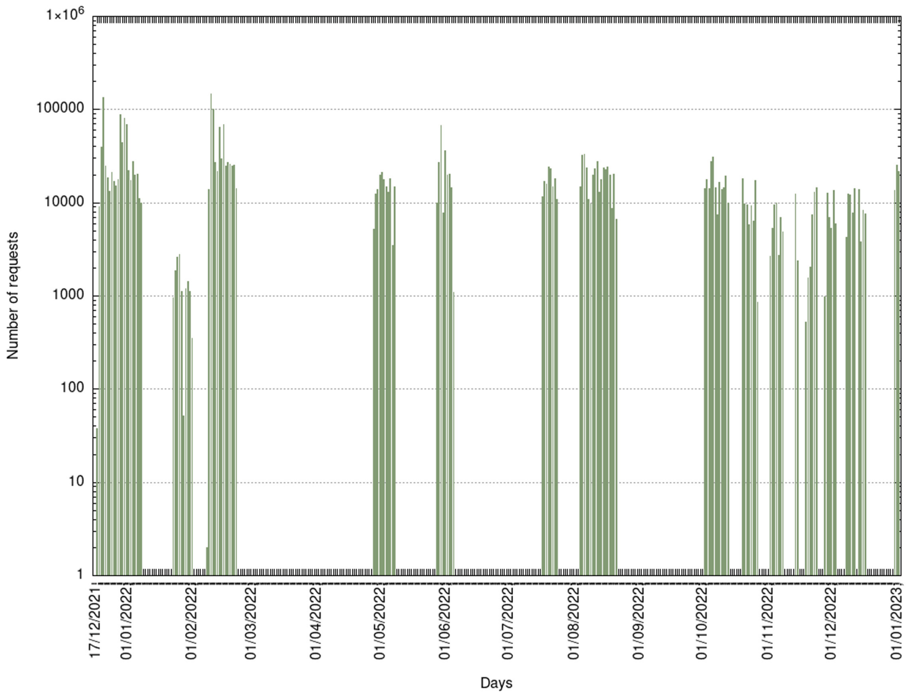

Fig. 8. Number of DHT requests received for each day.

seconds because it means that the node did not receive a response and retries to send the request.

Furthermore, we observe that the number of requests varies greatly with the days: 63 548 requests are sent on average each day, with a standard deviation almost equal to the average: 54 668 requests. However, we observe that, the number of requests at the end of January, was reduced significantly but increased again to reach the previous level on the 9th February.

5.5 Overhead of the DHT

A last question on the IPFS network is about its overhead in terms of network usage and energy consumption. In this part, we ignore the file transfer because if IPFS was replaced with a centralized solution, the same amount of data would have been exchanged. The only difference would have been that all flows originated from the same source rather than being distributed throughout the network. The real overhead of IPFS is the use of the DHT. DHT messages are composed of a message type on 4 bytes and the key that is looked for (on 38 bytes) Adding the IP and TCP headers, messages are 82 bytes long.

We received 9 278 089 requests. Because each request is 82 bytes long, we received $9278089 \times 82 = 760803298$ bytes in the period. From there, because the capture lasted 379 days 19 h 52 min, we can compute an average overhead throughput of 185 bps. However, the average throughput is not a relevant metric because, as it was shown in Fig. 8, requests are not received on a regular interval. If we look up at the peak of traffic on this figure, the maximum we observe is 37 687 requests on the 2nd February at 14:35. This corresponds to $37687 \times 82 = 3090334$ bytes for 60 s. Which is 412 044 bits per second (412 Kbps).

However, we saw in the previous section that more than 90% of nodes are hosted on a cloud computing platform and less than 10% are hosted on a computer connected to a consumer Internet Service Provider. Therefore, we can make the hypothesis that 90% of nodes are dedicated to IPFS and turned on only to run IPFS while 10% of nodes use the extra resources of computers used for other purposes. Nevertheless, all nodes are not necessarily physical ones, and the network is probably composed of virtual machines. The consequence is that running IPFS is not very different from running a cloud computing application in terms of energy consumption. The difference is mostly due to the fact that all nodes are not under the control of a single person with IPFS.

In conclusion, the overhead of the IPFS DHT in terms of network traffic is very low, but it is difficult to evaluate the number of dedicated physical nodes that have been added to the Internet to run the IPFS application.

5.6 Storage

In this last section, we study how the replicas are distributed among the IPFS nodes. We establish the list of peers, and for each file, we create a binary vector like (0 1 1) if we found that the file has a replica on the IPFS node 2 and IPFS node 3 but not on the IPFS node 1.

We therefore obtain a vector per file, and we can then compute correlations between files. A correlation of 1 indicates that the two files have their replicas on the same server, and a correlation of -1 indicates that the two files are stored on two complementary nodes.

Figure 9 shows a graph where each point is a file and the distance between two points is depending on the correlation value $distance = (1 - correlation)$. This distance means that two files that are stored on the same set of IPFS nodes are represented by two points at the same position (correlation = 1; distance = 0). On the contrary, two files that are stored on complementary nodes are represented far from each other (correlation = −1; distance = 2). We selected 200 files randomly.

It shows two clusters of files: one in the upper-right zone of the image and one in the bottom-left zone. The cluster of dots represented at the top of the figure corresponds to files that have only one replica on an IPFS server hosted on the network of DigitalOcean. The other cluster of dots, we can identify in the bottom of the figure, corresponds to the files that have a first replica on the node of the DigitalOcean network but also a second replica on a IPFS server hosted in the OVH network. All other files are files that have a unique placement of replicas: often a replica on a public gateway and another replica on the node of the user. These correspond to files that are not largely shared. This figure highlights different applications that use the IPFS network and have the tendency to place the file replicas on the same nodes.

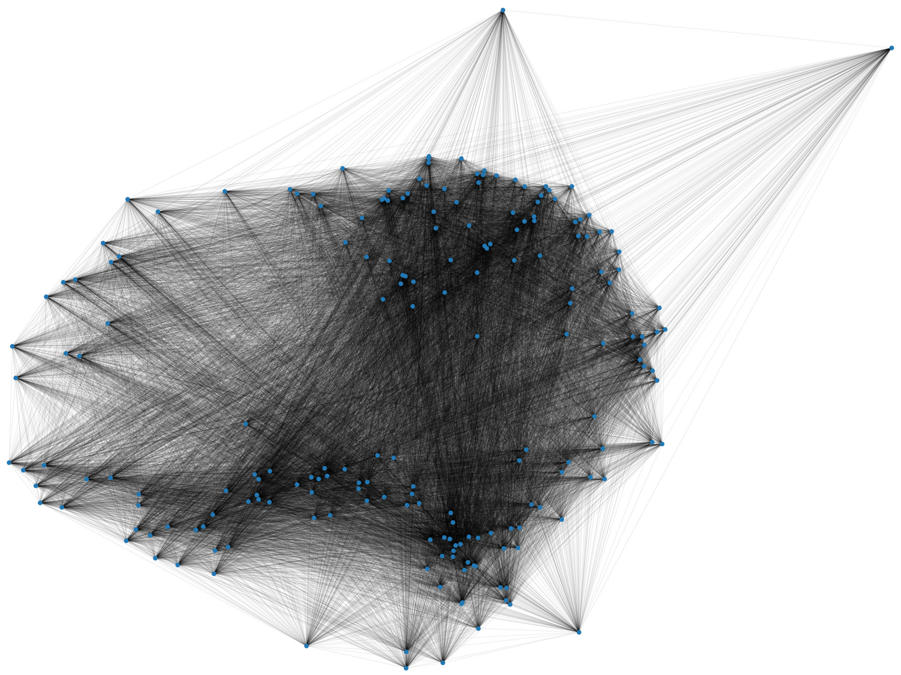

Fig. 9. Correlation between the storage of each file. Each dot represents a file and the edges between the nodes are depending on the servers the replicas are stored on.

5.7 Replication of Results on Amazon Web Services Platform (AWS)

In this last part, we try to determine if the observed results are reproducible. We run the same study by collecting DHT requests from a node hosted on Amazon Web Services[7]. This study was made over a shorter period of time: only 10 months from January to November 2022. Results show similar observations to those made using the node hosted on Google Cloud Platform in Table 1. We collected 593 451 DHT requests. We collected far fewer requests than on the Google node because of its latency and therefore probably a low score in the buckets of the Kademlia DHT.

File type distribution is similar to the distribution we observed on Google node. 75% of observed files are JSON objects and 18% are binary files as illustrated in Table 4.

Regarding the distribution of requests received, we observe the same Zipf distribution as we saw previously. Few files are requested many times and many others are requested only once. This is illustrated in Fig. 10. This short analysis with another node reinforces the observations we made over a longer period with the node hosted on Google Cloud Platform from Sect. 5.1 to Sect. 5.6.

6 Discussion

These results are surprising in the sense that IPFS is designed to share files efficiently with a large number of users. However, we observe that most of the files have few replicas and are therefore not shared with a large group of people. We can therefore wonder why these people use the IPFS solution that does not seem ideal because the BitTorrent-like protocol is not exploited to its full capabilities: we have not observed files with many replicas spread on a large set of nodes and we do not observe files becoming popular over time.

A possible reason is that IPFS is used for the use cases listed in Sect. 2.3 like non-repudiation storage [24] or traceability in supply chains [3]. IPFS seems to be more used for its immutability property that preserves files from modifications, allowing users to guarantee the integrity of the distributed files than for the file distribution around the globe. The distribution of accesses can also let us think that IPFS might be used by technical people who try it by creating a file, requesting it and leaving the network. In addition to this, we note that an IPFS gateway provides HTTP links to stored files, which can then be distributed easily among the users, making IPFS an alternative to services like drives (Dropbox, Onedrive, Google Drive) or File Transfer services (Wetransfer). Finally, as we observed many HTML files and images, we can tell that IPFS is used as a web hosting server. Websites are then accessed through IPFS gateways, achieving the scalability of web services with almost no extra cost (IPFS is acting like a Content Delivery Network).

[7] https://aws.amazon.com/.

Table 4. Types of files observed in the IPFS network from the AWS-hosted node (top 10)

MIME type	Number of occurrences	percentage
application/json	5 141	76%
application/octet-stream	1 222	18%
text/plain	168	2.5%
image/png	82	1.2%
image/jpeg	55	0.81%
text/html	24	0.35%
application/pdf	21	0.31%
application/gzip	16	0.24%
application/x-dosexec	12	0.18%
image/svg+xml	8	0.12%

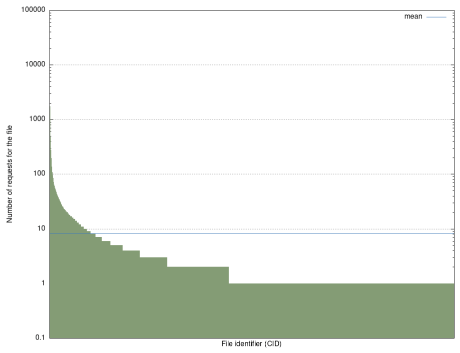

Fig. 10. Graph bar showing the number of requests for each file (log scale in Y). Blue line represents the average number of requests (8.22). (Color figure online)

A last remark is the concern about the need for security measures to preserve the users' privacy. In the current state of development, IPFS does not seem to give more guarantees to protect users privacy than the big cloud providers (GAFAM) do. It seems relatively easy for a malicious user to deploy many IPFS nodes in order to track most activities on the network. This tracking can be mitigated by spreading more replicas, even on nodes that do not access the piece of data. This prevents the DHT from being used when access to the data is made, because with IPFS, DHT requests are only sent when a local copy of the requested data is not found on the local node.

Also, a simple solution would be to encrypt both DHT records and files. The encryption key could be appended to the file identifier like "<CID>-<encryption key>". The user would still look for the CID in the DHT (the key would not be transmitted to the DHT) but the retrieved record is encrypted, and the key in the second part of the identifier must be known to decrypt it and to locate the replicates. Such a solution is not ideal because observing the requests in the DHT still leads to knowing the list of users that are accessing files, but the number of replicas, their location as well as the type of file could not be easily determined. Some articles [13,29] and the IPFS website list different projects that exploit content encryption[8] but most of them only encrypt data without considering the DHT.

Another solution would be to manage storage pools. The DHT would contain "pool1 managed on node1, node2 and node3". Then, the objects would be identified with the pool they belong to "<CID>-<pool>". In this way, the user locates the pool in the DHT, then directly contacts the nodes of the pool to get the replica. In this way, the DHT requests do not indicate the file the user is looking for, and observing the requests does not indicate how many files are stored in the network. The drawback of such a solution is that it makes the placement more difficult: many files must be stored together in order to avoid having a pool for each file, which recreates the problem we are trying to solve.

7 Conclusion

This study exploits the pieces of information publicly available on the IPFS network. These pieces of information led us to characterize the IPFS network by identifying the network operators through which, nodes are connected. It also enabled us to determine the popularity of files, the type of files exchanged, and the number of replicas of each file. We showed in Sect. 5.1 that in addition to the many PDF or image files stored, IPFS is also used as a hosting provider for websites. In Sect. 5.2, we showed that most nodes are from France and Germany in Europe and from the United States for the America continent. We then studied in Sect. 5.3 and Sect. 5.4 the frequencies of requests and their distribution along the days. We showed that a very small number of nodes send a lot of DHT requests and that most of the files are not popular, questioning us about the centralization of the network. In Sect. 5.5, we evaluated the overhead of the DHT of IPFS and concluded that it only consumes 185 bps which is very low even if peaks require more throughput. Before finishing, in Sect. 5.6, we identified in a subset of files two sets of files with their replicas placed on the same nodes. These files correspond to different applications relying on IPFS. These results have been obtained using an IPFS node hosted on Google Cloud Platform but were replicated on Amazon Web Services (AWS) in Sect. 5.7. In this situation, we can wonder what the interest is, in using IPFS for storing non-popular files because the main advantage of IPFS is the distribution of popular files. We considered several hypotheses, like non-repudiation storage or just because of the simplicity of sharing data compared to what cloud drives propose. We can also question the security of the IPFS protocol because, with few resources, it can be feasible for a malicious user to track many of the accesses performed.

[8] https://docs.ipfs.tech/concepts/privacy-and-encryption/#encryption.

We also note that this study has one major drawback: accessing files that have a replica stored on the IPFS node interrogated did not generate a DHT request and therefore could not be captured and analyzed. Evaluating the proportion of this kind of request is a bit hard to do as IPFS is a decentralized network.

Finally, this work can be continued by designing mechanisms to manage access permissions on files stored in IPFS or making improvements in the DHT in order to prevent the possibility of tracking the users' activity.

References

1. Abdullah Lajam, O., Ahmed Helmy, T.: Performance evaluation of IPFS in private networks. In: 2021 4th International Conference on Data Storage and Data Engineering. DSDE 2021, New York, NY, USA, pp. 77–84. Association for Computing Machinery (2021). https://doi.org/10.1145/3456146.3456159
2. Ali, M.S., Dolui, K., Antonelli, F.: IoT data privacy via blockchains and IPFS. In: Proceedings of the Seventh International Conference on the Internet of Things. IoT 2017, New York, NY, USA. Association for Computing Machinery (2017). https://doi.org/10.1145/3131542.3131563
3. Altmann, P., Abbasi, A.G., Schelén, O., Andersson, K., Alizadeh, M.: Creating a traceable product story in manufacturing supply chains using IPFS. In: 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), pp. 1–8 (2020)
4. Balduf, L., Henningsen, S., Florian, M., Rust, S., Scheuermann, B.: Monitoring data requests in decentralized data storage systems: a case study of IPFS (2021). https://arxiv.org/abs/2104.09202
5. Benet, J.: IPFS - Content Addressed, Versioned, P2P File System. Technical report, Protocol Labs, Inc. (2014). http://arxiv.org/abs/1407.3561
6. Benet, J., Greco, N.: Filecoin: A decentralized storage network. Technical report, Protocol Labs, Inc. (2018)
7. Bisong, E.: Containers and Google Kubernetes Engine. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform, pp. 655–670. Apress, Berkeley, CA (2019). https://doi.org/10.1007/978-1-4842-4470-8_45
8. Daniel, E., Tschorsch, F.: Passively Measuring IPFS Churn and Network Size (2022). https://arxiv.org/abs/2205.14927
9. Doan, T.V., Pham, T.D., Oberprieler, M., Bajpai, V.: Measuring decentralized video streaming: a case study of DTube. In: 2020 IFIP Networking Conference (Networking), pp. 118–126 (2020)
10. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google file system. SIGOPS Oper. Syst. Rev. **37**(5), 29–43 (2003). https://doi.org/10.1145/1165389.945450
11. Hasan, H.R., Salah, K., Yaqoob, I., Jayaraman, R., Pesic, S., Omar, M.: Trustworthy IoT data streaming using blockchain and IPFS. IEEE Access **10**, 17707–17721 (2022)
12. Henningsen, S., Rust, S., Florian, M., Scheuermann, B.: Crawling the IPFS network. In: 2020 IFIP Networking Conference (Networking), pp. 679–680 (2020)
13. Karapapas, C., Pittaras, I., Polyzos, G.C.: Fully decentralized trading games with evolvable characters using NFTs and IPFS. In: 2021 IFIP Networking Conference (IFIP Networking), pp. 1–2 (2021). https://doi.org/10.23919/IFIPNetworking52078.2021.9472196

14. Khatal, S., Rane, J., Patel, D., Patel, P., Busnel, Y.: FileShare: a blockchain and IPFS framework for secure file sharing and data provenance. In: Patnaik, S., Yang, X.-S., Sethi, I.K. (eds.) Advances in Machine Learning and Computational Intelligence. AIS, pp. 825–833. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-5243-4_79
15. Koshy, P., Koshy, D., McDaniel, P.: An analysis of anonymity in bitcoin using P2P network traffic. In: Christin, N., Safavi-Naini, R. (eds.) FC 2014. LNCS, vol. 8437, pp. 469–485. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45472-5_30
16. Krejci, S., Sigwart, M., Schulte, S.: Blockchain- and IPFS-based data distribution for the internet of things. In: Brogi, A., Zimmermann, W., Kritikos, K. (eds.) ESOCC 2020. LNCS, vol. 12054, pp. 177–191. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44769-4_14
17. Legout, A., Urvoy-Keller, G., Michiardi, P.: Understanding BitTorrent: An Experimental Perspective. Technical report (2005). https://hal.inria.fr/inria-00000156
18. Maymounkov, P., Mazières, D.: Kademlia: a peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_5
19. Nguyen, T.-T., Do, B.-L.: A novel model using CDN, P2P, and IPFS for content delivery. In: Dang, T.K., Küng, J., Takizawa, M., Chung, T.M. (eds.) FDSE 2020. CCIS, vol. 1306, pp. 51–62. Springer, Singapore (2020). https://doi.org/10.1007/978-981-33-4370-2_4
20. Nizamuddin, N., Salah, K., Ajmal Azad, M., Arshad, J., Rehman, M.: Decentralized document version control using ethereum blockchain and IPFS. Comput. Electr. Eng. **76**, 183–197 (2019). https://www.sciencedirect.com/science/article/pii/S0045790618333093
21. Partridge, C., Allman, M.: Addressing ethical considerations in network measurement papers: abstract. In: Proceedings of the 2015 ACM SIGCOMM Workshop on Ethics in Networked Systems Research. NS Ethics 2015, New York, NY, USA, p. 33. Association for Computing Machinery (2015). https://doi.org/10.1145/2793013.2793014
22. Shen, J., Li, Y., Zhou, Y., Wang, X.: Understanding I/O performance of IPFS storage: a client's perspective. In: Proceedings of the International Symposium on Quality of Service. IWQoS 2019, New York, NY, USA. Association for Computing Machinery (2019). https://doi.org/10.1145/3326285.3329052
23. Small, N., Meneghello, J., Lee, K., Sabooniha, N., Schippers, R.: A discussion on the ethical issues in peer-to-peer network monitoring (2012)
24. Sun, J., Yao, X., Wang, S., Wu, Y.: Non-repudiation storage and access control scheme of insurance data based on blockchain in IPFS. IEEE Access **8**, 155145–155155 (2020)
25. Trautwein, D., et al.: Design and evaluation of IPFS: a storage layer for the decentralized web. In: Proceedings of the ACM SIGCOMM 2022 Conference. SIGCOMM 2022, New York, NY, USA, pp. 739–752. Association for Computing Machinery (2022). https://doi.org/10.1145/3544216.3544232
26. Vorick, D., Champine, L.: SIA: Simple Decentralized Storage. Technical report, NebulousLabs, Boston (2014)
27. Williams, S.A., Diordiiev, V., Berman, L.: Arweave: A Protocol for Economically Sustainable Information Permanence. Technical report, Minimum Spanning Technologies Limited (2019)

28. Wolchok, S., Halderman, J.A.: Crawling BitTorrent DHTs for fun and profit. In: Proceedings of the 4th USENIX Conference on Offensive Technologies. WOOT 2010, pp. 1–8. USENIX Association, USA (2010)
29. Zhou, C., Sun, G., You, X., Gu, Y.: A slice-based encryption scheme for IPFS. Int. J. Secure. Network. **18**(1), 42–51 (2023). https://doi.org/10.1504/IJSN.2023.129898. https://www.inderscienceonline.com/doi/abs/10.1504/IJSN.2023.129898