CAPITAL©

Research-Focused Blockchain Investment Fund

IPFS Analysis

9/28/2018

zk Capital Research

IPFS: The Interplanetary File system

Analysis Summary

In this report, we analyze the Interplanetary File System (IPFS). IPFS is a novel hypermedia transfer technology that was introduced in 2014. We expect it to play a significant role in tomorrow's distributed world. Before we delve into this report, our goal is list the reasons that excite us about IPFS. We also pinpoint some areas of concern that would require improvement in order for IPFS to excel.

1. **IPFS is a significant step up from the predecessor technologies**
   IPFS has benefited significantly from multiple predecessor technologies like distributed hash tables (DHT), Bittorrent, git, and SFS. IPFS has built upon these technologies to provide an enhanced solution for hypermedia data sharing. In addition, IPFS is an open source project that accepts world wide research and development contribution to enhance the system. The Go-IPFS repository is one of the top hundred most starred Go repositories on Github.

2. **IPFS is an important component of the Web 3.0 infrastructure**
   Web 3.0 is a long term target that aims to replace the current internet infrastructure (Web 2.0). As decentralization is the essence of Web 3.0, many consider the distributed ledger technology (DLT), for example blockchains, as the core building block of Web 3.0. A blockchain is an immutable and append-only ledger that stores the network state. Distributed consensus between all the network nodes is required in order to extend the blockchain and store the critical network data among the network nodes. Therefore, it could be prohibitively expensive to store any other kinds of data into the blockchain. For multiple use cases, it may be more efficient to store other non-critical data in a secure fashion close to the security level of the blockchain. IPFS is the a very suitable storage medium for this category of data. IPFS allows for distributed storage of data that is immune to altering and forgery. Data stored on the IPFS network cannot be altered without changing the data identifier. In IPFS, the identifier is a cryptographic hash of the data. This way, non-critical data can be stored to IPFS while storing the data cryptographic hash to the underlying distributed ledger instead of saving all the data to the ledger.

3. **IPFS is an optimal storage platform for decentralized apps**
   Decentralized applications (dApps) are a class of apps that leverage decentralization to achieve unprecedented benefits. Among those are decentralized exchanges and decentralized marketplaces where centralized intermediaries are removed, hence eliminating/reducing any trading fees. Another example is decentralized social media and video platforms where content cannot be censored at the will of the operating company. Such dApps require the storage of significant amount of data. IPFS allows this data to be stored in a distributed way that is censorship-resistant. For these reasons IPFS is turning into a preferred storage platform for dApps.

4. **IPFS can provide better user experience**
   IPFS could lead to improved user experience in multiple cases. For example, trying to browse or download some popular content using the typical client-server models may exhaust the network bandwidth and lead to network congestions. This may result in an inconvenient user experience due to the larger latencies. In IPFS, content is delivered from the closest peers that possess a copy of the content removing the single-node pressure and improving the user experience. In addition, IPFS allows for continuous and smooth browsing of the content even if the owner of the content is no longer available. For example, you can browse the goods sold by your favorite online store or the videos created by your favorite youtuber, even if these nodes are offline for maintenance or any other reason.

5. **IPFS allows for new online business models**
   In today's internet, any online business that publishes content has to host this content on some dedicated servers. It is also essential for such a business to ensure the availability of their content and a sufficient bandwidth to satisfy the required demand. IPFS fundamentally changes this model. In IPFS, rather than having a single host server serving all users, data is shared in a distributed manner and can be served by any node in possession of the data. As a result, the bandwidth requirements are significantly reduced and the reliability is enhanced. That being said, new business models will begin to evolve. For example, using some projects like Filecoin it would be possible for content publishers to pay users a small reward to store the content. This would improve content distribution and ensure content availability.

6. **IPFS is receiving increased mainstream adoption**
   Due to the multiple attractive features of IPFS, it is getting increased mainstream adoption. In the report we discuss situations where IPFS helped mainstream users fight censorship in Turkey and Spain. In addition with the recent Cloudflare's announcement, it is now possible to host websites on IPFS and point to them using an easy standard domain name. IPFS hosted websites are censorship-resistant websites and now they can be easily recognized and securely browsed using HTTP and HTTPS and using the Cloudfare's IPFS gateway. In addition, Cloudflare has implemented techniques to guarantee that users do not need to trust Cloudflare to be serving them the right content they asked for.

IPFS is a new technology undergoing continuous development. Yet, we recognize a number of a challenges that need to be overcome to achieve widespread adoption. . These challenges include

1. **Bandwidth requirements**
   Running an IPFS node currently involves using significant bandwidth that may not be feasible to many users especially in developing countries. Excessive bandwidth usage may harm the adoption of the IPFS in multiple areas of the world. While there are multiple

suggestions on how to handle this problem, financial incentives may be the proper direction. Gaining financial rewards for hosting content on IPFS can help cover the costs of running nodes and encourage adoption.

2. **Availability**
   The current implementation of IPFS cannot guarantee data availability upon being requested. The only way to ensure availability is to constantly save copies of the published content on an IPFS node which is known as “content pinning”. In addition, this IPFS node needs to be online at all times to satisfy the availability guarantees. Projects like Filecoin build an incentive layer to encourage nodes to save content which would help solving the availability problem.

3. **Private content**
   Content published to IPFS is public by design. Anyone in possession of the a content hash can access such content. Currently, IPFS does not provide a built-in solution for storing private data. However, encryption can be used to save or transfer private data over IPFS. The other possible way, which is slightly complicated, is to create a private network using the IPFS protocol where the nodes can only connect to a specified list on nodes that form the private network.

Introduction

Currently, all of our existing computing devices interact with one another by staying connected to some sort of network. These networks connect with one another making it possible for all devices to communicate and share data. However, these networks differ in capabilities depending on the technology they are built upon. Some may be very powerful and have low latencies when responding to requests while others could be less powerful and require higher latencies. That being said, the way data is being shared across networks could become a bottleneck as the size and distance traveled by the data increases.

Network Topologies

We can generally categorize networks into three main topologies as shown in Figure 1.

Centralized networks generally represent star topologies where all the nodes of a network are connected to a common central node. In order for any node to communicate and/or share data with other nodes, it must initially communicate with the central node which then facilitates the communication. Clearly, such networks present threats to the network as the central node could be a single point of failure. If for any reason the central node fails, all the other nodes become disconnected from one another and communication and data sharing is no longer possible.

Decentralized networks aim at solving the single point of failure issue by distributing the central power to more than one node. Rather than having a single central node, the network consists of multiple star topologies in which the central node of each group of nodes is connected to at least one other central node. By that, if a central node fails for any reason, only that portion of the network becomes disconnected while the other nodes may still communicate and share data. Therefore, power is distributed among multiple central nodes rather than relying completely on a single node. Although distributed networks are more reliable than centralized ones, they may still pose some threats. Decentralized networks still require a certain degree of trust. For example, consider how Google provides a decentralized network of servers which can serve all its users. Although performance and reliability is definitely improved over centralized models, in such networks we must trust that those service providers are acting honestly and will efficiently facilitate data sharing as we desire.

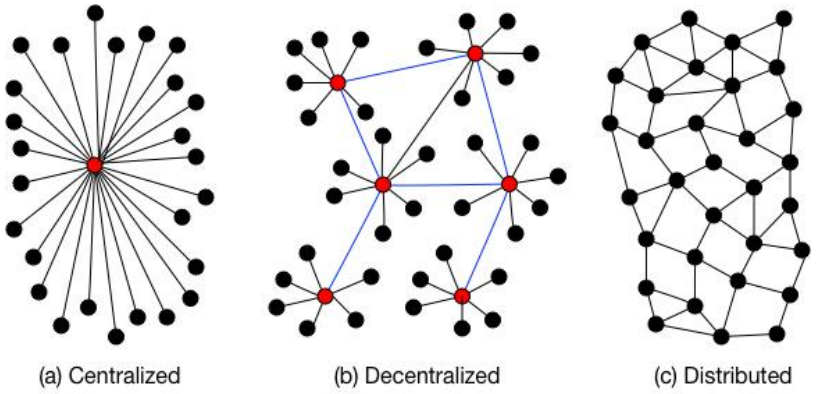

Figure 1. General view of network topologies.

Distributed (aka peer-to-peer) networks eliminate the concept of central nodes. In such networks, all nodes are equivalent in terms of *power*; the equal ability to provide information to the network and read content from it. As long as a node is connected to at least one other node, it remains connected to the entire network. The more connections a node maintains, the more reliable its communication and data sharing with the entire network becomes. In such networks, nodes communicate and share data with their peer nodes. As a result, distributed networks improve performance and reliability over decentralized networks. A good example of a distributed network is BitTorrent[1]. In this network, nodes connect in a peer-to-peer manner to share large data files. Any node within the network may serve data to a requesting node if it possesses the data required.

The Problem

Today, we rely mainly on centralized and/or decentralized networks. Within these networks, nodes usually communicate with one another via client-server models such as HyperText Transfer Protocol (HTTP)[2]. In such models, a client (for example a web browser) requests some files from a server (for example a computer hosting the files), as illustrated in Figure 2.

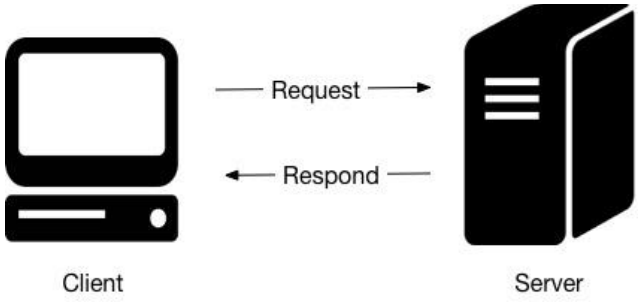

Figure 2. The general client-server model.

[1] https://www.bittorrent.com
[2] https://www.rfc-editor.org/rfc/pdfrfc/rfc2616.txt.pdf

The success and dominance of such protocols has been fulfilling to many users till this very day, however, the success seems to be threatened by the evolution of data. Data is being generated in continuously growing sizes, requiring enhanced methods of storage and efficient access in order to satisfy the same acceptable user experience. With this rapid growth, the capacity of the Internet infrastructure is quickly being outpaced and may collapse[3]. To further elaborate on these concerns, consider the example shown in Fig 3-(a).

1. An uploader uploads a file (represented in blue) to a centralized host server such as Google Drive.
2. Now, imagine that a very large number of users want to fetch this file. From their computers, each user will request the file (represented in red) from the hosting server.
3. As the number of users requesting the file increases, the network will become more congested and the hosting server will become slower in responding. Larger files will also require more time to be served and will consume more bandwidth of the network. This is the phenomenon that DDoS attacks take advantage of.

Some might argue that caching exists to solve these problems by keeping copies of these files close to the requesting users. However, even caching has proven to be not so efficient in all cases[4]. It even contradicts the concepts of security and privacy of the users since it may compromise the data being transferred or may even leak information about the users[5]. Moreover, we must not forget that this entire system is centralized. If our files are not well protected and the host server is not guaranteed to always be online, the requesting users might not even be able to access the files when they need them. Therefore, in order to continue to provide users with an efficient method to communicate and share data, novel distributed sharing protocols have begun to evolve.

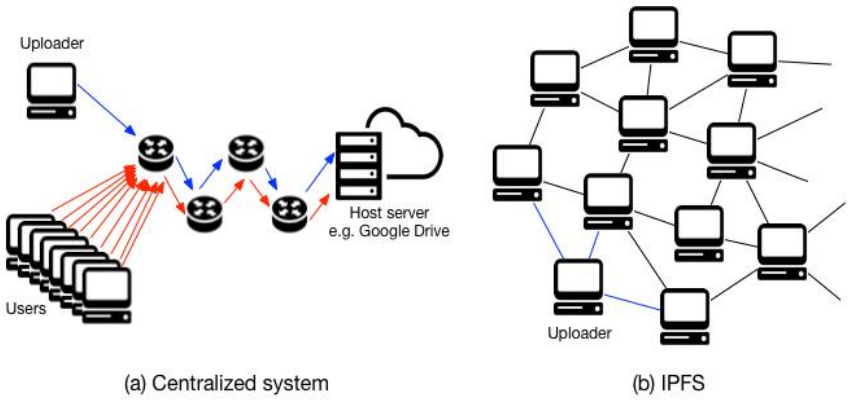

Figure 3. Sharing data in a centralized system vs. IPFS.

[3] R. Fielding, and R. Taylor. *Architectural styles and the design of network-based software architectures*. Vol. 7. Doctoral dissertation: University of California, Irvine, 2000.

[4] L. Dowdy and D. Foster. *Comparative models of the file assignment problem.* ACM Computing Surveys (CSUR) 14.2 (1982): 287-313.

[5] A. Sengupta, R. Tandon, and T. Clancy. *Fundamental limits of caching with secure delivery.* IEEE Transactions on Information Forensics and Security 10.2 (2015): 355-370.

InterPlanetary File System (IPFS)

Building on these concepts, IPFS[6] was introduced in 2014 by Juan Benet as a content addressed, versioned, P2P File System. IPFS leverages a distributed network to provide an efficient means of transmitting data over large distances, even if it was interplanetary transmission (data traveling between planets), hence the name, *InterPlanetary File System (IPFS)* . It could be thought of as a peer-to-peer distributed file system that seeks to connect all computing devices into a single system of files. With that target kept in mind, IPFS is also referred to as the *distributed* web, the *permanent* web, or the *merkle* web as it is based on the merkle graphs. IPFS was first implemented in Go. Since then, IPFS development continued as an open-source project that is spearheaded by Protocol Labs. As a result of these development efforts other implementations of IPFS in JavaScript and Python were introduced or in progress. IPFS aims to replace HTTP and provide a more open and decentralized web.

Therefore, IPFS aims at handling such concerns by building a protocol that runs over a distributed network. Figure 3-(b) illustrates how the same scenario can be handled. Assume we have an uploader as shown in the figure. In IPFS, when a user uploads data to the network, the data is initially segmented and stored locally on the user's machine. The entire network is then notified that the data are stored and located on the user's machine. If any node within the network wishes to obtain the data, it will reach out to the user's node, download the data, and keep the data in its cache. At that point, the data becomes accessible at multiple nodes. As we proceed through this report, we will describe how this process is accomplished.

Technical Design

In this section, we aim at briefly introducing the main technical concepts that form IPFS. We first introduce the building blocks that the system builds upon. Next, we discuss how these components are used to develop the different layers of the IPFS protocol.

Building Blocks

IPFS is an inspiration of previously existing peer-to-peer systems. We first begin by simplifying the major concepts behind each of these systems.

Distributed Hash Table (DHT): A DHT is a distributed system that provides a lookup service of stored values. The general form of a lookup service consists of a *<key,value>* pair. *Values* that are stored correspond to particular *keys*. Given a unique *key*, the DHT returns its corresponding stored value. Intuitively, the name implies that this service is distributed among several nodes. In

[6] J. Benet, *IPFS-content addressed, versioned, P2P file system, arXiv preprint arXiv:1407.3561* (2014).

IPFS, such a service can be used to locate the nodes that store certain data files being requested within the network. Given the key (the actual data we are looking for), the DHT will return the value (the addresses of the nodes storing the data we are looking for). IPFS expands on already implemented DHTs such as Kademlia DHT, Coral DSHT, and S/Kademlia DHT because of the benefits they provide. Such benefits include efficient and secure lookup through very large networks to reduce latency and counter attacks, for example, Sybil attacks.

BitTorrent: BitTorrent is a successful and widely implemented protocol used to share large data files in a distributed manner. It is best known for digital video sharing such as TV shows, movies, video clips, ...etc. The protocol partitions such large data files into segments that are distributed over different nodes of a peer-to-peer network. When a user requests a certain file, the online storing nodes begin to share the partitions they store. As a result, data is being shared from more than one source and we are not exhausting a single server. In BitTorrent's exchange protocol, nodes are incentivized to act honestly by being rewarded while those being non-cooperative are punished. The connection of those punished nodes becomes choked; they remain connected to the network but cannot download any data from other peers until they become unchoked. To become unchoked, the non-cooperative node must strongly cooperate in other requests. In addition to this, the exchange protocol prioritizes sending data that is less available in the online nodes over those that are widely distributed, to guarantee all files are delivered upon being requested. The exchange protocol in IPFS is inspired by that used in BitTorrent and is referred to as **BitSwap**[7].

git[8]**:** When multiple authors are working remotely on a single document it is hard to efficiently update each author about the changes made by every single author. This becomes more exhausting as the number of authors and the size of the file increase. As a solution, git provides a distributed revision control system that tracks changes made to data files authored by multiple parties. It is mainly used for source code management in order for each individual to recognize the new amendments made before making any other changes. It is known to be efficient in terms of speed, data integrity (tamper-resistant) and non-linear workflows. Given those advantages, IPFS also makes use of git's content-addressable way of storage. Data is not stored as is, in fact, it is first cryptographically hashed and the resulting hash is used to find data within the network. The hash relates to the *key* needed to search for in the DHT in order to find the corresponding *values* (nodes that store it). However, as mentioned earlier, data files are partitioned into smaller segments. Therefore to keep an organized storage system, git links all segments into a Merkle Directed Acyclic Graph (DAG). The Merkle DAG is similar to the merkle tree data structure. It uses the cryptographic hash of the content itself to produce pointers to the data. As a result, the links (pointers) between files become *immutable*.

---

[7] D. Dias, J. Johnson, and J. Benet, *BitSwap*, https://github.com/ipfs/specs/tree/master/bitswap.

[8] S. Chacon and B. Straub. *Pro git*. Apress, 2014.

Self-Certified File System (SFS)[9]**:** SFS is a distributed file system that separates key management to grant access to files from the actual files being shared over the internet. As a result, the files can be shared with everyone over the web and each group of users can manage their own keys the way they desire. This approach facilitates a secure network file system that avoids internal key management. In contrast to other systems, SFS does not require a key management system to map file names to encryption keys. In fact, in SFS, file names incorporate public keys, hence, they are self-certifying pathnames. To simplify this, the SFS file pathname allows a user to certify the server it is fetching the data from. IPFS makes use of such technology by implemented it in its naming system (IPNS). Using SFS allows users to generate verifiable addresses that can be verified. That being said, data users may have more confidence when accessing data over IPFS.

IPFS Protocol Layers

Now, since we have a good understanding of the major building blocks that form IPFS, we can delve into the IPFS sub-protocols and learn how they expand on the previously discussed systems.

Node Identities: Inspired by S/Kadmelia, IPFS nodes must generate identities before being able to communicate with other nodes. Using a Public Key Infrastructure (PKI), the node first generates a public key and private key. Next, the node cryptographically hashes the public key to derive the its identity. However, similar to Proof-of-Work (PoW) algorithm, this process is repeated several times until the derived identity contains a certain number of zeros in its prefix. The number of zeros identifies the difficulty of generating a new identity[10]. That being said, the process of generating a new identity becomes expensive. This is necessary to mitigate several types of attacks such as Sybil attacks where a single user can generate too many identities. Having multiple identities can be used later to perform multiple kinds of network attacks[11]. In the Security section of this report, we delve deeper in the details about such concerns.

Network: The protocol allows nodes to establish connections with other nodes regardless of the type of network they are connected to. In contrast to common practice in contemporary networks, IPFS does not require nodes to possess IP addresses and is designed to run on top of any network. In order to achieve this, it uses customized addresses in which each node defines the details of the protocols it runs and its addresses corresponding to those protocols.

---

[9] D. Mazières, *Self-certifying file system*. Diss. MIT, 2000.

[10] This current design is still not implemented in IPFS, however, accounted for in their roadmap.

[11] https://medium.com/zkcapital/beginners-guide-on-blockchain-security-attacks-part-1-network-ca4e74435723

Routing: When data is requested by a node within the network, the routing protocol locates the addresses of nodes and the data they store in order to find the data. This protocol uses a DHT inspired by S/Kademlia, Coral and Mainline. Objects that are 1 KB or smaller in size are placed directly on the DHT. With objects that are larger, the node identities of the nodes that store a copy of the objects are stored on the DHT.

Exchange: Once the requested data is located, IPFS uses its BitSwap exchange protocol for nodes to exchange the data they store. Similar to BitTorrent, IPFS uses its own BitSwap credit system that keeps track of how data was previously shared between peers. Nodes within the system can learn how their peer nodes are behaving in terms of how much data have they shared and how much they have requested. This history is stored in a BitSwap ledger that is exchanged between the nodes.

Objects: IPFS uses an IPFS object data structure format to store data files. Before being stored, files are partitioned and cryptographically hashed. The partitions are linked to one another to facilitate fetching the entire file. Links are maintained using a Merkle Directed Acyclic Graph (DAG).

Files: Inspired by git, IPFS can model a versioned file system on top of the Merkle DAG.

Naming: IPFS uses a naming system known as InterPlanetary Naming System (IPNS). IPNS builds on SFS allowing mutable naming to immutable data content maintained by the Merkle DAG.

As shown in Figure 4, we can now construct the entire IPFS protocol into six main layers. The figure maps each distributed systems to a layer in the IPFS protocol. The major components of IPFS are the *IPNS*, *IPLD* and *libp2p* which were built as inspirations of their corresponding distributed systems. *IPLD* is a data format for merkle-linked objects. Its main purpose is to facilitate hash linked data structures and propagate them around the network. On the other hand, *libp2p* is a modular peer-to-peer networking stack used to establish connections between network nodes, search for data and deliver it when requested. It basically represents the exchange, routing and network layers within the IPFS protocol layers. Finally, as previously discussed, IPNS is a component that allows users to always point their references links to the latest data versions stored on IPFS. This allows a user experience similar to that achievable by HTTP. Users can keep using the same link even when the referenced content is updated.

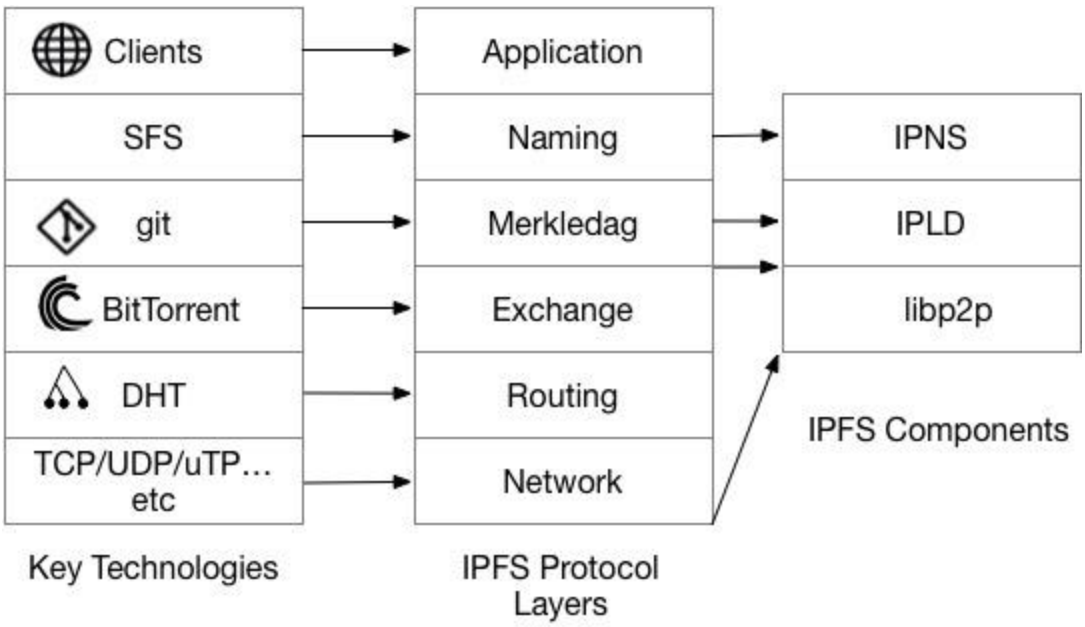

Figure 4. Influential technologies that the IPFS protocols draw from.

P2P Node Analysis

In this section, we perform an analysis on the P2P layer of IPFS to provide some insight about how IPFS nodes operate. We monitor and analyze the dynamic behavior of establishing connections between nodes. In addition, we discuss the bandwidth costs that an IPFS node incurs to support the IPFS network and the possible areas of user experience improvement.

For this analysis, we develop a script to monitor the behavior of the peer nodes that connect to our experimental nodes existing in the US-central region. We track the IP address, geolocation and latency in 1 minute intervals of these nodes to analyze the dynamic connection behavior. In addition, we monitor the bandwidth usage of an experimental IPFS node. Bandwidth monitoring is used as a guide to understand the costs incurred by users and companies when they operate IPFS nodes.

We analyzed how the number of connected IPFS nodes change over time. For example, Figure 5 shows the number of nodes connected to an experimental IPFS node (referred to as node #1) located in Chicago, Illinois for a period of time around 20 hours. The main observation here is the large number of connected IPFS nodes. The number of nodes were consistent between 600 and 900 nodes almost at all times. A similar behavior was observed using a second experimental node (node #2) in Lansing, Michigan. The number of connections of this node for a longer period of time (~ 4 days) is shown in Figure 6. The number of nodes peak to around 900 nodes but during periods of low activity the number of connections drop to about 350.

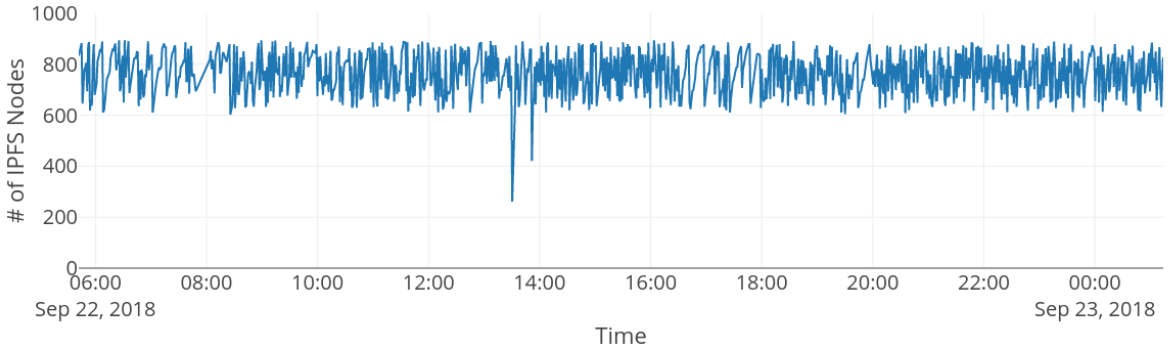

Figure 5. Change of the number of connected IPFS nodes over time for experimental node #1

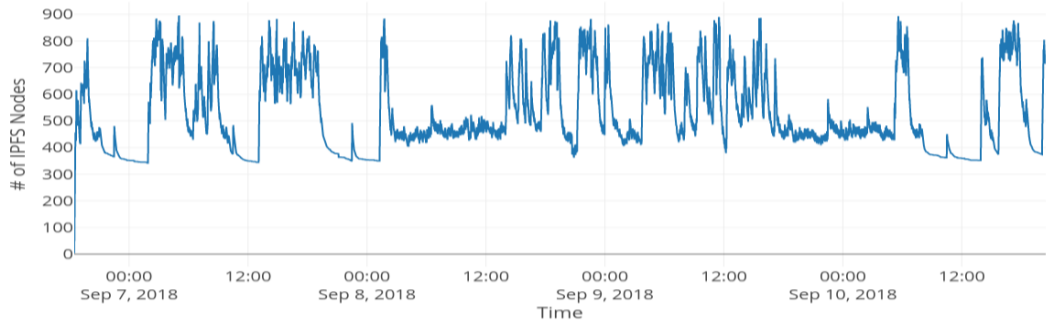

Figure 6. The number of connected IPFS nodes over time for experimental node #2.

To further understand the dynamics of IPFS node connections, we analyzed the length of time other nodes were connected to our experimental nodes. For this analysis, the IP address of each connected node was used as the node identifier. For each node, we calculated the total number of minutes the node was connected to our experimental node #1. The histogram in Figure 7 summarizes this data. The histogram shows the total number of unique nodes that remained connected for a specific range of time. For example, the first bar of the histogram shows that about 750 nodes remained connected to our node for less than 30 minutes. Only a few nodes (15 nodes) were continuously connected to our experimental nodes for the whole experiment time. This indicates a highly dynamic IPFS network where the connected peers continuously change, i.e. enhanced security.

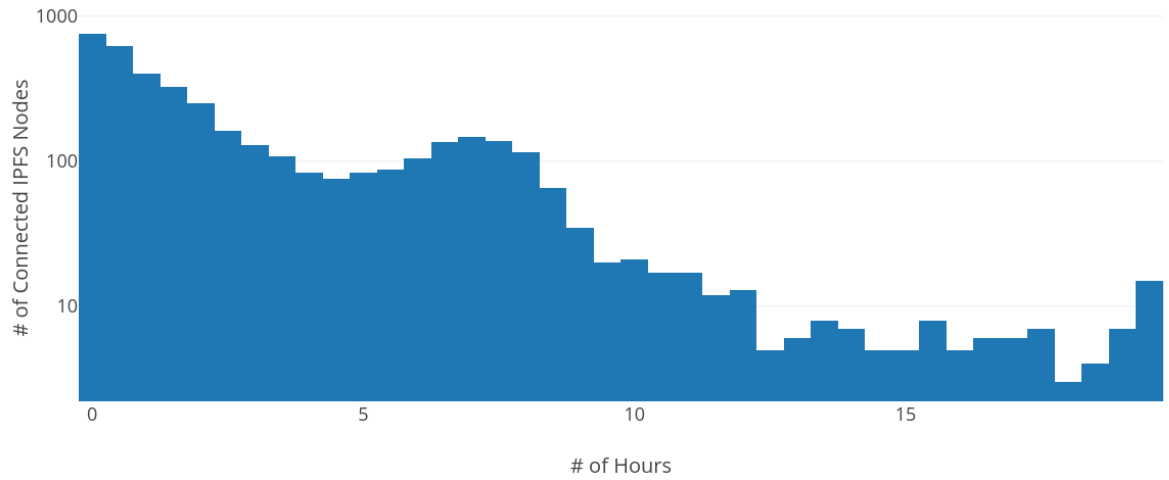

Figure 7. A histogram showing the the number of IPFS nodes connected versus the length of the connection time. Each bar represents a 30-minute range. Y-axis is using log scale[12].

Peer nodes distribution

After analyzing our peer node connections, we extracted the IP addresses of each peer node that established a connection with our nodes. We used those addresses to generate a node geographical distribution as shown in Figure 8. The figure shows that a single IPFS node interacts with other IPFS nodes from every part of the world. IPFS nodes exist in every continent demonstrating the global nature of the technology. However, the figure shows that nodes appear to be concentrated in certain areas like Europe, the US and Southeast Asia. This is reflected by the density and size of the circles represented on the map. A continent like Africa has fewer number of nodes which indicates adoption challenges in developing countries.

[12] An interactive plot can be seen here: https://plot.ly/~fouda/1

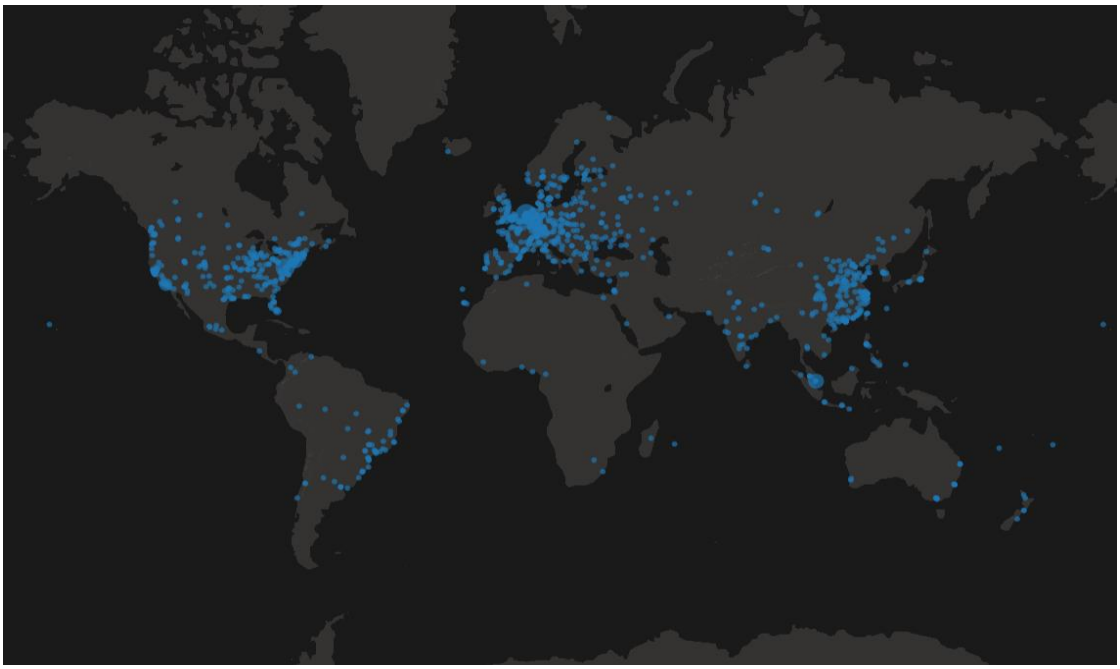

Figure 8. Node distribution of peer nodes that connected to our experimental nodes.

Data traffic and Bandwidth

The large number of connected peer nodes could indicate significant IPFS data traffic. To verify that, a third-party tool[13] was used to monitor the bandwidth of our IPFS node #1. The node was only set to run IPFS with zero upload/download interactions. Figure 9 shows the recorded download and upload rates (in mbps) for 8 hours of the node running. The rates were averaged over a period of 1 minute. The overall average download rate was approximately 1 mbps while the average upload rate was 0.7 mbps. This totals to about 3.3 GB of downloaded data and 2.3 GB of uploaded data for only 8 hours. For the same period of time, the IPFS client reported only 2.3 GB total downloaded data and 1.4 GB total uploaded data. This is indeed an aggressive use of bandwidth for a node that is not used to serve content and not requesting any IPFS content[14].

In our experiments, we purposely chose not to download any data to make sure our nodes do not possess any data requested by other nodes. Therefore, all the data activity produced by our nodes was essentially the DHT data being updated all the time and communicated by the neighboring peers. That being said, a user running an IPFS node will end up paying a bandwidth usage cost whether actively interacting with the network or remaining idle.

[13] http://www.bwmonitor.com/

[14] We have performed multiple IPFS bandwidth tests. We only report a single test for clarity. The excessive IPFS bandwidth usage is a consistent phenomenon.

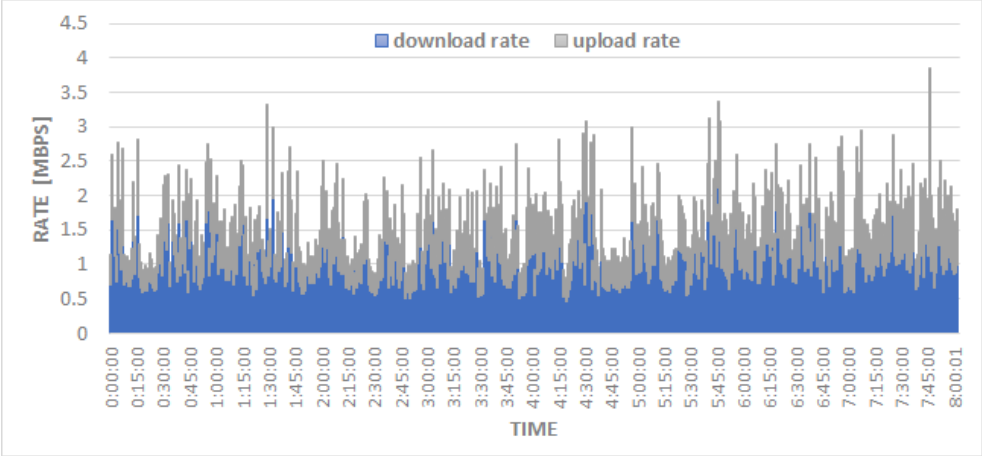

Figure 9. Bandwidth of an IPFS node. Rate is averaged every minute.

In fact, IPFS users have reported excessive use of bandwidth by their IPFS nodes[15] which seems to be an ongoing problem that needs to be solved. It should be noted that in our tests we have used the most recent Go-IPFS client which has already implemented several bandwidth utilization improvements over earlier version. There are still open discussions on how to improve IPFS routing and reduce the bandwidth, CPU and memory utilization of IPFS nodes[16].

[15] https://github.com/ipfs/go-ipfs/issues/3429
[16] https://github.com/ipfs/notes/issues/162

IPFS Challenges

In addition to the aggressive bandwidth usage discussed earlier, IPFS by design is facing a number of challenges. In this section, we discuss these challenges briefly to highlight how they could affect IPFS mainstream adoption.

Availability

Although IPFS provides a means for distributed storage, it alone, cannot provide any guarantees that data which the users upload to the network remains permanently available. In IPFS, the nodes that connect to the network have no incentive to store copies of other people's data. As a result, data may disappear from the network at some point or become unavailable if the nodes that store it are not online. Users may handle this concern in one of two ways.

1. The first and simplest method is to *pin* data to nodes on the network. The term pinning basically means uploading the data onto a node that will continue to remain online and telling that node to keep the data. By that, the data will always be available when requested by any other node. However, if for any reason that node fails and/or there are no other nodes providing the data, this data will no longer be available.
2. Another solution would be for the data owners to pay other network nodes to store the data for them. Examples such as Filecoin are being developed to provide such services to the users. In a future report, we will delve into the concerns that appear with such systems and discuss countermeasures.

Replication

Initially, when a user adds data to an IPFS node, the data is segmented, each segment is cryptographically hashed, the resulting hashes are linked, and finally stored locally on the machine where the node is running. The DHTs stored over the network nodes are then updated to map these segments to the user's node. When any other node requests the data, the requesting node searches the DHT to locate the address of the node storing the data segments. Once located, the requesting node or its peers can download the data from the storing node and store a copy in their cache memory. The DHTs are then updated to include the addresses of the other nodes storing the data segments. At that point, the data have been replicated over the nodes that store a copy. Each time data is requested by a new node, the data replicates more and the DHTs are updated to incorporate these nodes. Nodes continue to store a copy in their caches as long as they can afford to. If at any point a node runs out of cache memory, it replaces older pieces of data segments that are less important with the newer ones. As a result, certain files are replicated more than others across the network based on how frequent they are being shared between the nodes. In contrast, files that are not shared frequently may not exist at many nodes, hence, may not always be available.

Security

Any system that runs over a P2P network is liable to network attacks, for example, Sybil and DDoS attacks. The main purpose of such attacks is to disrupt the service when uploading/downloading data from the network. IPFS incorporates several protocols to limit such attacks.

a) IPFS uses S/Kademlia DHTs. The purpose behind using S/Kademlia is twofold. First, S/Kademlia helps protect against Sybil attacks[17] by making the process of generating nodes expensive. An attacker would want to generate many nodes to control a big portion of the network, hence, potentially control the flow of data. Recall, any node that wishes to join the network must first generate a node identity. To make this an infeasible process for attackers, S/Kademlia incorporates an expensive Proof-of-Work (PoW) cryptopuzzle before being able to generate the identity. Second, S/Kademlia helps protect against DDoS attacks[7] as it allows nodes in the network to search for data in a diverse range of DHTs. That being said, the attempt to perform a DDoS attack becomes infeasible. An attacker would not have the advantage of being able to discover which DHTs a node will lookup the values it wants to request, hence, attacking the nodes hosting them in the network becomes impracticable.
b) Since IPFS runs over a P2P network, users may always be liable to connecting and interacting with malicious nodes. For example, malicious nodes may share incorrect data when it is requested by a certain user. IPFS uses SFS to secure the data traffic as it propagates over the network. In SFS, users are able to certify the other nodes they are obtaining their data from, hence, avoid interacting with malicious nodes.
c) Another issue that arises when sharing data over a P2P network is the refusal of nodes to participate when they possess certain data required by others. Such nodes are known as leechers. IPFS protects against such malicious behavior by using its BitSwap exchange protocol. When the BitSwap credit of a node becomes too low, it becomes *choked;* unable to download data from the network until it begins to participate and help others.
d) Currently, IPFS does not provide a method to restrict particular nodes or even the number of established connections when joining the network. This may result in certain network attacks, for example Eclipse attacks[7], where a resourceful attacker is able to monopolize a user's inbound and outbound connection. Therefore, a user's node may be liable to being disconnect from the entire network and not uploading/downloading data as desired.

[17] https://medium.com/zkcapital/beginners-guide-on-blockchain-security-attacks-part-1-network-ca4e74435723

Privacy

As of today, IPFS does not provide privacy protocols to protect the privacy of a user's data being stored on the network. Users can encrypt the data before adding them to IPFS. Recall, before data is stored on the network, it is initially segmented and cryptographically hashed. An attacker that is able to obtain the cryptographic hash of certain segments will be able to obtain these certain segments and any other segments linked to them. However, IPFS give its users flexibility in terms of how data is segmented and the cryptographic hash functions used. By default, IPFS segments a data file into partitions of 256 KBs and uses SHA-256 hash function. A user is allowed to change these variables as desired, hence, making the process for the attacker to discover the resulting cryptographically hashed segments more difficult.

Applications

In this section, we discuss the different applications of IPFS. We begin by a succinct discussion of the different ways to access content published using IPFS. We refer to multiple useful tutorials for users who wish to run a node and support the network. In addition, we refer to multiple projects that use IPFS to perform user services. Next, we discuss popular incidents where IPFS has played a significant role in fighting censorship. We conclude this section by discussing various decentralized applications that use IPFS as the storage layer.

How to connect to and use IPFS?

There are multiple ways for a user to connect to the IPFS network. The two most common approaches are:

1. Launch an IPFS node and become part of the IPFS network.
2. Interact with IPFS through a "Gateway".

The first approach is more advantageous to the IPFS network, as it expands the network by adding more nodes. The second approach is more user-friendly but it does not help improve the IPFS network.

Launching an IPFS node

To launch your own IPFS node, the standard technique is to download and install the Go IPFS software[18] in order to join the network. Once the installation is complete, a user can upload/download any content from the network. This content can be in any file format. There are many online tutorials that demonstrate how to use *Go-IPFS*. For new users, we recommend Juan Bennet's IPFS alpha demo. The tutorial explains the steps required to install, initialize, and run an IPFS node. It also discusses how to upload/download content from the network using the content hash. We also recommend this article for those who prefer written material.

The main barrier here is that *Go-IPFS* is a command line terminal tool which is not user-friendly for the general public. For that reason, there are ongoing efforts to make using IPFS easier, e.g., building ipfs-desktop, a desktop application with OS-native installers, ipfs-companion, a browser extension that runs an IPFS node inside Chrome or Firefox, and an in-browser file manager for IPFS like IPFS Drive. These efforts are coordinated through the IPFS GUI Working Group. There are also dozens of IPFS-based apps and tools listed which may even install IPFS for you.

[18] Go IPFS is the most popular implementation. Another popular implementation is js-ipfs which is a JavaScript implementation : https://github.com/ipfs/js-ipfs

Using a Gateway to access IPFS

A public IPFS gateway is simply a website that provides an interface to an IPFS node. Users around the world can connect to gateway website through HTTP to access the IPFS network. All that a user needs to access a file in the IPFS network is the file hash. For example, IPFS hosts a copy of the English Wikipedia. The content hash for the English wikipedia on IPFS is *QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco*. Simply going to one of the public IPFS gateways such as ipfs.io and adding "/ipfs/content_hash" at the end of the web address will retrieve the content and display it in the internet browser. There are multiple public IPFS gateways that can be used to access files on IPFS. The list of active gateways can be checked on this website.

Recently, Cloudflare, one of the most well-known internet security companies, introduced their own user friendly IPFS gateway that does not require any installations by its users. Cloudflare also introduced a service that allows users to host their websites on IPFS while having a standard domain name. More importantly, Cloudflare is issuing security certificates for the IPFS-hosted websites. This is a significant development as IPFS websites can now be browsed with the same security level of HTTPS.

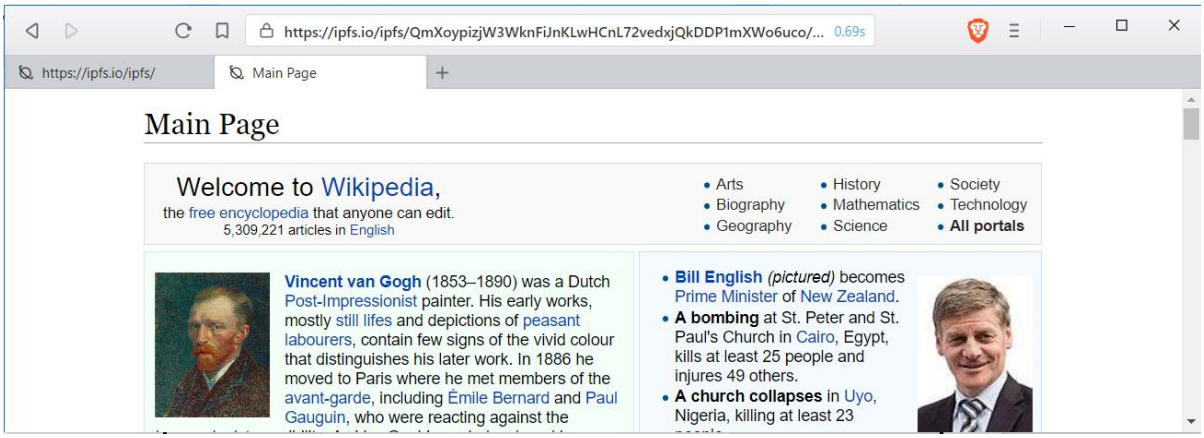

Figure 10. Accessing the English Wikipedia on IPFS using the content hash in the address bar of the ipfs.io public gateway

In addition to the public gateways, other applications strive to make IPFS as easy as possible for specific use-cases. For example, there are efforts to build apps that use IPFS for photo storage and sharing, for example, ipfs.pics[19] and Textile Photos[20] . Other apps strive to make it easy to publish articles that cannot be censored like the IPFessay app[21] which can be easily launched

[19] https://github.com/ipfspics/ipfspics-server
[20] https://www.textile.photos/
[21] https://gitlab.com/stavros/IPFessay

using a web interface[22] or from your own IPFS node. General file transfer using IPFS can be performed without diving into the technical details using a service like FileNation[23].
Such apps that provide an easy route to use IPFS are great but they suffer a common problem. Most of these apps do not generate any profit and depend on community donations to continue development and to cover running costs. These costs include the expenses to run nodes to support the IPFS network. When donations are not enough, the development of such projects stalls and these services are interrupted.

Prominent IPFS use cases

In spite of the inherent challenges present in IPFS, it has proven to be very successful in some use cases. IPFS played a crucial role in challenging censorship in countries like Turkey and Spain.
In Turkey, a court order allowed the Turkish government to block access to Wikipedia. This can be easily done in the current web structure by forcing internet service providers (ISPs) to block the IP addresses of the servers that host Wikipedia. However, this technique would not work if Wikipedia was hosted on IPFS. IPFS is a distributed network meaning that you can access content hosted on IPFS (like Wikipedia) by connecting to different random peers (different IP addresses) that are not related to Wikipedia (the blocked original IP address). Using these features of IPFS, on May 4, 2017, a few days after the Turkish government blocked access to Wikipedia, a snapshot of the Turkish, English and Kurdish Wikipedia were added to IPFS network[24]. This action has effectively made Wikipedia uncensorable and demonstrated the censorship-resistance features of IPFS. Since that date, interest in using IPFS in Turkey has been steadily growing. Currently, the interest in IPFS because of this incident in Turkey has exceeded that from the USA as evident in Google Trends activity.

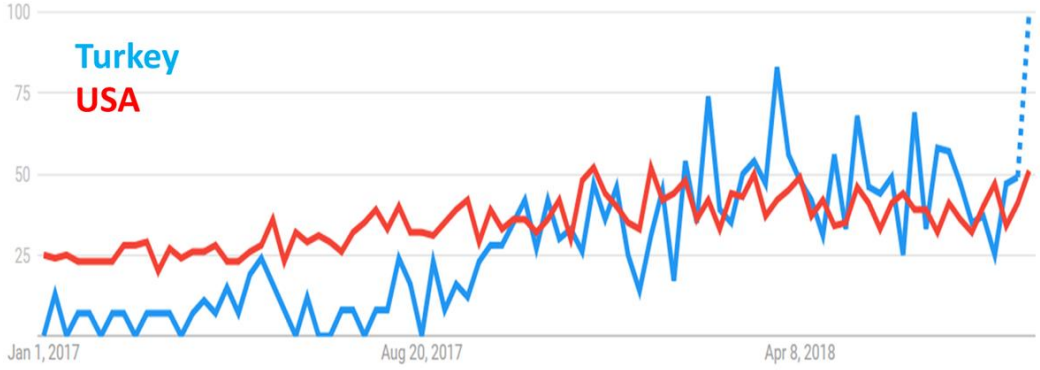

Figure 11. Interest in IPFS in Turkey compared to that in the US (Google Trends)

[22] https://www.eternum.io/ipfs/QmRWeczoWjVoZSY4cvTAp6YaGJSwYJSWvANHXaHiNVd8to/
[23] https://filenation.io/
[24] https://ipfs.io/blog/24-uncensorable-wikipedia/

In Spain, the people of Catalonia wanted to run a referendum to gain independence from Spain. The government of Catalonia and the Catalonia parliament decided to run this referendum on October 1st, 2017. This referendum was deemed illegal by the spanish government. Consequently, the spanish authorities proceeded to ban any website that publishes voting instructions or polling locations. The goal was to suppress voters' turnout to deem the referendum void. Due to these censorship attempts, the Catalan government published a static website that can be hosted on IPFS that allows voters to know the location of their polling station. As IPFS can be accessed by any IPFS node or through any IPFS gateway, the launched website was much harder to be censored. Some interesting technical details of how this website was designed were discussed in this article. In response to the failed censorship attempts, the Spanish government resorted to violence and closed multiple polling location to disrupt the voting process[25]. However, more than 2.5 million voters representing more than 40% of voters have shown up to polling stations according to the Catalan government numbers.
In both discussed situations, IPFS has proven to be a crucial tool to fight censorship. Because of this interesting feature, many decentralized projects are working to utilize IPFS in their systems.

IPFS and Decentralized Apps

The adoption of IPFS is significantly growing by the decentralized applications (dApps). Most of these dapps use IPFS along with a blockchain protocol for financial settlements. Here is a comprehensive list of the dApps that use IPFS as their storage layer. In this section, a few of these dApps are discussed to highlight how IPFS plays an important role in the decentralization aspect of these dApps.

- **Decentralized Marketplaces**

  There are numerous projects that are building decentralized marketplaces which strive to reduce costs by cutting any intermediaries. OpenBazaar is one of the earliest decentralized marketplaces that allow users to exchange products. In this dapp, payments are settled using cryptocurrencies like BTC, BCC, and ZEC. OpenBazaar decided to integrate IPFS into their platform to allow participating storefronts to be available even if the originating storefront node is offline[26].

  Another project that uses IPFS, for example Origin Protocol, is building a platform for sharing economies that eliminates these intermediaries. In the Origin protocol, users can post ads to rent rooms in their homes, share their cars or anything similar. The ad content

[25] https://www.nytimes.com/2017/10/01/world/europe/catalonia-independence-referendum.html
[26] https://bitcoinmagazine.com/articles/openbazaar-integrating-interplanetary-file-system-to-help-keep-stores-open-longer-1460660998/

would be too expensive to store on the Ethereum blockchain that is used by the Origin protocol. In addition, there is no need to permanently save the ad data by writing it into a blockchain transaction. Instead, the ad content is saved to IPFS while only the payments are settled on the Ethereum blockchain. This allows for better decentralization compared to saving the content in a centralized server.

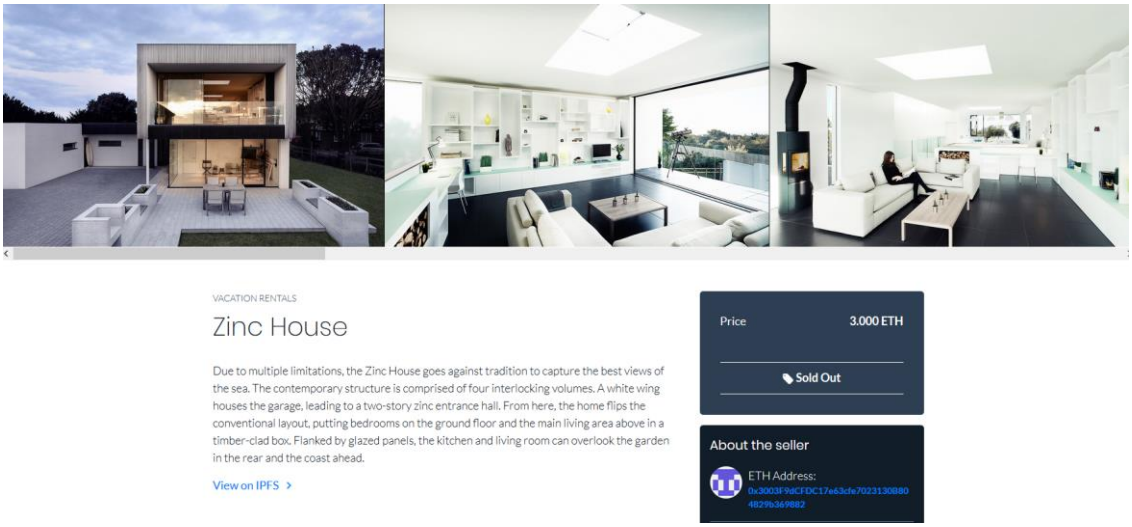

Figure 12. An example of a home sharing ad on Origin demo software where the ad data is stored in IPFS.

- **Decentralized Social Media**

  IPFS could be the future of decentralized censorship-resistant social media. An example of that is PeepEth which is a decentralized Twitter-like social media app. PeepEth stores the content posted by users (Peeps) to IPFS. To provide censorship-resistant content, the links to the IPFS data are then stored to the immutable Ethereum blockchain[27].

- **Decentralized video platforms**

  IPFS is also the storage platform of choice for many decentralized video and live streaming apps like DTube, Bittube and Dlive. IPFS significantly helps with the bandwidth requirements of such dApps. Since IPFS is content addressable, when a user requests to play a video, the content is served from the closest IPFS node that stores the content and not necessarily the node hosting it. This improves performance and reduces the bandwidth requirement of the originating node.

[27] https://peepeth.com/how

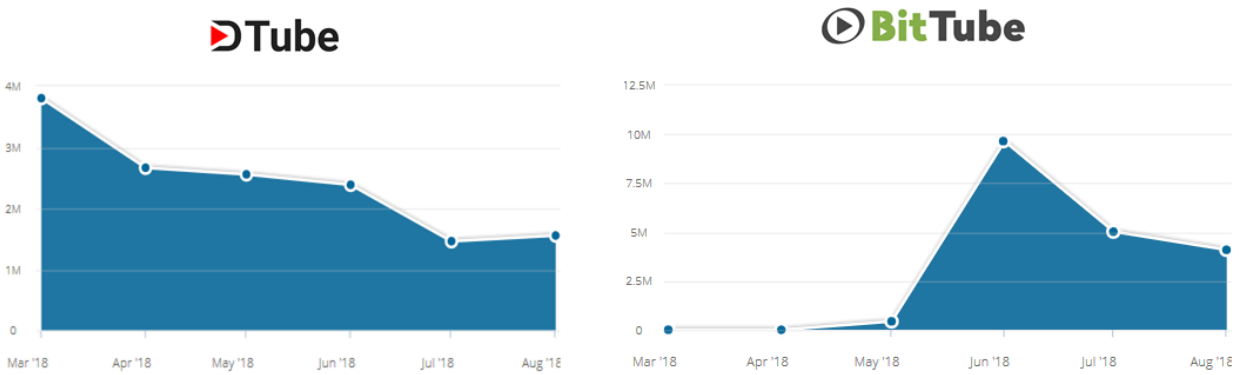

Figure 13. Number of visits of DTube and Bittube website. Both platforms' videos are hosted on IPFS. Source: Similarweb

- **Decentralized Gaming**

  Even decentralized games based on Non-Fungible Tokens (NFTs) have found that IPFS would be useful to them. NFTs greatly enhance digital collectable games. In these games, a user buys a digital collectable hoping that the *game demand* on this collectable would make it more valuable. As a result, the user could sell it for more money. The most famous example of these games is CryptoKitties. A similar game, Decentraland, has built a virtual city where players can buy and sell pieces of land around this virtual city. The game designers decided to use IPFS to store the information about the different pieces of land in the game[28].

- **Decentralized encyclopedia**:

  In 2017, Everpedia, which started as a fork of Wikipedia in 2014, decided to switch to a blockchain model and to launch a cryptocurrency to encourage content creation. They also announced future plans to use IPFS to store Everpedia content. In 2018, Everpedia completed integration with EOS blockchain. However, no specific plans about IPFS utilization have been announced yet.

How Filecoin can solve major IPFS problems

IPFS has proven to be a great system that protects content from censorship. However, the true potential of IPFS is still out of reach due to the availability problem. Users cannot keep their content that are hosted on IPFS available all the time without running their own IPFS node and/or pinning the content to make it available. To put this issue into context, this resembles asking every company that wants to own a website to run its own web server instead of paying a web hosting provider a monthly fee to host the website. The cause of this issue is that there are

[28] https://wiki.decentraland.org/index.php?title=How_the_engine_works

no guarantees that any IPFS node will keep a copy of your IPFS hosted content since nodes have no incentive to do so.

Filecoin is positioned to solve this major problem. The basic idea of Filecoin is to build an incentive system that allows users to pay other nodes to store their content and have it available on the IPFS network. At the same time, Filecoin provides guarantees to the paying users that their content is properly hosted and available through the use of the Proof of Space-Time (PoSt)[29] consensus mechanism. Through Filecoin, it will be possible to create censorship-resistant websites and simply paying a fee to have these websites hosted on a distributed network that can be accessed by everyone.

Another area where Filecoin would allow IPFS to achieve it's real potential is to create a decentralized alternative to the centralized cloud storage services like Dropbox, OneDrive or Google Drive. Cloud storage services have proven to be extremely useful and essential for many users. However, in the case of centralized storage services, users are essentially giving the storage service providers their data and assume they act honestly. It turns out that this is not always the case. While many of these companies may offer this useful service for free, they have multiple ways to benefit from the data they store. For example, In July 2018, it was revealed that Dropbox was sharing some of their users' data with Northwestern University researchers for more than two years between May 2015 to July 2017[30]. This data was used to conduct research about the collaboration habits of successful teams. The data sharing spurred controversy as Dropbox had not obtained the consent of these users before sharing their data. Many researchers questioned whether this practice was ethical[31].

Together, Filecoin and IPFS can improve the privacy of users. First of all, when users decide to use the Filecoin storage market to store their data on IPFS, it is normal that different files could be stored by different storage nodes (the users who store files to obtain fees). In addition, files in Filecoin are broken into multiple smaller segments that could be stored by different storage nodes. This means that, it is highly unlikely that a single storage nodes would have complete information about the paying users' data that they store. It is important to recognize that the significant improvement in users' privacy is not accompanied by severe loss of user experience. The reason is that IPFS is content addressable. Hence, when users decide to retrieve a file, the file would be constructed from its different segments that may be stored by a single or multiple storage nodes.

---

[29] J Benet, D. Dalrymple, and N. Greco, *Proof of Replication*, 2017

[30] https://hbr.org/2018/07/a-study-of-thousands-of-dropbox-projects-reveals-how-successful-teams-collaborate

[31] https://www.wired.com/story/dropbox-sharing-data-study-ethics/

Conclusion

In this analysis, we discussed why we believe in IPFS as a critical technology for the future's distributed internet. We started by discussing the problem that IPFS is trying to solve and a simplified technical discussion of the components of the IPFS system. We followed by introducing our analysis of the performance of the current IPFS network and its dynamics. The goal was to educate the readers about the expected behavior related to running an IPFS node. This analysis reveals some interesting information about the nodes' connection patterns and the IPFS node bandwidth usage. Afterwards, the analysis covered IPFS points of strength and the areas where further development is required.

We have also covered the diverse applications of IPFS. We discussed how IPFS has played a critical role in censorship resistance in multiple international incidents. IPFS is also turning into a preferred storage platform for decentralized apps. Many applications are already building on top of IPFS to benefit from its features. We concluded the analysis by analyzing how incentivization layers built for IPFS like Filecoin can play a significant role in increasing the usability and adoption.

Acknowledgements

This work is the cumulative effort of multiple individuals within zk Capital. We would like to thank Protocol Labs team who provided useful contributions, review and feedback; in particular Matt Zumwalt, Jeromy Johnson, Arkadiy Kukarkin, Lars Gierth, David Dias, and Juan Benet.