Designing a User Friendly Interface for IPFS

Dhruvi Shah • Dr. Diogo Oliveira

Florida State University
College of Communication & Information

Abstract

The InterPlanetary File System (IPFS) is a peer-to-peer distributed file system that focuses on content based addressing across the same system of files. Creating a merkle DAG file structure, IPFS encompasses a block storage structure with content addressed hyperlinks. This protocol can be seen as an improvement to the universal HyperText Transfer Protocol (HTTP). With IPFS being such a new topic in the field, it was imperative to start off by learning about the protocol itself and accessing the IPFS Desktop. The IPFS Desktop is an application that can be downloaded onto a laptop. It allows users to upload and retrieve files to and from IPFS. In accessing the Desktop, we found that its limitation in configuration has become a challenge for its adoption. Due to the lack of tools, users are not able to customize aspects of the protocol in order to fit their needs. To handle this, we propose a user interface (UI) that will allow users to have access to the different properties of IPFS and be able to alter them to best fit their necessities.

Introduction

Peer-to-peer file systems have been widely used for some time now. However, focusing on a decentralized version is something that is relatively new. Over the last few years, researchers have delved into creating a protocol whose core shifts from the norm of a centralized system like HTTP. Aiming to replace HTTP, IPFS uses a decentralized system to have access to its files. This type of system allows a node to base their requests on finding a node nearby that may have the content it is looking for. With a focus on interacting with your peers and using unique content identifiers (CIDs) to access files, the focal point of IPFS is featured to be its immutable data. Being able to adjust properties such as the garbage collection period, renaming your files, viewing your peers, and many more, allows a user to be able to navigate through this protocol in a friendly manner.

Exploring the current IPFS desktop along with the official IPFS documentation, I was able to navigate my way towards figuring out what the best way to install IPFS is. Learning the inner workings of IPFS, I was able to start to focus on which aspects of the protocol may have an impact on the way that files are being shared. Throughout this research project, I focused on learning about IPFS and altering the configuration metrics of the protocol to measure their impact on the upload and retrieval of files. Upon doing this, I was able to focus on reflecting my findings through building a GUI installation wizard that will allow users to install IPFS in a much more friendly manner.

Figure 1: IPFS Logo

Source https://en.wikipedia.org/wiki/InterPlanetary_File_System

References

Ali, M. S., Dolui, K., & Antonelli, F. (2017, October). IoT data privacy via blockchains and IPFS. Proceedings of the Seventh International Conference on the Internet of Things (IoT '17), 14, 1-7. https://doi.org/10.1145/3131542.3131563

Mah, B. A. (1997, April). An empirical model of HTTP network traffic. Proceedings of INFOCOM'97, 2, 592-600. doi: 10.1109/INFCOM.1997.644510

Shen, J., Li, Y., Zhou, Y., & Wang, X. (2019, June). Understanding I/O performance of IPFS storage: a client's perspective. 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), 1-10. Retrieved from https://ieeexplore.ieee.org/abstract/document/9068631

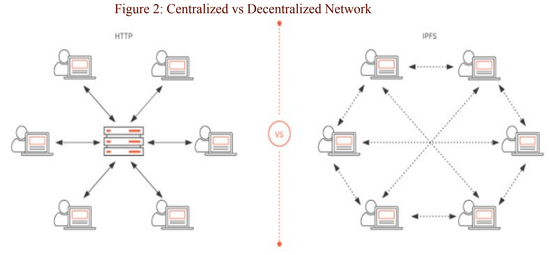
Steichen, M., Fiz, B., Norvill, R., Shbair, W., & State, R. (2018, July). Blockchain-based, decentralized access control for IPFS. 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 1499-1506. Retrieved from https://ieeexplore.ieee.org/document/8726493

Zheng, Q., Li, Y., Chen, P., & Dong, X. (2018, December). An Innovative IPFS-Based Storage Model for Blockchain. 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 704-708. Retrieved from https://ieeexplore.ieee.org/document/8609675

Discussion

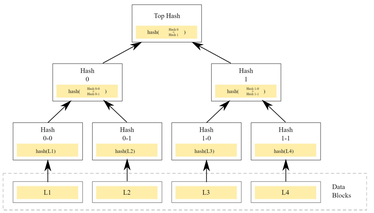
- IPFS is structured to focus on creating a protocol that does not change much over time.
- It is a decentralized network which means that one single point of failure cannot block others from accessing the information from that location. This is because, now, there will be many other local hosts who are providing access to that same information.

Figure 2: Centralized vs Decentralized Network



Source (https://medium.com/coinmonks/a-hands-on-introduction-to-ipfs-ee65b594937)

Figure 3: Merkle Tree Structure



Source: (https://en.wikipedia.org/wiki/Merkle_tree)

- It uses a distributed hash table and creates block exchanges. A distributed hash table is used to store the addresses of nearby peers that can provide the data blocks you are looking for.
- Its structure allows a node to base their requests on finding a node nearby that may have the content it is looking for
- Once a user adds a file to their local repository, it is stored as blocks. Each of these blocks has a unique content identifier (CID) which is created from the bytes that it contains and can be used to access the blocks. The CID is a special type of hash that is used to make sure the user is receiving the information it is requesting.
- Once a user requests for a block, a bitswap will occur. This is when a user can request blocks they need from their connected peers and also send blocks they have to other peers that might be requesting for them.
- The main structure of IPFS is the Merkle Directed Acyclic Graph (Merkle DAG). This allows IPFS to split the content of files into blocks. These blocks would allow different parts of the files to come from different sources and be authenticated quickly. Once the user receives a block from one of their peers, that peer is added to their network and they can become "connected"
- When a user has a block, they also have the option of pinning that block to their machine.
- In IPFS, there is a configuration metric called Record Lifetime. This is the amount of time your file will stay on your machine before it is automatically sent to the garbage (with the default being 24 hours). However, if you pin your block, it will stay on the machine until you unpin it.
- Once a user accesses the file, they can then become hosts of that file. This will allow nearby users to access that same file from the new host rather than traveling to that one primary centralized location every time.

Methods

- I started out by reading articles about what IPFS is and how it can be used
- I watched videos of camps based on learning about IPFS and on demonstrations of what the protocol is supposed to do in order to understand how to use it
- I used the official webpage to install IPFS and refer back to any commands I was unsure of
- I used the IPFS desktop to try to add and retrieve files from IPFS
- I also used the command line to access IPFS and found that it was much easier to use than the IPFS desktop
- I started exploring different configuration metrics of IPFS and what they meant
- Some of the main metrics that I focused on exploring were AutoNAT.Throttle, Datastore.GCPeriod, Datastore.HashOnRead, and Ipns.recordLifetime.
- To explore those methods, I used 3 virtual machines (VMs) to mimic adding and retrievin files between users
- I altered the configurations to see if they had any impact on the way files were being sent and received
- I had to disable the firewalls on the VMs in order to actually allow files to be added and retrieved

Figure 4: Some of IPFS configuration Metrics

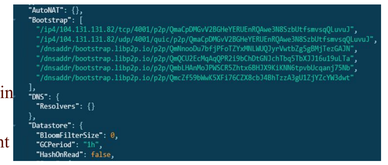
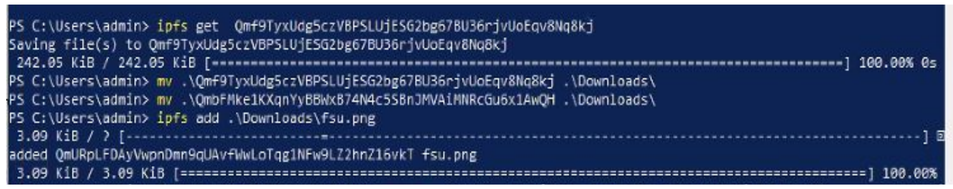


Figure 5: Using VM's to add files onto IPFS



Results

Altering the configuration metrics mentioned in the methods section, I found that for:

- AutoNAT.Throttle
  - If this is enabled, it will allow AutoNAT to limit the number of Network Address Translation (NAT) checks it does to every 30 minutes, limiting it to 3 peers per check with the default setting being enabled
  - Unless you are trying to upload multiple files in a short period of time, the default should be sufficient for the average use
- Datastore.GCPeriod
  - This will allow you to specify how frequently you want garbage collection to run with a default setting is of 1 hour
  - The default setting should be sufficient unless you are uploading a massive amount of files at a time
- Datastore.HashOnRead
  - If this is enabled, any block that is read will be hashed and verified before it is accepted by IPFS
  - The default setting is not enabled
  - Enabling this will cause IPFS to be a little slower because the CPU will need to be used so unless you are working with very important files, it would be best to leave this as the default
- Ipns.RecordLifetime
  - This allows you to alter the amount of time your file will stay on the network with the default setting being 24 hours
  - Increasing the lifetime of the file was very important because if I did not retrieve the files that I added within 24 hours, I was no longer able to access the file and would have to redo the entire process

Conclusion

IPFS is rapidly being researched. As it becomes more and more popular, we believe it is important to understand what it, how it is used, and what impact it can have on future of sharing and retrieving files. As I learned and experimented with properties of IPFS, I was able to understand the significance of the values placed on those metrics. In order to allow users to alter those metrics to best fit the purpose for their usage of IPFS, I tested a few of the properties that I deemed the most important. This includes AutoNat.Throttle, Datastore.GCPeriod, Datastore.HashOnRead, and Ipns.RecordLifetime.

Further Directions

Currently, I am working on building a GUI Installation wizard to allow users to have quick access to downloading IPFS. I am working with Java and the Swing library hoping to create a working product before the end of the semester. Ultimately, the goal is to create the installation wizard that will also allow users to go in and change the configuration metrics to their own preference but in a more user friendly way.

Figure 6: Future GUI sample