



École Polytechnique Fédérale de Lausanne

Dissecting IPFS and Swarm to demystify
distributed decentralized storage networks

by Sixiao Xu

Master Semester Project Report

Prof. Bryan Ford
Project Advisor

Vero Estrada-Galiñanes
Project Supervisor

EPFL IC IINFCOM DEDIS
BC 210 (Bâtiment BC)
Station 14
CH-1015 Lausanne

January 6, 2023

Contents

1	Introduction	4
2	Background	5
2.1	Distributed and Decentralized Storage	5
2.2	IPFS and Swarm	5
2.2.1	Network Architecture	6
2.2.2	Peer Discovery	6
2.2.3	Content Addressing and CID	7
2.3	IPFS Gateway	7
2.4	Incentives	8
2.5	Related Work	8
3	Visualization and Analysis of IPFS Gateway Dataset	10
3.1	Dataset Description	10
3.2	Data Preprocessing	11
3.2.1	Data Cleaning	11
3.2.2	Extracting CID	12
3.3	Data Analysis	12
3.3.1	Time Series Analysis	12
3.3.2	CID Popularity	15
3.3.3	Agent Request	20
3.3.4	Agent Churn Rate	24
3.4	Conclusion	27
4	Performance Evaluation of IPFS and Swarm nodes	28
4.1	Experiment Environment	28
4.2	Experiment Design	28
4.2.1	Monitoring Resource Consumption	28
4.2.2	Observing Connected Peers	29
4.3	Installation Process	29
4.4	Measurement Procedure	30
4.5	Experiment Results	30
4.5.1	Resource Consumption	31

4.5.2 Connected Peers	34
4.6 Conclusion	37
5 Limitation and Future Work	38
Bibliography	39

Chapter 1

Introduction

Decentralized networks have emerged as a promising solution for building secure and decentralized infrastructure for a variety of applications. Decentralized storage networks, in particular, play a crucial role in hosting and distributing large amounts of data in a decentralized manner. One of the key challenges is developing effective incentive protocols to reward participating nodes for their contributions to the network, which is essential for maintaining the network's health and functionality.

In this project, we aim to provide a comprehensive analysis of two popular decentralized storage networks: IPFS and Swarm. By studying these networks, we hope to gain a deeper understanding of some characteristics, such as workload and user behavior, and provide insights that will help make realistic assumptions and design incentive algorithms that take into consideration peer behavior patterns.

In chapter 2, we first provide a background on distributed and decentralized storage systems, with a focus on IPFS and Swarm. We describe the network architecture, peer discovery, content addressing and CID. We also discuss the role of IPFS Gateway in the network. In addition, we review related work in the field.

In chapter 3, we describe the dataset used for visualization and analysis, and the data preprocessing steps of cleaning and extracting CID. We then present the results of our data analysis, regarding time series analysis, CID popularity, agent request and churn rate.

In chapter 4, we first outline the experiment environment and design, including the monitoring of resource consumption and observation of connected peers. Then, we describe the installation process and measurement procedure. Finally, we present the results of the experiment.

Chapter 2

Background

2.1 Distributed and Decentralized Storage

Distributed storage refers to a type of data storage system in which data is stored across multiple devices or nodes, rather than on a single, central server. This type of storage can be either decentralized or centralized, depending on how the nodes are organized and how data is distributed among them.

Decentralized storage specifically refers to a type of distributed storage system in which the nodes are organized in a peer-to-peer network and data is distributed across all the nodes. This type of storage is characterized by its lack of a central authority or controlling entity, which makes it more resilient, secure, and efficient than traditional, centralized storage systems.

2.2 IPFS and Swarm

IPFS (InterPlanetary File System) [1] is a decentralized, peer-to-peer file-sharing system that enables users to store and share data in a distributed manner. It works by allowing users to access files from multiple computers on a network, rather than from a single, central server. This distributed nature of IPFS makes it a powerful tool for storing and sharing data in a decentralized way.

Swarm [2] is another decentralized storage platform and content distribution service, it is a native base layer service of the Ethereum web3 stack. It is built on top of the Ethereum blockchain and utilizes its underlying technology to provide security and reliability for data storage and sharing [3].

However, there are a few differences between the two systems. One main difference is that

Swarm was born as part of the Ethereum ecosystem with incentives embedded in the system. It can be used in conjunction with smart contracts and other Ethereum-based technologies. While IPFS operates independently, incentives can be added by combining it with Filecoin (Described in section 2.4). Also, IPFS is primarily used for storing and sharing immutable content, while Swarm is utilized for distributing dynamic content and for hosting decentralized applications (dApps). Both IPFS and Swarm are powerful tools for decentralized storage and file sharing, and both systems have their own unique features and capabilities.

2.2.1 Network Architecture

In distributed networks, Distributed Hash Tables (DHTs) are often used for nodes to efficiently store and retrieve data. Kademlia [4] is a popular DHT. It uses a XOR-based metric topology and minimizes the number of control messages, providing efficient lookup through massive networks. The system has provable consistency and performance in a fault-prone environment.

Both IPFS and Swarm builds an overlay network, which is a logical network of nodes on top of the real network. The nodes need to be identified in the network. IPFS nodes are identified by a `NodeId`, the cryptographic hash of a public key, generated with S/Kademlia's static crypto puzzle [5]. While in Swarm the identity of a node is created from the network ID and its Ethereum address.

To discover other peers in the network, as well as find peers who can serve particular content, it is necessary to have a routing system. Both IPFS and Swarm uses DHTs. A peer in IPFS connects to every other peer it comes across, creating an unstructured network. Swarm, on the other hand, creates a Kademlia topology, where the neighbors are determined by identity, leading to a structured network [6].

2.2.2 Peer Discovery

The ways IPFS and Swarm discover peers are different. Upon instantiating a node, IPFS first performs lookups for its own node ID and some random IDs for each bucket, so it knows all nodes in its direct neighborhood and some nodes from each bucket. It also accepts every incoming connection, and starts a new connection with every node it encounters when requesting data [7]. Connections will be pruned if the number of connections exceeds a certain limit, except for those actively used or required by the DHT. IPFS's routing system find other peers using a distributed sloppy hash table (DSHT) based on S/Kademlia and Coral [1].

On the other hand, Swarm uses RLPx, which implements node discovery based on the routing algorithm of Kademlia [8]. When a node is freshly started, it initially adds a hard-coded set of bootstrap node IDs to its routing table in order to find peers. Then, when locating a node, it searches its routing table and performs the Kademlia algorithm to discover new nodes.

2.2.3 Content Addressing and CID

IPFS uses content addressing to address and retrieve files and other content on the network. In contrast to traditional addressing schemes, which use location-based addresses (such as URLs) to identify content, content addressing uses the content itself as the identifier.

The identifiers in IPFS are called CID (Content Identifier). A CID is made up of two parts: the content's multihash, which is a cryptographic hash of the content, and the content's multicodec, which specifies the format of the content. Together, these two components provide a unique and tamper-resistant identifier for the content on the IPFS network. The same content will always have the same CID, regardless of where it is stored or how it is accessed.

Similar to IPFS, Swarm also uses content-based addressing. However, the content-based addressing in Swarm also decides the storage location. Close neighbors of the storage node also provide chunk redundancy.

2.3 IPFS Gateway

IPFS gateways provide HTTP-based service to access IPFS content for applications that do not support IPFS natively. In this study, we focus on read-only HTTP gateways that fetch content from IPFS via HTTP GET method. Any IPFS gateway can resolve access to any requested CID, regardless of the gateway provider or location. There are some public gateway operators. For example, Protocol Labs, which deploys the public gateway <https://ipfs.io>. This gateway is a community resource to help developers build on IPFS.

In IPFS gateways, three resolution styles are employed:

1. Path: <https://gatewayURL/ipfs/content ID/optional path to resource>
2. Subdomain: <https://CID.ipfs.gatewayURL/optional path to resource>
3. DNSlink: <https://gatewayURL/ipns/IPNSIdentifier/optional path to resource>

The last method is used in scenarios when an application needs to access the latest version of a file or website but is unable to determine the specific CID for that latest version. Using the InterPlanetary Name Service (IPNS), an version-independent IPNS identifier can be resolved into the current version's IPFS CID.

2.4 Incentives

In a decentralized storage network, incentives are used to motivate participants to contribute their resources such as storage, bandwidth, and computing power to the network. The incentives can take a variety of forms, including financial rewards, reputation points, or access to premium features.

Filecoin and IPFS are two protocols developed by Protocol Labs that work together to create a decentralized storage network. IPFS allows peers to share and store data with each other, while Filecoin provides a system for storing data persistently. Under Filecoin's incentive structure, clients pay to store their data with a specific level of redundancy and availability, while storage providers earn payments and rewards by continuously storing and proving the data. These two protocols are separate but complementary, and can be used independently or together.

Swarm has two main incentive systems: Bandwidth Incentives and Storage Incentives. Bandwidth Incentives use the Swarm Accounting Protocol (SWAP) [9] to motivate nodes to contribute their bandwidth to the network. When information is retrieved, each node keeps track of the balance of payments between it and other peers. If a peer's debt becomes too large, it is disconnected and its reputation suffers. As for Storage Incentives, nodes that are willing to provide long-term storage put up a security deposit and sell storage services, promising to store certain chunks. These nodes can be challenged and, if they cannot prove that they are storing the promised chunk, they lose their security deposit.

2.5 Related Work

There has been a growing interest in decentralized storage networks in recent years, as they provide various advantages over conventional centralized storage systems.

One paper [10] describes the design and implementation of IPFS, and evaluates the deployment, usage and performance of the IPFS network. The evaluation data in this paper is published and available for further research. We will use the IPFS gateway usage dataset in subsequent sections. In the paper, results mainly focus on object size, retrieval delays, cache hit rates and gateway referrals. While in this project, we want to explore more on content popularity and user behavior.

Furthermore, several studies have focused on measuring the performance of decentralized storage networks. One paper [8] presents a study of the Ethereum network, measuring the properties of the nodes in long term, and compared it with other P2P networks. Other work [11] studies the I/O performance of IPFS storage with different access patterns, identifying the factors that affect performance and propose optimization to avoid high-latency operations. Another paper [12] presents an analysis regarding storage and retrieval of data stored on the Ethereum

blockchain, comparing hybrid approaches by using IPFS and Swarm as storage. Other study [13] evaluates the performance of IPFS uploading and retrieving operations in private networks, and compared with the File Transfer Protocol (FTP).

In addition to performance, other aspects of decentralized storage networks have also been studied. One paper [7] presents the results of crawling the IPFS' DHT, concluding that the topology is closer to an unstructured overlay network. Other work [14] presents a methodology for passive monitoring of requests in IPFS. Through their analysis, they gain insight into the size and structure of the IPFS network, and content popularity distributions. Other study [15] provides a passive measurement of IPFS peer dynamics, and estimates the network size.

Bandwidth incentives within Swarm have been investigated and simulated in this paper [16]. The author used Gini coefficient to define two fairness characteristics to evaluate reward sharing in decentralized storage networks, and demonstrated an approach to make the current bandwidth incentives more equitable.

Chapter 3

Visualization and Analysis of IPFS Gateway Dataset

In this section, we analyze an IPFS gateway usage dataset containing requests taken from a public IPFS gateway, to gain insights about traffic and user behavior.

3.1 Dataset Description

Protocol Labs published a gateway usage dataset covering 7.1M user requests in January 2022 [10]. Each entry contains information of a single user request. The dataset is available on IPFS with the CID: `bafybeiftyvcar3vh7zua3xakxkb2h5ppo4giu5f3rkpsqgcfh7n7axxnsa`

The dataset contains the following fields:

1. Encrypted IP address of the caller
2. Server timestamp of the request
3. HTTP request information (method, path, version)
4. HTTP response status code
5. Bytes returned (body bytes sent)
6. Request length
7. Request time
8. Upstream response time

9. Upstream header time
10. Cache hit/miss
11. HTTP referrer
12. User agent
13. Server name
14. HTTP host
15. HTTP schema

The evaluations in the original research concentrate on object size, retrieval delays, cache hit rates, and gateway referrals. For the purpose of this project, we are looking for patterns in content popularity, user behavior, and network traffic. We will focus mainly on timestamp, bytes returned, user agent, path, and http host.

The raw data contains personal information, such as IP addresses and CID, that are anonymized due to ethical concerns. Still, unique users can be partially distinguished by agents. And, we can extract CID from the path and http host. In the paper, there are 101k users accessing 274k unique CIDs. Using this dataset, we can find 21.26k agents and 254.57K CIDs after data cleaning.

3.2 Data Preprocessing

This section describes the procedures used to transform the raw data into csv files for further analysis and interpretation. The relevant code is in the Data_Cleaning notebook.

3.2.1 Data Cleaning

There are 7,169,922 entries in the original log file. After applying the following steps for each row, we get 6,645,871 rows in a dataframe.

1. Split the fields by white space and special characters.
2. Drop rows with missing values.
3. Remove scrambled text.
4. Keep only GET requests.

5. Filter agents starting with white space or special characters.
6. Remove empty path.
7. Extract CID.
8. Remove empty or invalid CID.
9. Parsing dates.
10. Keep only data in 02/01/2022.

3.2.2 Extracting CID

The requested CID is not explicitly listed in the dataset. However, we can extract it from other fields. The request path corresponds to the gateway path. *https://ipfs.io/* is omitted. A sample record is shown in table 3.1. The http host corresponds to the subdomain resolution style, and *https://* is omitted. A sample record is shown in table 3.2.

resolution style	https://{gatewayURL}/ipfs/{CID}/{optional path to resource}
example	/ipfs/QmewCrTqsMEC*****1QBsjxjAgZSwfy/9033
extracted CID	QmewCrTqsMEC*****1QBsjxjAgZSwfy

Table 3.1: Extracting CID from request path example

resolution style	https://{CID}.ipfs.gatewayURL/optional path to resource
example	bafybeifqhn*****674xu24kxjt7j25ebw2tej5wiiqy.ipfs.dweb.link
extracted CID	bafybeifqhn*****674xu24kxjt7j25ebw2tej5wiiqy

Table 3.2: Extracting CID from http host example

3.3 Data Analysis

In this section, we discuss our findings regarding traffic, users, and files from the dataset. The relevant code is in the correspondingly named notebooks.

3.3.1 Time Series Analysis

In this section, we aim to understand how the total number of requests, size, and traffic change over the course of a day. By analyzing this data, we can identify patterns and trends in usage of the network.

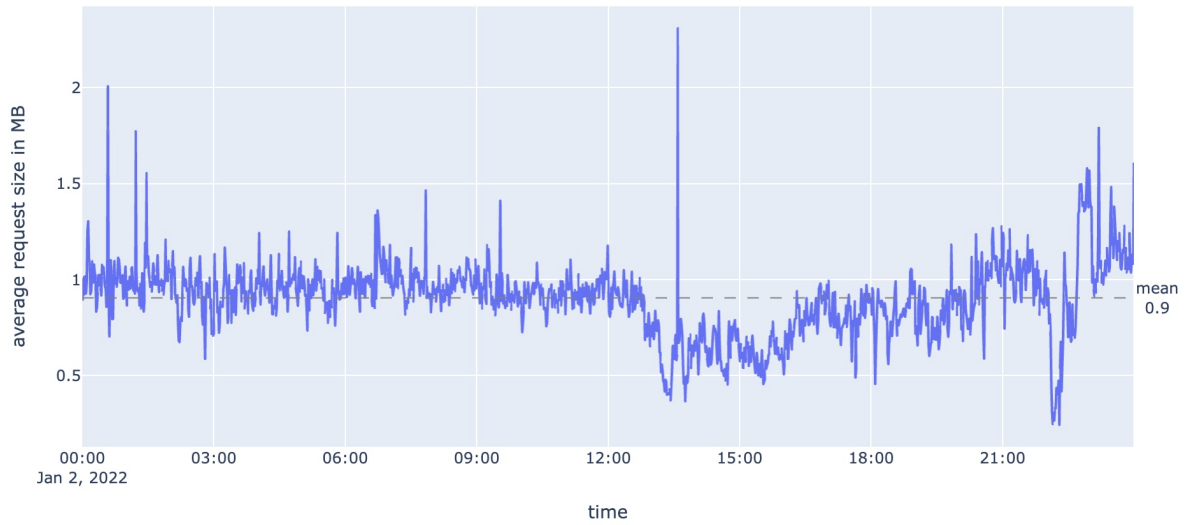


Figure 3.1: Average request size per minute

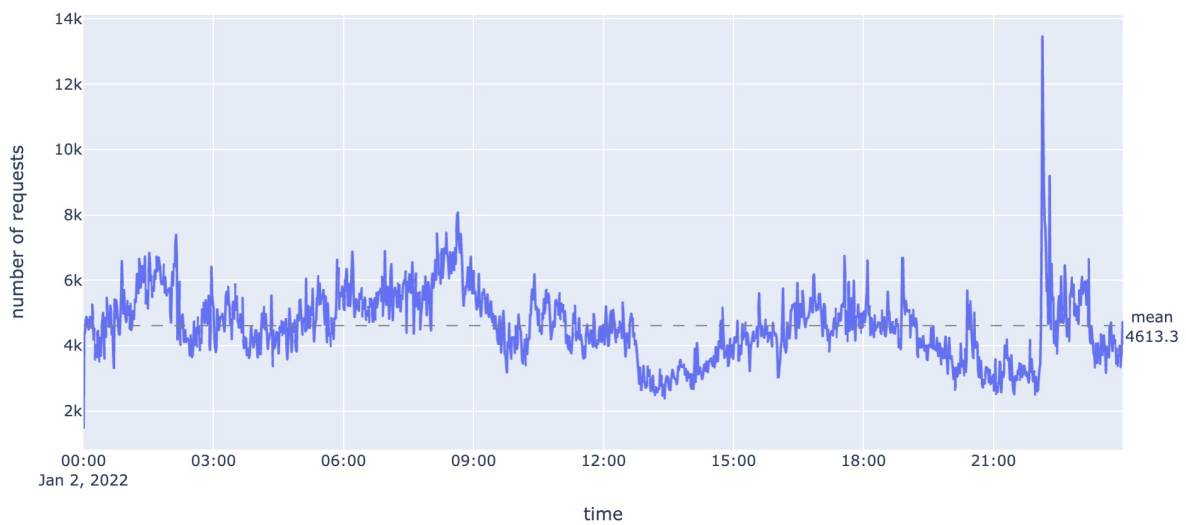


Figure 3.2: Number of requests per minute

Figure 3.1 shows the average request size in each minute during the day, with a mean of 0.9MB. Figure 3.2 shows the number of requests captured in each minute, with a mean of 4613.3. The fluctuations in the data can be attributed to a variety of factors. For example, changes in users with different interests can impact the number and size of requests. Additionally, the time of day can also play a role, as there may be more or less activity during different hours. These fluctuations are normal and to be expected, but they can also provide insights into the

performance of the system and help identify potential areas for improvement.

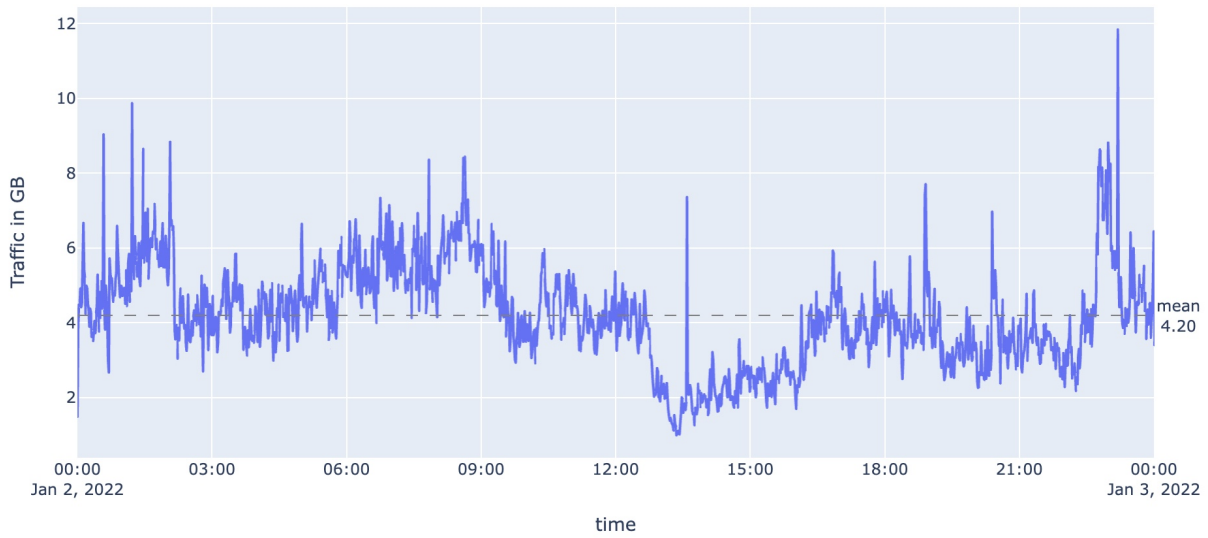


Figure 3.3: Average traffic

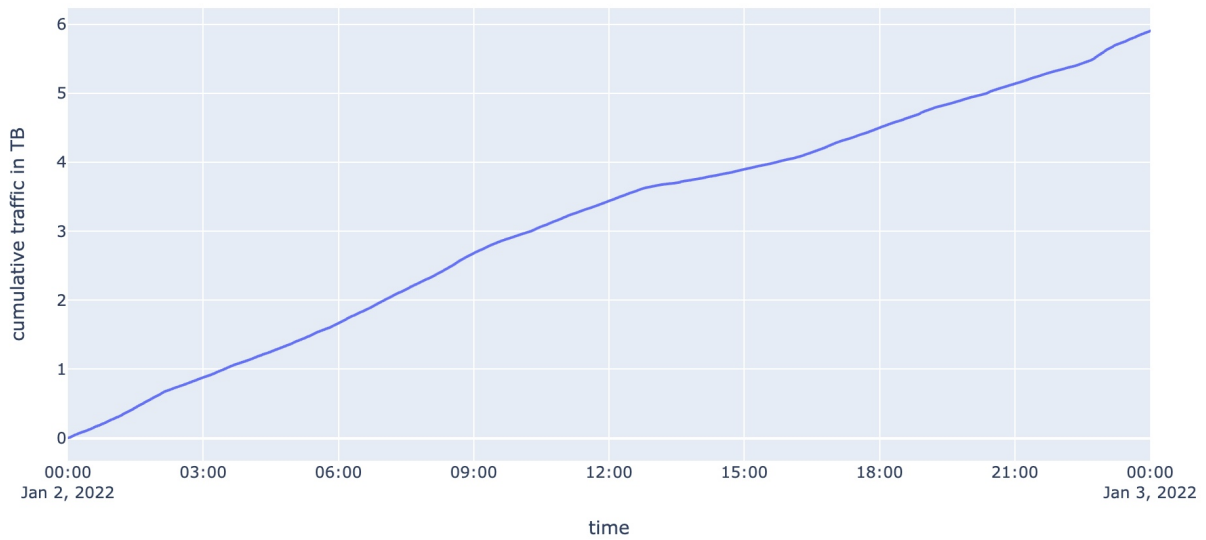


Figure 3.4: Cumulative traffic

From figure 3.3 we can see that there is a significant amount of traffic on the system, with an average of 4.20GB per minute. This indicates that the system is handling a large volume of data and is likely serving a large number of users.

Additionally, the steady and continuous build-up of traffic shown in Figure 3.4 suggests that the system is handling the demand well and is not experiencing any sudden spikes or drops in traffic. This is a good sign, as sudden changes in traffic can impact the performance of the system and potentially cause disruptions or issues.

Overall, these figures suggest that the system is handling a large volume of requests with relatively stable performance.

3.3.2 CID Popularity

Analyzing the popularity of CIDs can be useful in different ways. For example, analyzing the request frequencies of CIDs can help identify which CIDs are being accessed most frequently and which are being accessed less often. This can provide insights into the types of content that are most popular or widely used. Additionally, examining the relationship between request frequency and other variables, such as file size or the number of unique users who requested a CID, can help identify correlations and patterns that may be present. This can inform strategies for optimization.

File Size and Traffic

We first tally the size distribution of all requested files. Then group the requests by file size and calculate each group's contribution to the total requests and traffic across the network. By performing this analysis, we can gain insights into which types of files and how much resources are being devoted to serving these requests.

mean	std	min	25%	50%	75%	max
804.98	7575.89	0	0.41	10.77	282.79	2702699.65

Table 3.3: Statistics on the requested file size (in KB)

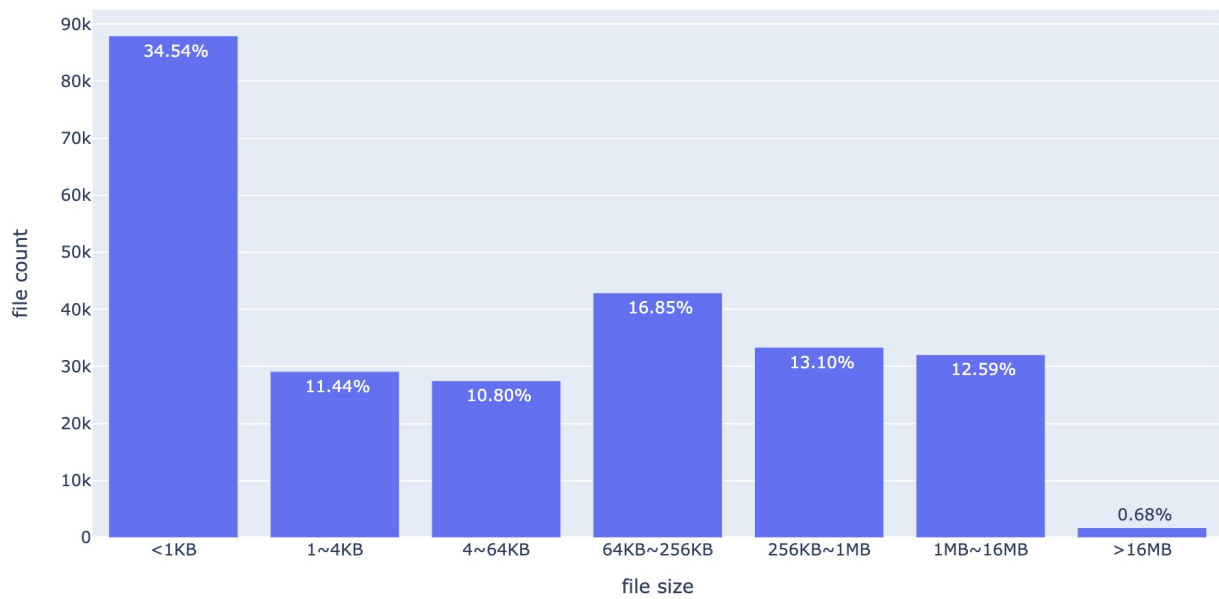


Figure 3.5: Number of files in each size range

The data shown in Table 3.3 and Figure 3.5 provides insight into the distribution of the file sizes being accessed in the network. We can see that a significant portion of the files (34.54%) are below 1KB in size, indicating that many of the requests being made are for small files. Additionally, the average and median file sizes are 804.98KB and 10.77KB, respectively, suggesting that the majority of the files being accessed are relatively small. However, there is also a significant number of large files, with the largest file being 2.58GB in size.

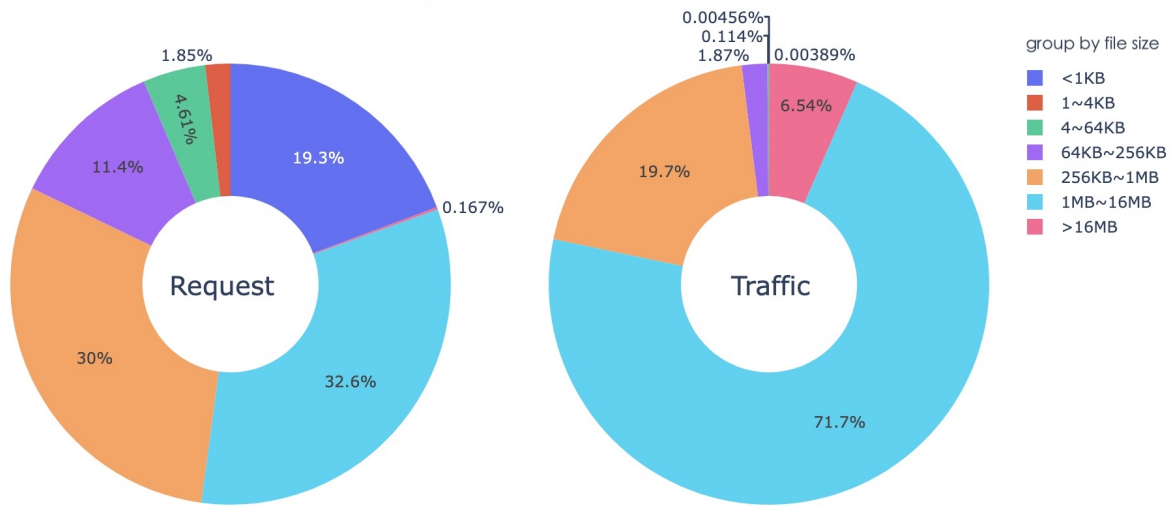


Figure 3.6: Percentage of requests and traffic in each size range

Then, for each range of request times, we group the files and count the total number of requests and traffic. Figure 3.6 provides further insight into the relationship between file size and request volume. We can see that while small files under 1KB make up a significant portion of requests (19.3%), they only account for a very small amount of overall traffic (0.00456%). This suggests that while these small files are being accessed frequently, they are not contributing significantly to the overall traffic on the system. On the other hand, files between 1MB to 16 MB (32.6%) make up a significant portion of overall traffic (71.7%), indicating that these files are contributing significantly to the amount of data being transferred on the system.

Request Time

We will begin by tallying the number of requests for each unique CID and group them by this count. From there, we can calculate the size of the files within each group, allowing us to understand the distribution of file sizes for the most requested content.

	mean	std	min	25%	50%	75%	max
request count	26.10	625.84	1	1	1	2	101717

Table 3.4: Statistics on the number of time each CID is requested

For each CID, we count how many times it is requested. By examining the data in 3.4, we can see that a majority of the CIDs are requested by only one user agent. To further understand the request patterns for the CIDs, we look at each range in detail.

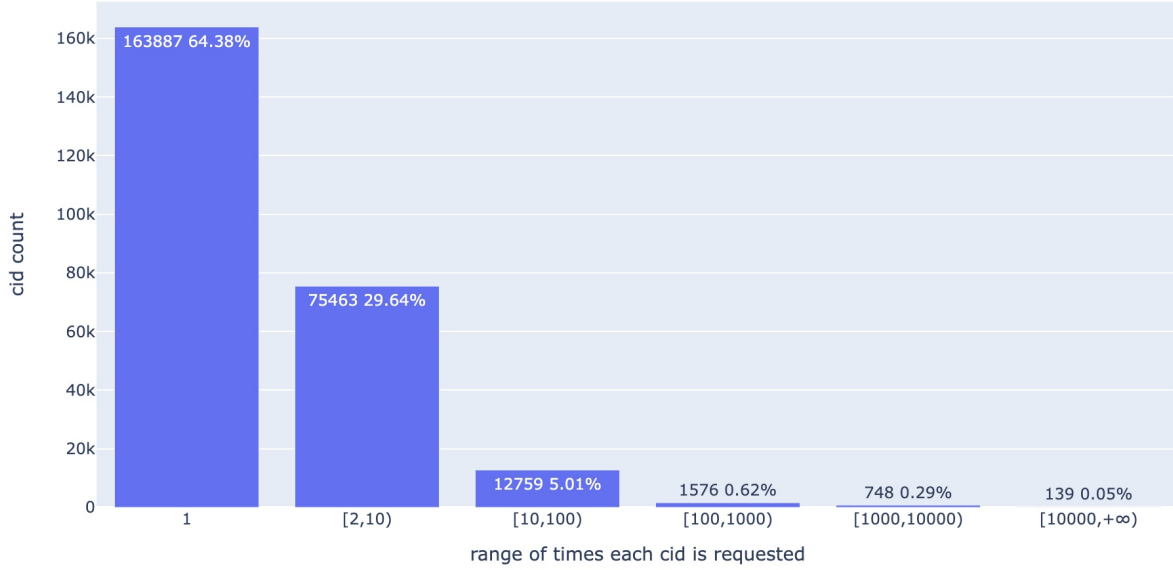


Figure 3.7: CID count in each range of request times

Figure 3.7 shows the distribution of request times for the CIDs in the dataset. It is clear that the majority of CIDs are requested relatively infrequently, with 64.38% of them being requested only once. This suggests that there may be a large number of CIDs that are not highly popular or widely used. On the other hand, a small percentage of CIDs are requested much more frequently, with a small number being requested over 1000 times. These CIDs may represent content in popular online platforms or services.

To understand the relationship between request frequencies and file sizes for the CIDs in the dataset, we grouped the CIDs by the number of times they were requested and calculated the percentiles of their file sizes. The results are shown in table 3.5 and figure 3.8.

number of time cid is requested	percentile file size (MB) for each range				
	min	10%	50%	90%	max
1	0	0	0	1.00	2639.35
[2,10)	0	0	0.11	2.219	1590.45
[10,100)	0	0	0.18	3.09	69.37
[100,1000)	0	0	0.59	1.80	41.17
[1000,10000)	0	0	0.80	1.22	11.55
[10000,+∞)	0	0.37	0.89	1.38	2.39

Table 3.5: Percentile of file size for each request count range

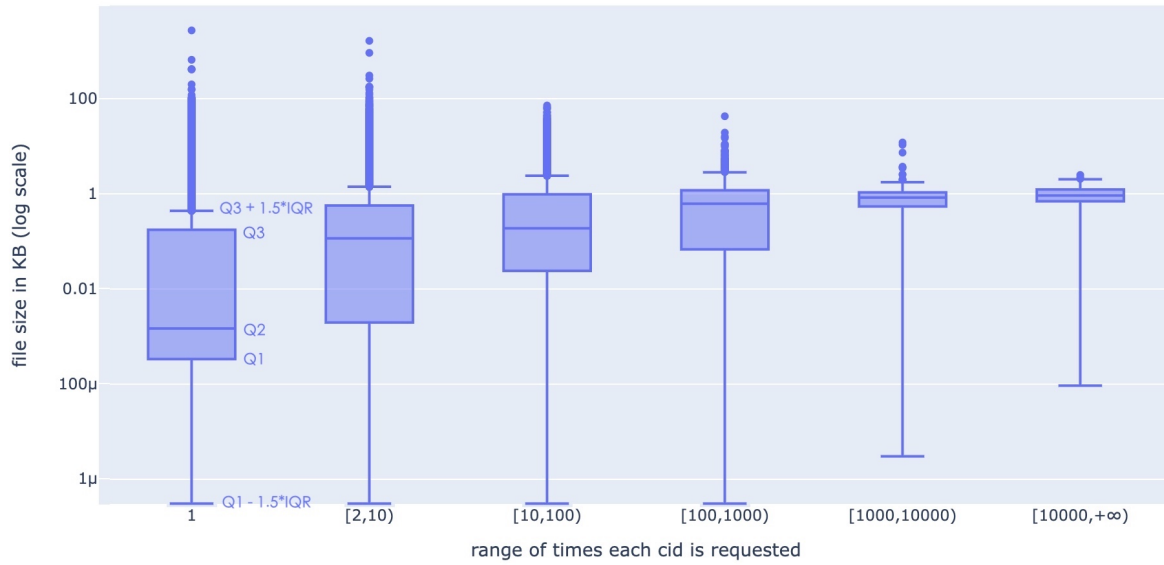


Figure 3.8: File size in each range

In figure 3.8, the ends of the boxes represent the lower and upper quartiles, while the median (second quartile) is marked by a line inside the box. The upper and lower lines, known as fences, are calculated based on the interquartile range (IQR) and represent the values that fall outside the range of the lower and upper quartiles. The upper fence is calculated as $Q3 + 1.5 \cdot IQR$, while the lower fence is calculated as $Q1 - 1.5 \cdot IQR$, where $Q1$ and $Q3$ represent the first and third quartiles, respectively.

From the results, we can see that all of the files that are requested over 10000 times are small, with file sizes under 10MB. In contrast, all of the big files with sizes over 1GB are requested less than 10 times. This suggests that there may be a relationship between request frequency and file size, with smaller files being requested more frequently and larger files being requested less frequently.

Request by Unique Agent

To visually represent the popularity and distribution of the content, we can create a heatmap based on the total number of times each CID is requested and the number of unique users that have made a request for it. This visualization can help us quickly identify which pieces of content are most in demand and how widely they are being accessed by different users.



Figure 3.9: CID count in each range of requests by unique user agents

According to Figure 3.9, the majority (81.21%) of CIDs on the IPFS network are only requested by a single user agent. A small percentage (1.47%) of CIDs are accessed by more than 10 different users. Two particularly popular CIDs are requested by over 1000 users (not shown in the figure). Additionally, the data shows that the number of requests for these CIDs far exceeds the number of unique agents, suggesting that the content is only popular among a limited group of users.

This suggests that the distribution of CID popularity is heavily skewed towards a small number of highly popular CIDs, while the majority of CIDs are only requested by a single user. This could potentially be due to the fact that a small number of files are of wide general interest, while the majority of CIDs represent more specialized content. Additionally, the high ratio of requests to unique agents suggests that even among the more popular CIDs, there is a relatively small number of active users who repeatedly access the content.

3.3.3 Agent Request

It is also useful to conduct analysis to understand the characteristics of the agents who are making these requests. For example, are certain agent groups more likely to request large files or request CIDs more frequently? Understanding the behavior of different agent groups could provide valuable insights into how CIDs are being used and help inform strategies for optimizing their delivery.

Request Time and Traffic

In this part, we aim to understand how different user agents (e.g. web browsers) are accessing the network and the volume of requests they are generating. We will begin by tallying the total number of requests made by each agent and group them accordingly. Then, we will calculate each group's contribution to the total requests and traffic across the network.

mean	std	min	25%	50%	75%	max
312.32	3846.60	1	2	10	94	396655

Table 3.6: Statistics on the number of request each agent made

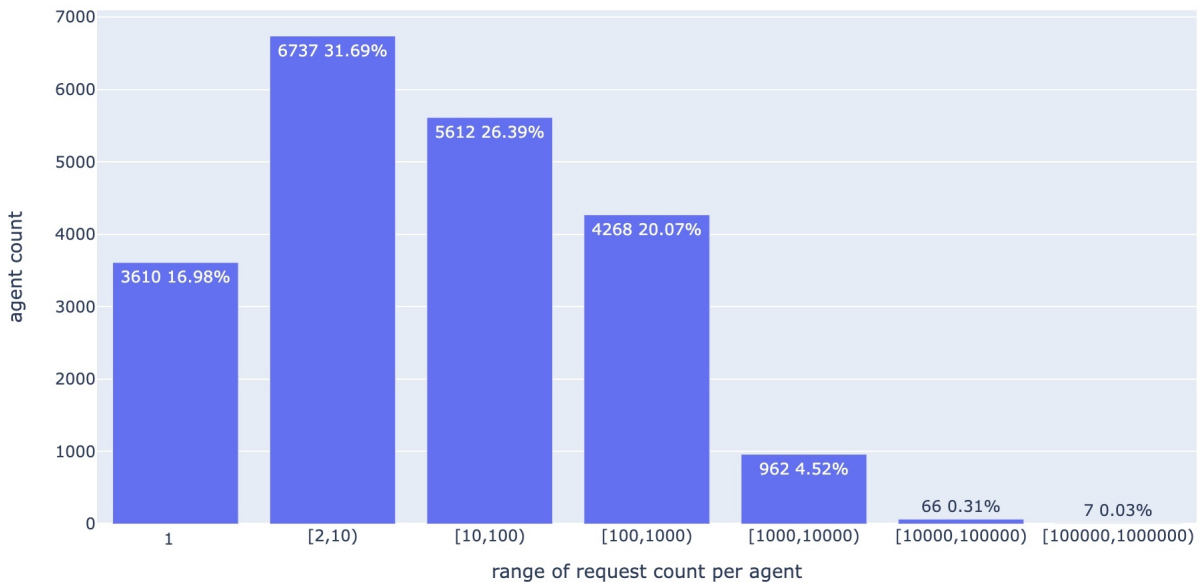


Figure 3.10: Number of agents in each group

We count how many requests each agent made. Based on the data presented in table 3.6 and figure 3.10, it appears that the majority of agents made relatively few requests, with 75.67% making less than 100 requests. On the other hand, a small minority of agents made significantly more requests, with 0.34% making more than 10000 requests. This suggests that there is a significant imbalance in the distribution of requests made by the agents, with a few agents making a disproportionate number of requests compared to the majority.

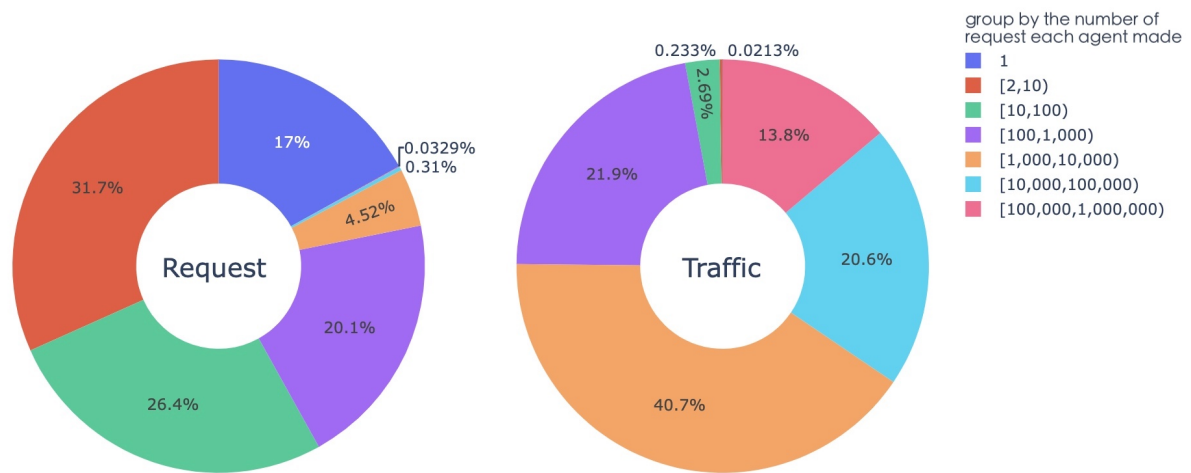


Figure 3.11: Percentage of requests and traffic for each group of agents

For each range of request times, we group the agents and count the total number of requests and traffic. Based on the data presented in figure 3.11, it seems that the agents who made only one request account for a relatively small proportion (0.0213%) of the overall traffic, despite representing a significant portion of the total number of requests (17%). This suggests that these agents may be using the system in a way that is less resource-intensive. On the other hand, the agents who made requests more than 1000 times are responsible for the majority of the overall traffic, indicating that these agents are generating significantly more data.

Cumulative Traffic

To better understand the distribution of traffic among different agents, we can first sort them by the total size of their requests, then tally the cumulative traffic.

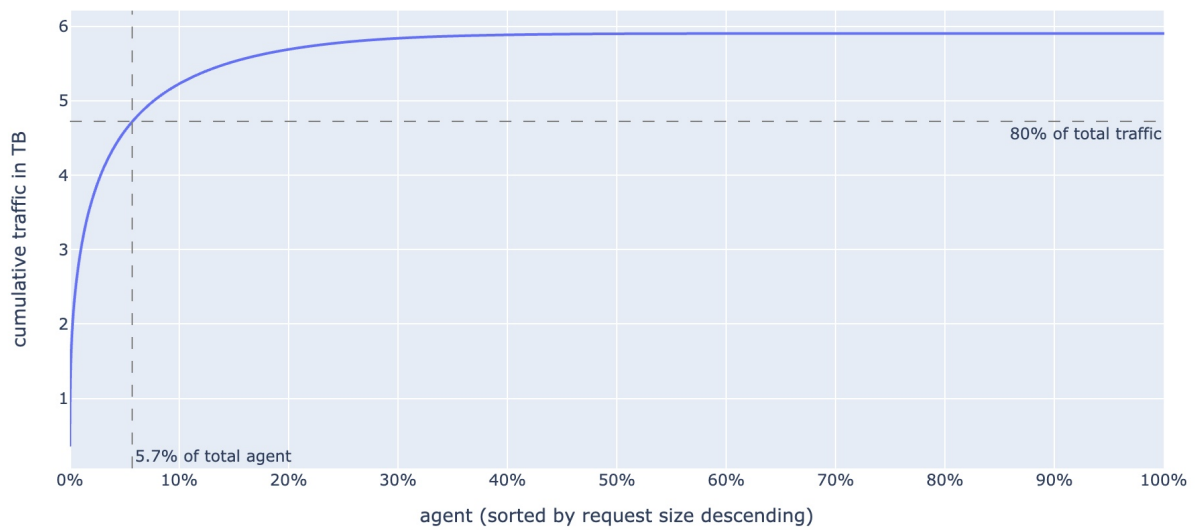


Figure 3.12: Cumulative traffic by agent

Based on the data presented in figure 3.12, it appears that a relatively small proportion of agents are responsible for the majority of the traffic. Specifically, 5.7% of the agents are responsible for 80% of the traffic. This suggests that there is a significant imbalance in the distribution of traffic among the agents, with a few agents generating a disproportionate amount of traffic compared to the majority.

Request Size

In order to provide insights into the relationship between file size and download behavior, we create a line chart of maximum and minimum request sizes for each agent. To create this chart, the data was first filtered to exclude agents with more than 10,000 downloads (0.34% of total agents) and files larger than 16 MB (0.80% of all files). This was done to eliminate extreme outliers that could distort the overall trends in the data. After this filtering, 60.34% of the records were left. In order to more clearly display the trends, the values were rounded to the nearest integer. Finally, the agents were sorted by minimum value and then by maximum value to enable comparison of the request sizes for each agent.

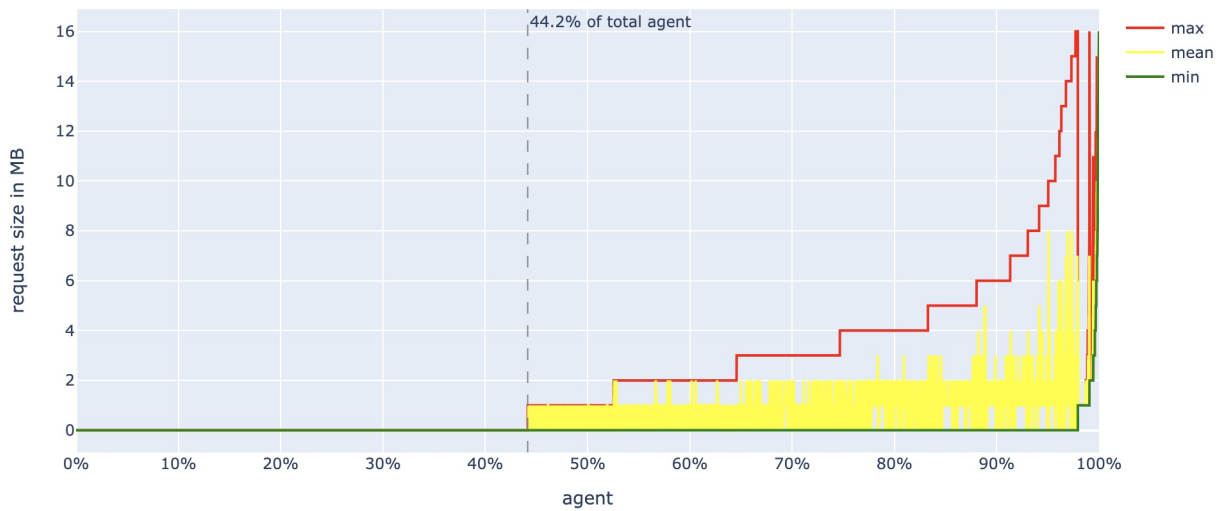


Figure 3.13: Request size by user (outliers removed)

Based on the data presented in figure 3.13, it appears that the majority of agents tend to download relatively small files. Specifically, 44.2% of the agents only download files that are less than 1 MB in size. On the other hand, a very small percentage of agents tend to only download large files. The distribution of file sizes downloaded by the agents is significantly imbalanced, with the majority of agents focusing on small files and a small minority focusing on large files.

3.3.4 Agent Churn Rate

Analysing agent churn rate can provide valuable insights into the characteristics and behavior of different groups of agents, which can be helpful to design incentives that are tailored to the needs and preferences of specific groups, and potentially improve their satisfaction with the system and encourage their continued engagement.

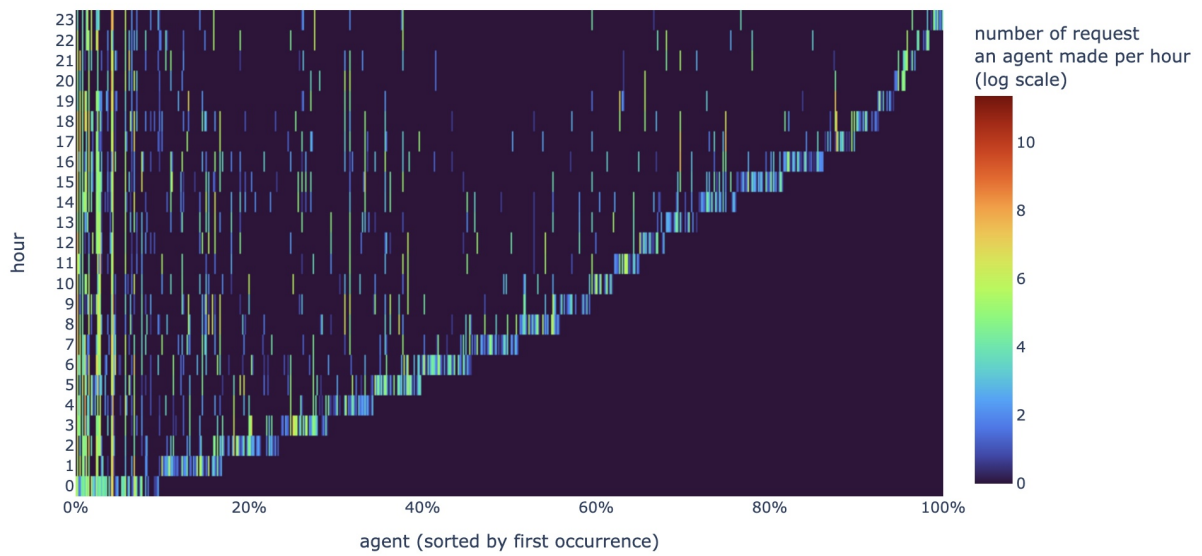


Figure 3.14: User request heatmap

Figure 3.14 is a heatmap of the number of requests made by each agent in each hour. The x axis represents the agents, sorted in the order that they first appeared in the dataset. The y axis represents the hours of the day. The color of each cell in the heatmap indicates the number (in natural logarithmic scale) of requests made by a specific agent during a specific hour. From the heatmap, it is clear that the majority of agents are only active during the first hour after joining the network, while a smaller number of agents are active throughout the day. This suggests that there is a significant variation in the level of activity among the different user agents, with some being much more active than others. The heatmap is denser on the left than on the right, indicating that agents who join the network at the beginning tend to stay active for a longer period of time. However, it is important to note that the dataset only contains measurements from a single day, so it is not possible to know whether the agents were already making requests or will continue to be active beyond the time period captured in the dataset. Next, we discuss the user churn rate in more detail.

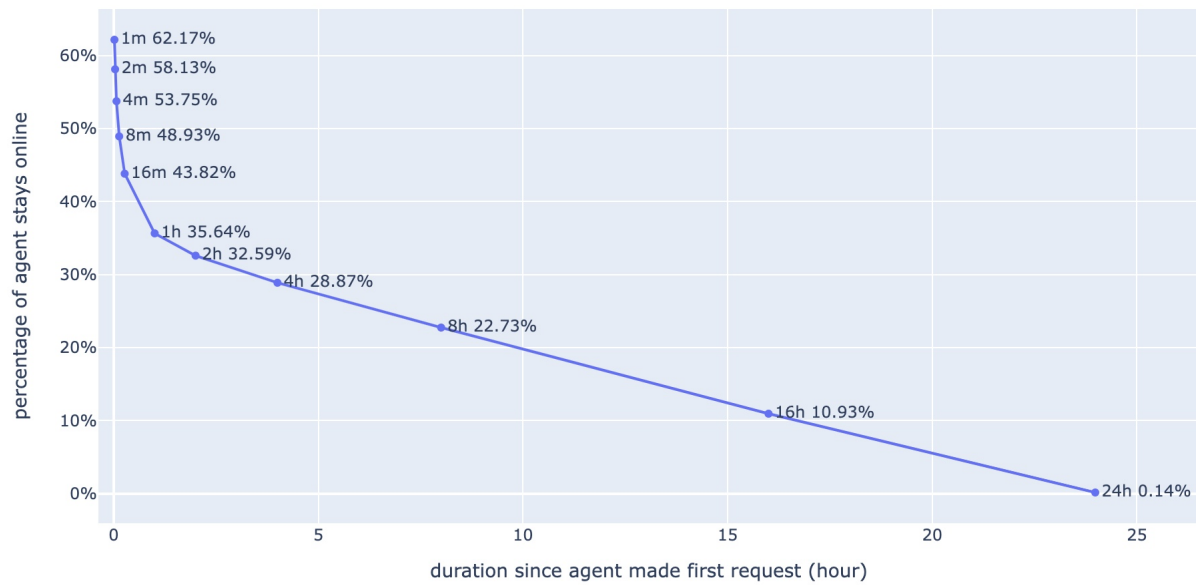


Figure 3.15: User churn rate

Figure 3.15 presents data on the percentage of users who stay in the network after a specific time period. From this figure, we can observe that a relatively large proportion (37.83%) of users disconnect from the network in less than a minute. However, as the time period increases, the percentage of users who disconnect gradually decreases. This suggests that the user churn rate tends to decline over time. It is possible that users who disconnect early in their time on the network are more likely to be casual or temporary users, while those who remain active for longer periods of time are more invested in using the system and are less likely to churn.

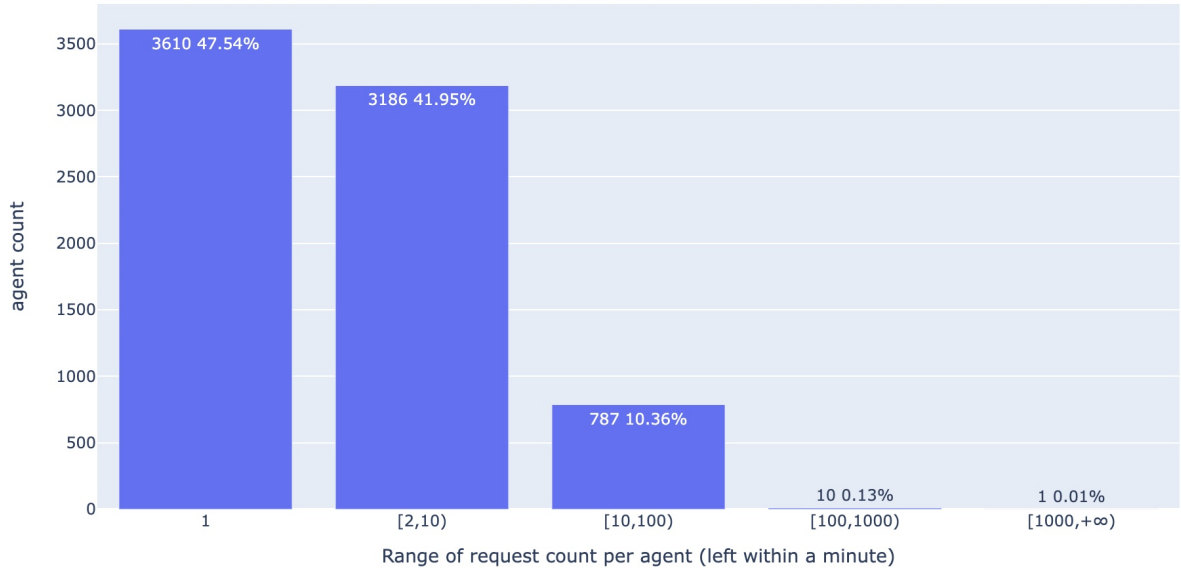


Figure 3.16: Number of request users left within a minute made

Furthermore, as shown in figure 3.16, of those users who disconnect from the network within a minute, a large proportion made only a small number of requests. Specifically, 47.54% of these users have made only one request, and 65.73% have requested only one unique CID. This suggests that the majority of users who disconnect quickly after joining the network are relatively inactive, making only a small number of requests before disconnecting.

3.4 Conclusion

Our analysis of the IPFS gateway dataset revealed a number of interesting findings. We found that IPFS is handling a large volume of requests with relatively stable performance. We also observed a potential relationship between request frequency and file size, with smaller files being requested more frequently, but not significantly contributing to overall traffic on the system. In terms of CID popularity, we found that the distribution is heavily skewed towards a small number of highly popular CIDs, while the majority of CIDs are only requested by a single user. Additionally, we observed a significant imbalance in the distribution of traffic among agents, with a few agents generating a disproportionate amount of traffic compared to the majority. Most agents tend to download small files, while a small minority focuses on large files. Finally, we found that the majority of agents are only active during the first hour after joining the network, while a smaller number are active throughout the day.

Chapter 4

Performance Evaluation of IPFS and Swarm nodes

4.1 Experiment Environment

Our experiments were conducted on two virtual machines with 2 vCPUs, 16GB RAM, and 50GB storage running Debian 10. The IPFS-client version 0.16.0 and Swarm Bee Client version 1.8.2 were used for IPFS and Swarm, respectively. For the virtual machines (nodes), we specified the roles of each node and labeled them as following:

1. Node 0: The IPFS client will be run on this node.
2. Node 1: The Swarm client will be run on this node.

During the experiment, we started both nodes with default settings and a public IP address. The node were idle and not searching for any content or peers.

4.2 Experiment Design

4.2.1 Monitoring Resource Consumption

To understand the resource consumption of a node, we measured the following standard metrics of each client process:

1. CPU: The process's share of the elapsed CPU time since the last screen update, expressed as a percentage of total CPU time.

2. Memory: The process's current share of available physical memory, expressed as a percentage of total memory.
3. Network: Upload and download bandwidth of the process (KB/second).

We use the *top* command to measure CPU and memory usage, and *nethogs* command to measure network traffic. We use the *cron* command to schedule the measurements every 10 seconds and redirect the output to file.

4.2.2 Observing Connected Peers

We use different commands to observe the peers directly connected to the nodes, specifically:

1. for IPFS: *ipfs swarm peers*
2. for Swarm: *curl http://localhost:1635/peers*

Other measurement settings and export methods are the same as measuring resource consumption, only the frequency is once a minute.

4.3 Installation Process

In this section we give a brief summary of the command line installation process of the two networks. Kubo [17] is an IPFS implementation in Go. Bee [18] is a Swarm implementation in Go. They provide command line interface to IPFS and Swarm nodes.

Process to run an IPFS node:

1. Install Kubo.
2. Initialize the repository.
3. Run the ipfs daemon.

Process to run a Swarm node:

1. Install Bee and bee-clef.

2. Configure Bee.
 - (a) Set full-node flag to true.
 - (b) Configure NAT Address.
 - (c) Change swap-initial-deposit value to zero.
 - (d) Change the blockchain RPC endpoints.
3. Fund the node with XDAI.
4. Restart the node.

A Bee node must have stable access to a Gnosis Chain RPC endpoint, so that it can interact with the chequebook contract and postage batches. We use GetBlock [19], a third-party RPC endpoint provider, in this experiment. We also need gas fees to fund the node. The BAO Community xDAI Faucet allows us to obtain a small amount of xDAI for free. Here, we use the official Gnosis faucet [20].

We can see that there are several configurations that need to be made in order to run a Swarm node, which IPFS does not require. Swarm node installation is more complex.

4.4 Measurement Procedure

We run the IPFS and Swarm clients on the two nodes respectively. The nodes were started around 11:50 PM on December 4, 2022, and the measurements began shortly after. As some of the measurements require the pid of the process, they had to be started after the node was up and running, which caused some delay.

During the measurement period, the clients run as background tasks, and we do not perform any other interactions with the nodes. Specifically, we took measurements regarding resource consumption every 10 seconds and calculate the average in each minute. The measurements of connected peer were taken every minute. The measurements continued until 10 AM on December 6. Finally, we extracted and analyzed data from December 5.

After the measurement period, we download the output files from the virtual machines. Then, analyze using Python3.

4.5 Experiment Results

In this section, we will assess the 24-hour monitoring data acquired as described before. We will compare the CPU, memory and network usage of IPFS and Swarm nodes, as well as difference in connected peers.

4.5.1 Resource Consumption

CPU

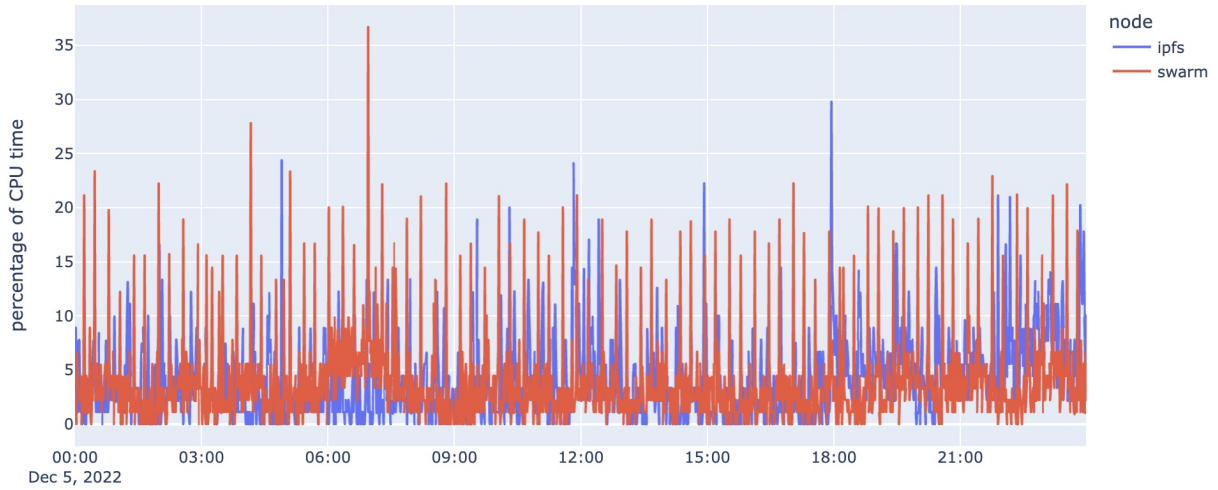


Figure 4.1: CPU usage of IPFS and Swarm clients

Figure 4.1 presents data on the CPU usage of both IPFS and Swarm clients. From this figure, it is clear that the CPU usage of both clients is constantly changing. The IPFS client has a maximum share of CPU usage of 29.78%, while Swarm has a maximum share of 36.68%. The average share of CPU usage for the IPFS client is 3.94%, while the average for Swarm is 3.82%. These data suggest that both clients are able to effectively utilize the available CPU resources, but that the Swarm client tends to have a slightly higher maximum and average share of CPU usage compared to the IPFS client.

Memory

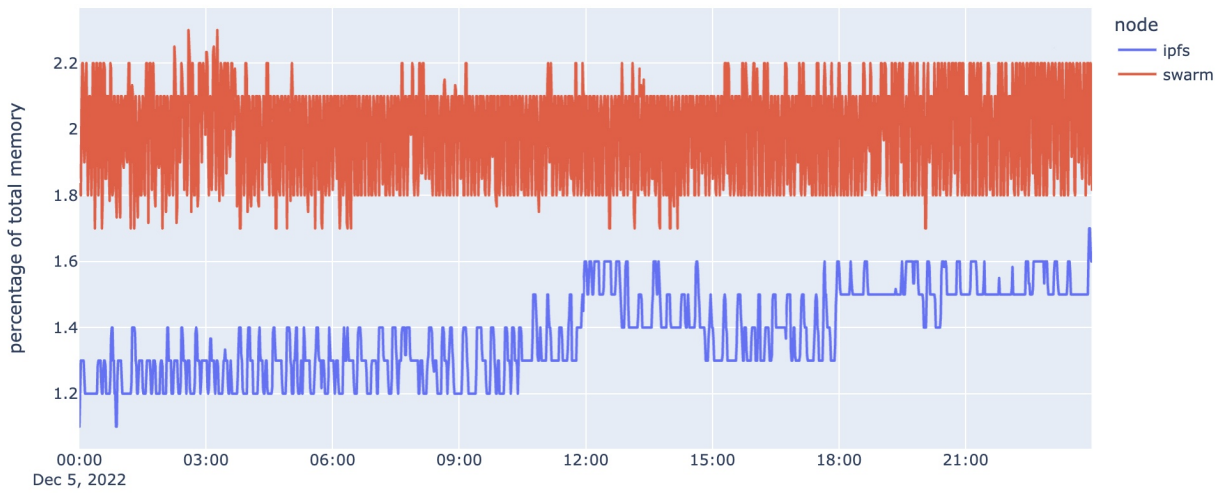


Figure 4.2: Memory usage of IPFS and Swarm clients

Figure 4.2 shows the memory consumption of the IPFS and Swarm clients. The Swarm client has a consistently higher memory usage than the IPFS client, with an average of 1.99% compared to 1.38%. This suggests that the Swarm client tends to require more memory resources than the IPFS client in order to perform its functions.

Network

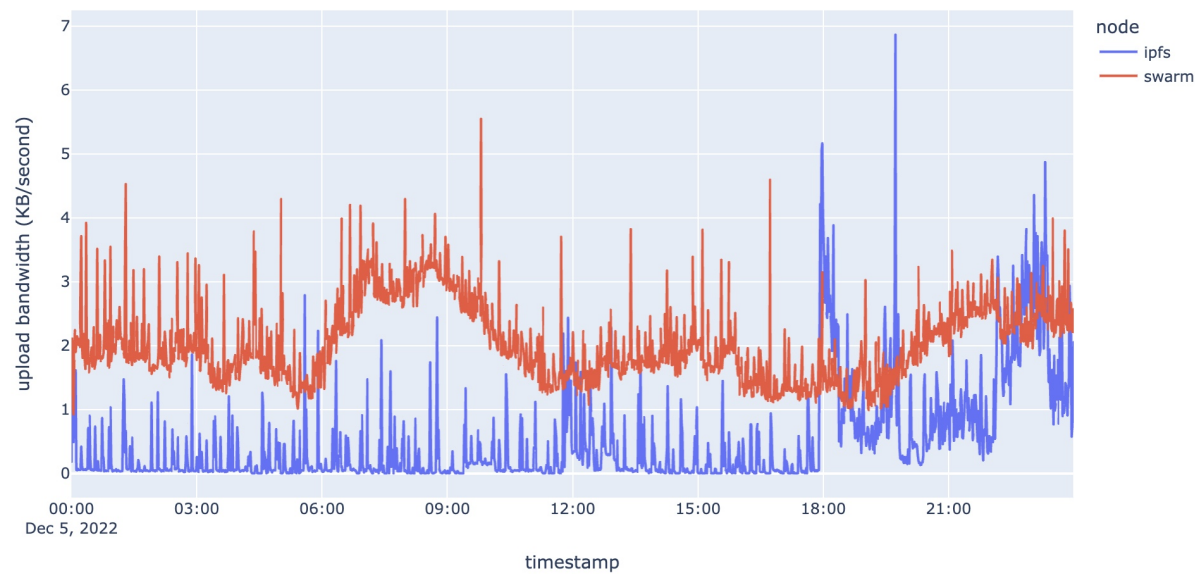


Figure 4.3: Upload bandwidth

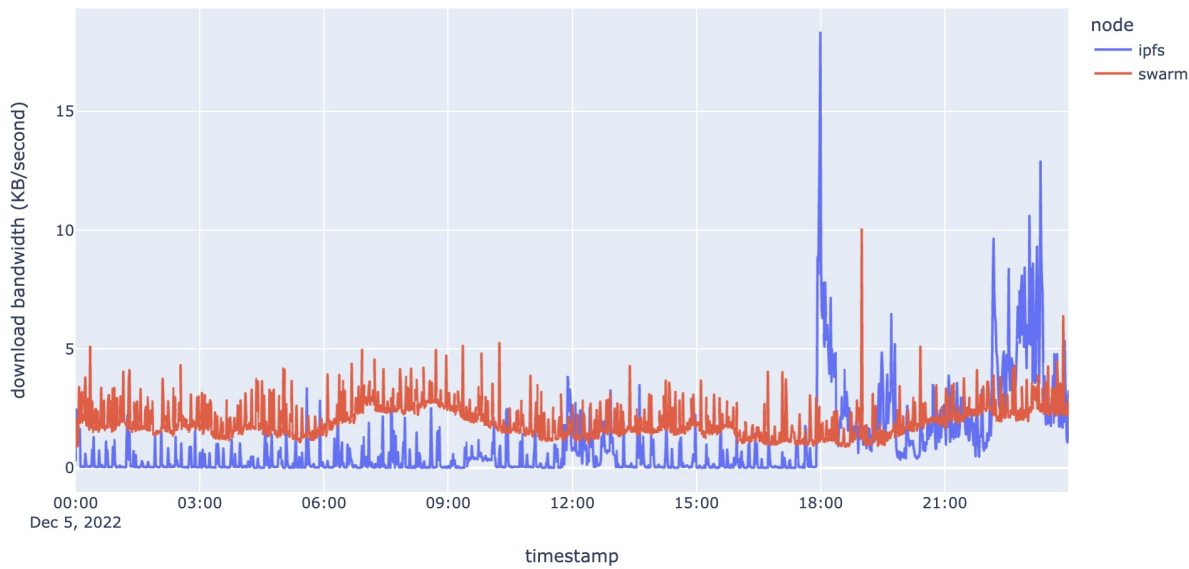


Figure 4.4: Download bandwidth

Figure 4.3 and figure 4.4 show the network consumption of the IPFS and Swarm clients. The nodes were running in the background and were not performing any uploading or downloading operations at the time of measurement. As in the previous experiment, the data was collected by measuring every 10 seconds and taking the average of each minute.

From the data shown in the figures, we can see that Swarm client typically has a larger average upload bandwidth than IPFS, with an average of 2.03 KB/s compared to 0.57 KB/s for IPFS. Similarly, Swarm's average download bandwidth is also higher than that of IPFS, at 1.98 KB/s and 1.10 KB/s, respectively. The Swarm client appears to require more network resources than the IPFS client in order to carry out its functions.

4.5.2 Connected Peers

Same as previous section, we run the IPFS and Swarm clients, and regularly observe their directly connected peers. Data was collected every minute for 24 hours on December 5, 2022.

Number of Connected Peers

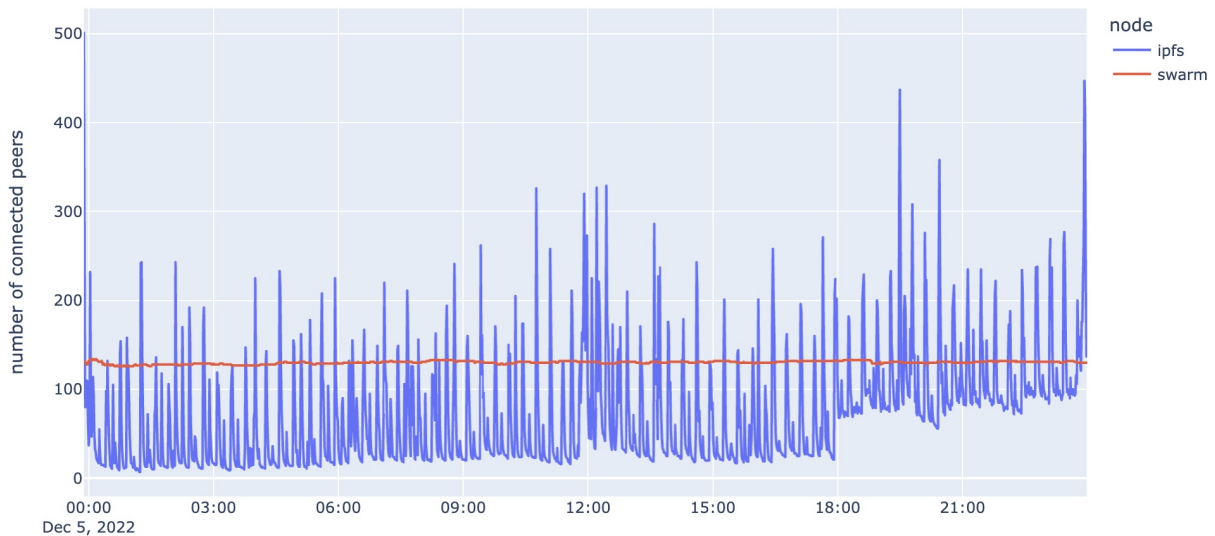


Figure 4.5: Number of connected peers

Figure 4.5 shows the number of peers that the node is directly connected to. We can see that the number of peers connected to the IPFS node fluctuates significantly, while the number of peers connected to the Swarm node is more stable. The average number of connected peers for

the IPFS node is 71.87, compared to 130.24 in Swarm. Next, we will delve further into the analysis of the changes in connected peers.

Peer Availability

We also want to know if the peers are consistently available throughout the day, or if the connected peers frequently change. We create the following graph similar to a heat map, indicating whether a peer is connected or not.

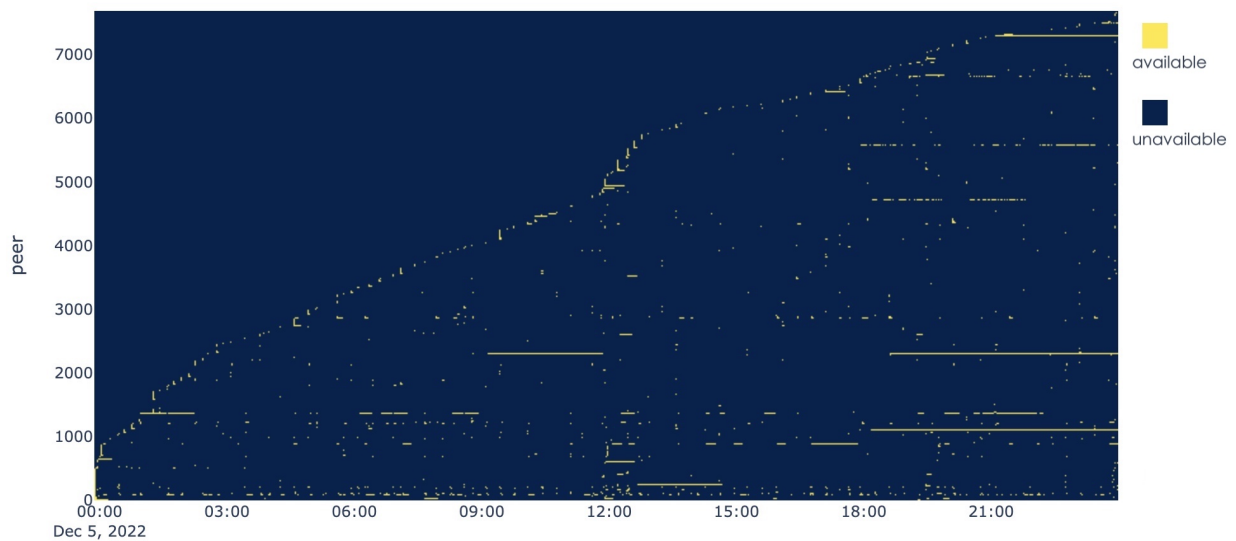


Figure 4.6: Available peers in IPFS

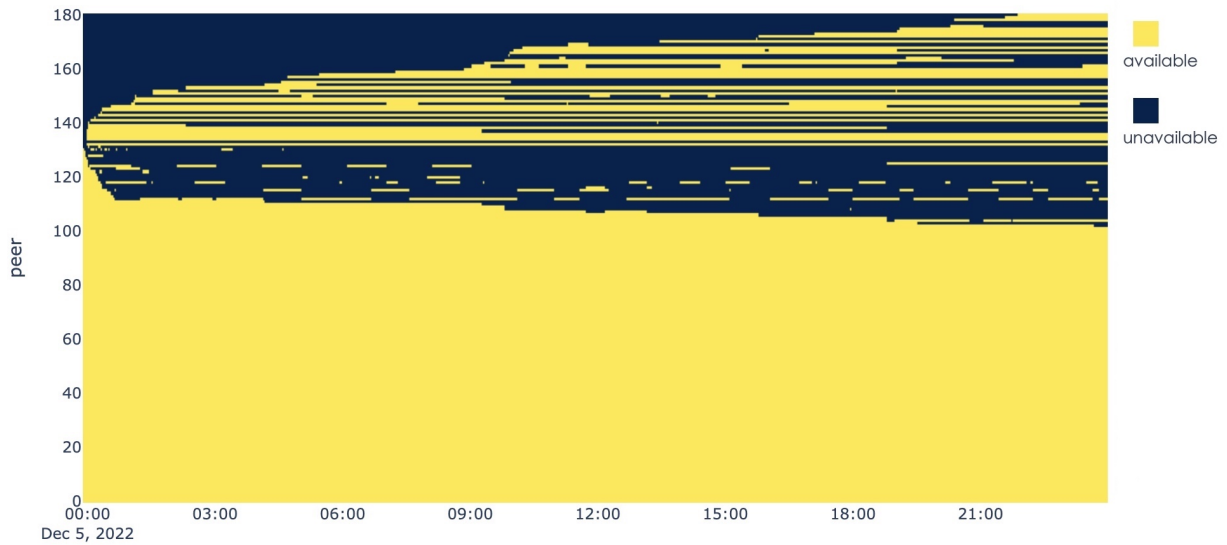


Figure 4.7: Available peers in Swarm

From figure 4.6 and figure 4.7, we can observe that the patterns of connected peers are quite different between the two networks. In the IPFS network, the connections are usually short-lived and the connected peers change constantly, with a total of 7688 unique peers present throughout the day. This suggests that the IPFS network is more dynamic, with peers frequently connecting and disconnecting from the network.

The experimental results are consistent with those of previous studies, which have shown that, typically there are between 15,000 and 20,000 nodes participating in the public DHT at the same time [10]. The nodes refresh their Kademlia buckets every 10 minutes, and due to the high level of churn in the IPFS network, it is likely that the peers in these buckets will change frequently.

In contrast, in the Swarm network, the peers typically maintain connections for a longer period of time, with only 181 different peers present throughout the day. This suggests that the connections in the Swarm network is more stable, with fewer peers joining and leaving the network.

This difference in connectivity patterns between the two networks could be due to a variety of factors. One reason may be related to the different peer discovery strategies of the two networks. As mentioned in section 2.2.2, IPFS establishes connections with all the nodes it encounters, while Swarm initially establishes connections with a hard-coded set of bootstrap nodes. Another reason may be related to the use cases of the two networks. As mentioned in section 4.3, IPFS is simple to use and does not require tokens to join the network, so many users may simply leave after completing the download. On the other hand, using Swarm requires gas fees to fund the node, so users may tend to stay online for a longer period of time to balance expenses.

4.6 Conclusion

Our analysis found several differences between the IPFS and Swarm clients. Swarm requires additional configurations, and tends to use slightly more CPU, memory and network resources than IPFS. In terms of connectivity, IPFS has a higher average number of connected peers, but these connections are usually short-lived and the connected peers change constantly. While in the Swarm network, peers tend to maintain connections for a longer period of time.

Chapter 5

Limitation and Future Work

There are several limitations to this project that should be noted. First, the dataset used for visualization and analysis is limited in one day, and may not be representative of the entire IPFS network. This could lead to biased or incomplete conclusions. Second, the experiments conducted to evaluate the performance of the IPFS and Swarm clients may not be fully representative of real-world usage scenarios in different environments or under different conditions. Third, the analysis and conclusions drawn in this project are based on the current state of the IPFS and Swarm networks. It is possible that the characteristics and usage patterns of these networks may change over time, which could invalidate some of the findings presented in this project.

There are several possible improvements and areas for future work that could be considered based on the findings of this project. One possible improvement would be to repeat the performance experiments multiple times in order to increase the reliability of the results. Another could be to explore the difference between user behavior of these decentralized networks and HTTP, social media or microservices. Also, investigate how the system can be tuned or modified to improve certain parameters. Overall, there is much potential for further research and optimization of decentralized networks in order to make them more reliable and effective for all users.

Bibliography

- [1] Juan Benet. “Ipfs-content addressed, versioned, p2p file system”. In: *arXiv preprint arXiv:1407.3561* (2014).
- [2] Viktor Trón. *The Book of Swarm, V1.0*. URL: <https://www.ethswarm.org/The-Book-of-Swarm.pdf>.
- [3] Gavin Wood et al. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.
- [4] Petar Maymounkov and David Mazières. “Kademlia: A peer-to-peer information system based on the xor metric”. In: *International Workshop on Peer-to-Peer Systems*. Springer. 2002, pp. 53–65.
- [5] Ingmar Baumgart and Sebastian Mies. “S/kademlia: A practicable approach towards secure key-based routing”. In: *2007 International conference on parallel and distributed systems*. IEEE. 2007, pp. 1–8.
- [6] Erik Daniel and Florian Tschorsch. “IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks”. In: *IEEE Communications Surveys & Tutorials* 24.1 (2022), pp. 31–52.
- [7] Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. “Mapping the interplanetary filesystem”. In: *2020 IFIP Networking Conference (Networking)*. IEEE. 2020, pp. 289–297.
- [8] Seoung Kyun Kim, Zane Ma, Siddharth Murali, Joshua Mason, Andrew Miller, and Michael Bailey. “Measuring ethereum network peers”. In: *Proceedings of the Internet Measurement Conference 2018*. 2018, pp. 91–104.
- [9] Viktor Trón, Aron Fischer, Dániel A Nagy, Zsolt Felföldi, and Nick Johnson. “Swap, swear and swindle: incentive system for swarm”. In: *Ethereum Orange Paper* (2016).
- [10] Dennis Trautwein, Aravindh Raman, Gareth Tyson, Ignacio Castro, Will Scott, Moritz Schubotz, Bela Gipp, and Yiannis Psaras. “Design and evaluation of IPFS: a storage layer for the decentralized web”. In: *Proceedings of the ACM SIGCOMM 2022 Conference*. 2022, pp. 739–752.
- [11] Jiajie Shen, Yi Li, Yangfan Zhou, and Xin Wang. “Understanding I/O performance of IPFS storage: a client’s perspective”. In: *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*. IEEE. 2019, pp. 1–10.

- [12] Periklis Kostamis, Andreas Sendros, and Pavlos Efraimidis. “Exploring Ethereum’s Data Stores: A Cost and Performance Comparison”. In: *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE. 2021, pp. 53–60.
- [13] Omar Abdullah Lajam and Tarek Ahmed Helmy. “Performance evaluation of ipfs in private networks”. In: *2021 4th International Conference on Data Storage and Data Engineering*. 2021, pp. 77–84.
- [14] Leonhard Balduf, Sebastian Henningsen, Martin Florian, Sebastian Rust, and Björn Scheuermann. “Monitoring data requests in decentralized data storage systems: A case study of IPFS”. In: *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2022, pp. 658–668.
- [15] Erik Daniel and Florian Tschorsch. “Passively Measuring IPFS Churn and Network Size”. In: *arXiv preprint arXiv:2205.14927* (2022).
- [16] Vahid Heidaripour Lakhani, Leander Jehl, Rinke Hendriksen, and Vero Estrada-Galinanes. “Fair Incentivization of Bandwidth Sharing in Decentralized Storage Networks”. In: *2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops (ICD-CSW)*. IEEE. 2022, pp. 39–44.
- [17] *ipfs/kubo*. URL: <https://github.com/ipfs/kubo>.
- [18] *ethersphere/bee*. URL: <https://github.com/ethersphere/bee>.
- [19] *Getblock: Blockchain nodes provider - crypto node as a service*. URL: <https://getblock.io/>.
- [20] *Gnosis chain xDAI faucet*. URL: <https://gnosisfaucet.com/>.