An improved P2P File System Scheme based on IPFS and Blockchain

Yongle Chen, Hui Li* , Kejiao Li and Jiyang Zhang
Shenzhen Key Lab of Information Theory & Future Network Arch.
Future Network PKU Lab of National Major Research Infrastructure
Shenzhen Engineering Lab of Converged Networking Technology
PKU Inst. of Big Data Technology,
Huawei & PKU Jointly Engineering Lab of Future Network Based on SDN,
Shenzhen Graduate School, Peking University, P. R. China.
Email: lih64@pkusz.edu.cn

***Abstract*—IPFS [1] is a peer-to-peer version controlled filesystem that synthesizes learnings from many previous successful systems. IPFS combines a distributed Hash table, an incentivized block exchange, and a self-certifying namespace [1]. IPFS is a peer-to-peer hypermedia protocol to make the web faster, safer, and more open. According to the characteristics of IPFS, we propose an improved P2P file system scheme based on IPFS and Blockchain. We address the high-throughput problem for individual users in IPFS by introducing the role of content service providers. Consider data reliability and availability, storage overhead and other issues for service providers, we provide a novel zigzag-based storage model to improve the block storage model that IPFS provides. Moreover, we introduce blockchain to combine IPFS with this storage model. According to analysis, this proposed scheme can effectively solve the above problems.**

Keywords--- P2P File System, Blockchain, Zigzag Codes.

I. INTRODUCTION

There have been many peer-to-peer file-sharing applications all over the world. Some applications have failed completely, and others are still alive and successful. BitTorrent is the latter, which succeeds in coordinating networks of untrusting peers (swarms) to cooperate in distributing pieces of files to each other [2]. Most notably, more than 170 million people use BitTorrent every month [3]. BitTorrent's protocols move as much as 40% of the world's Internet traffic on a daily basis [3]. However, these applications cannot cover all the use cases when compared with HTTP. As we all know, HTTP is one of the most successful file distributed system and being worldwide used. Many programs are implemented by Browser/Server architecture instead of Client/Server architecture due to the progressive development of browsers and enormous impact of HTTP.

However, The InterPlanetary File System (IPFS) [1], which is a peer-to-peer distributed file system, aims to replace HTTP and build a better web for all of us. IPFS seeks to connect all computing devices with the same system of files. IPFS thinks that HTTP fails to take advantage of dozens of brilliant file distribution techniques invented in the last fifteen years [1]. For example, Web infrastructure is not easy to be changed because of its large scale. HTTP/2 [4] was standardized in 2015 while the version of HTTP in common use is still the first definition of HTTP/1.1 [5], which was occurred in 1997. In some way, times are growing so fast that we need to upgrade user experience with something more than HTTP. IPFS is the first P2P file system working on this issue and people who proposed this novel system have seen more challenges.

While IPFS provides a high throughput content-addressed block storage model, we also notice that storage is a new challenge because we are entering the big data era. With the arrival of the information age, information has been integrated into people in all aspects of production and life. Internet, cloud computing, large data, mobile Internet and the emergence of a variety of intelligent terminals are to accelerate the explosive growth of the amount of data. In other words, lots of data transfers everywhere and it is quite difficult to make version control over these data. IPFS does not take into account the special circumstances of large content service providers. It is not easy for them to join the IPFS network. They need a set of nodes to store large amounts of data, and they need a new strategy to ensure the availability and reliability of large amounts of data and also reduce storage overhead since the block storage model that IPFS provides can bring some loss of benefits to service providers in the event of nodes failure, so we propose a scheme that combines three replication scheme and erasure codes storage scheme. IPFS draws lessons from many past successful systems and overcomes many challenges. Nevertheless, we think some aspects of IPFS can be improved better coupled with blockchain even though IPFS suggests placing the immutable, permanent IPFS links into a blockchain transaction [6] and until now this idea has not yet been realized.

Blockchain like Bitcoin [7], Litecoin [8] and Ethereum [9] has achieved significant success in the past few years and start to involve people's daily life. Narrowly speaking, a blockchain is a kind of chain data structure that combines data blocks in the sequence of time and it resembles a distributed ledger which is guaranteed by cryptography and cannot be tampered or forged. Broadly speaking, using block-chain data structure to verify and store data, using distributed node consensus algorithm to generate and update data, using cryptography to ensure the security of data transmission and access and using intelligence contract to program and manipulate data are all blockchain technologies [10]. These technologies have contributed to the development of all walks of life such as notarization security, file storage and e-commerce. Blockstack is a new blockchain-based system, which uses the Bitcoin

978-1-5386-2715-0/17/$31.00 ©2017 IEEE

blockchain and separates its control and data plane considerations [11]. In some ways, Blockstack and IPFS are a nice match and this is what we are going to describe.

This paper introduces a new IPFS seeking to upgrade user experience and deal with issues for content service providers. First, we address the high-throughput problem for individual users in IPFS by introducing the role of content service providers. Second, we provide a novel zigzag-based storage model to improve the block storage model that IPFS provides. Third, we introduce blockchain to combine IPFS with our storage model.

The structure of this paper is as follows. We provide the relevant background on Blockstack, IPFS and erasure coding in the next section. In section 3, we give a detail of the proposed scheme. We discuss the advantages of the proposed scheme in the section 4. Section 5 is the conclusion.

II. Related Work

In this section, we introduce some technologies related to our scheme.

A. IPFS

IPFS represents The InterPlanetary File System, which is a peer-to-peer distributed file system, aims to replace HTTP. IPFS synthesizes many of the best ideas from the most successful systems to date [1]. BitSwap Protocol is one of the best ideas we think that makes IPFS different from other block storage distributing system. Firstly, BitSwap Credit is a simple credit-like system which solves the problem of free-loading but never sharing. And BitSwap Strategy is continuing work, but their choice for now works fine in practice. They properly introduce the debt ratio in BitSwap Strategy and the debt ratio becomes a measure of trust which incentivize nodes to exchange lots of data. Moreover, BitSwap Ledger is very important to a connection between BitSwap peers. In the lifetime of a peer connection, BitSwap peers are looking to acquire a set of blocks (want_list), and have another set of blocks to offer in exchange (have_list) [1]. Sketch is as follows.

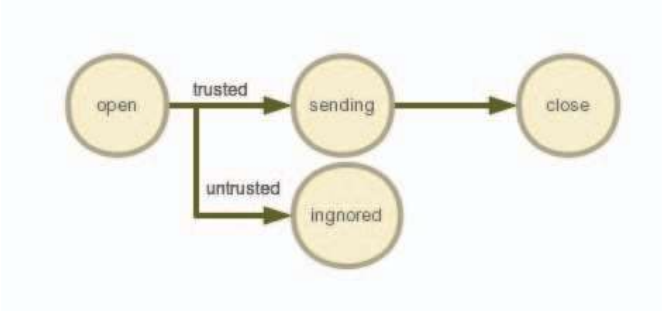

Figure 1. The lifetime of a peer connection

Object Merkle DAG is almost the best part of IPFS because Content Addressing, Tamper resistance and Deduplication are all useful properties from that design. They learn from the Git data structure and make the Git object model fit on top of the Merkle DAG. IPFS also defines many objects to model a versioned filesystem above the Merkle DAG. IPFS finds out that it is pretty hard to split large files into independent blocks and IPFS offers several alternatives which can be customized by users. Considering that using a hash of the content as the address of an object is obviously unfriendly, IPNS is implemented to help dealing with this problem.

B. Blockstack

Blockstack is a global naming and storage system secured by blockchains. Blockstack is released as open source software and currently powers a production PKI system for 55,000 users [11]. Blockstack is famous as a new, decentralized version of DNS and public-key infrastructure. Our scheme is deeply inspired by the design of Blockstack. In fact, Namecoin [12] is the first blockchain that aims to replace DNS root servers. However, the same people who create Namecoin decide to design Blockstack for all sorts of reasons such as blockchain security, network reliability and throughput, potential selfish mining, consensus-breaking changes and failure of merged mining [11]. Finally, they build Blockstack on top of Bitcoin network.

Specially, Blockstack does not have its own blockchain. In other words, Blockstack now is built on top of the blockchain of Bitcoin, which is layer 1 of Blockstack. Consequently, there are three more layers in Blockstack. Layer 2 is the virtualchain layer, which defines new operations that blockchain does not support. New operations are processed to construct a database that stores information on the global state of the Blockstack system. Above the virtualchain layer is the routing layer. Blockstack uses zone files for storing routing information, and the hash (zone file) is stored in the control plane. The top-most layer is the storage layer, which stores the real data. Users do not need to trust the top 2 layers because they can verify the data in the control plane. Impressively, Blockstack provides two modes of using storage layer and they are completely different from each other. One is mutable storage and the other is immutable storage. Blockstack uses its four layers to implement a new naming system and it is quite incredible.

C. Zigzag codes

To ensure data reliability and availability, distributed storage systems, such as sia [13] and storj [14], use erasure codes as the storage strategy. Obviously, these systems are similar to each other and files firstly encoded by Reed-Solomon (RS) codes will be then sent to other nodes to get stored. Addresses of the stored files are recorded in the blockchain. Decoding process is the opposite.

Zigzag code [15] is one of the traditional (n, k) Erasure codes. Erasure codes usually divide the original file into k blocks, and then k blocks are encoded into n blocks. Any k blocks out of these n blocks can reconstruct the original file, which is the most important property of MDS codes. As far as data fault-tolerance and storage space are concerned, erasure codes storage scheme can be better than replication storage scheme when some storage nodes fail. However, we also need to consider that reconstructing original files can be costly which brings a significant increase of disk I/O and repair bandwidth. Reed-Solomon (RS) code is one of the most famous erasure codes and also commonly used in the real life, but we employ zigzag codes to store our data for a better performance. In the case of one node failure, the repair bandwidth of zigzag codes can reach the theoretical minimum. The storage efficiency of zigzag codes is also high. To ensure the same double fault-tolerant as three replication scheme, we

Identify applicable funding agency here. If none, delete this text box.

can set the parameter (n, k) to ($k+2$, k) and the storage efficiency of that is $k/(k+2)$ which increases as k is increased.

III. PROPOSED SCHEME

Our scheme is designed to improve the high throughput problem of IPFS and provide a new solution for content service providers to better participate in the network of our scheme. In this section, we describe how our scheme deal with these problems and upgrade user experience. The other work in IPFS that we have not mentioned here is beyond our consideration.

A. Motivation

IPFS was originally designed to rely on high throughput to deliver data, but for personal computers, this is a bit inappropriate. According to akamai's research, in the fourth quarter of 2016, the global average connection speed was 7 Mbps, which means that the download speed was less than 1 MB/s (1 Mbps = 131072 B/s = 0.125MB/s). However, after opening the IPFS daemon, bandwidth occupancy directly soared to 700-800 KB/s, which causes the computer to react slowly and degrades user experience. In addition, the reliability and availability of data is also a problem. IPFS believes that every individual can save data or lose data at will since it is definitely for sure that files required by a person must be able to be found by BitSwap protocol if they really exist in the IPFS network. But this problem will become complicated when people who need files are not ordinary individuals but content service providers. First of all, service providers cannot rely solely on other people to provide content to their customers, which is very unwise for business. Once they lose the data, they lose the customers. Secondly, service providers need to store large amounts of data because of their role in the service market and data storage scheme becomes important consequently. However, IPFS only describes the block storage model and does not provide any corresponding storage strategy for this issue.

IPFS provides a block storage model, which is very vulnerable to lose data reliability and availability when servers of IPFS break down. Maybe IPFS has properly taken this situation into account and considers it as a normal using case, but it will be not allowed to happen in large data centers because they are content service providers. Service providers need to provide long-term stable service. Once videos, files or some services cannot be fetched online, customers and business opportunities will be both lost. This is what IPFS has not considered. Traditional distributed clusters are directly using the three replication strategy. Once a node breaks down, users can get data from another node, and it is nearly impossible for all three nodes to fail, but the three replication strategy need additional twice the storage space. We can refer to erasure codes storage strategy in this case, because it can reduce storage overhead while also ensuring data availability and reliability. Therefore, in the current distributed storage system, the erasure codes storage scheme and the replication storage scheme coexist.

So we propose a new storage strategy which combines triple replication scheme and erasure codes storage scheme to solve the large data storage problem of service providers. For high throughput issues, we propose a new system solution.

B. System Model

The scheme we propose is inspired by the Blockstack layers idea and Client/Server architecture. First, in the network of our scheme, individual users and content service providers are different from each other in some ways. Service providers need to maintain one or more nodes to protect the availability of their services, but individual users do not bother to do that. They can easily choose to join a service provider's network and take the service provider as a proxy node. So they do not have to afford the cost of maintaining a node or bear the impact of high throughput. In this way, individual users only need a client, which can be a simple browser, to possess a series of data exchange functions, such uploading and downloading. Assuredly, individual users can also be available to become a fully functional node. Figure 2 shows a typical case of the network of our scheme.

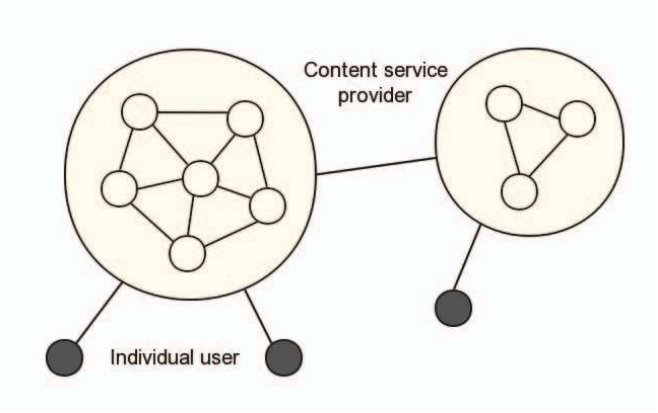

Figure 2. network of our scheme

Blockstack has four layers, with two layers in the control plane and two layers in the data plane. We also take this approach, but we make some changes to these layers as we are concerned about the original characteristics of IPFS network.

- Layer 1: Blockchain Layer

There are two ways to maintain a blockchain. One is to build a new blockchain in the new network, the other is to choose Bitcoin's blockchain as the underlying blockchain. We think both methods are desirable, but the second one is more recommended.

- Layer 2: Virtualchain Layer

This layer is the core functional area of our scheme. All the transactions are dealt with in this layer and then sent to be stored in the blockchain layer. Verifying the legality of transactions is the main job in the virtualchain layer. The legality of transactions refers to the fact that every transaction is constructed to include a signature from the sender's private key. Nodes that receive the transaction verifies the authenticity of the signature through the transaction sender's public key. The verified transaction will be added to the underlying blockchain and the unverified will be abandoned. We define a transaction for a node to issue a request to bind its own IP and its corresponding account (a public key generated by RSA algorithm, similar to a hash). We also define another kind of transaction for a node to issue a request to declare what files under its account. We divide files into two kinds, long-term immutable and occasionally mutable. Files that declared in the

transaction are usually immutable and mutable files under one's account do not need to write data into transactions. If some nodes want to claim that they have some mutable files to be changed, they only need to broadcast messages to other nodes. Other nodes who have the files will verify the authenticity of messages in the same way as described above. Finally, files get updated. These files can be broken down into blocks and blocks can be the same things as files in the network of our scheme. These files or blocks are open to the whole network and used to barter in BitSwap protocol, thereby exchanging data all over the world.

- Layer 3: Routing Layer

This layer is actually extracted from the second layer because the routing information is exactly from layer 2. This information holds the routing address of each account and their files or blocks under their account. Users do not need to trust the routing information because all the information can be verified in the first two layers.

- Layer 4: Storage Layer

This layer is where the data is actually saved. Since there are multiple types of data under each account, our scheme also provides two storage modes, mutable storage and immutable storage as described earlier. But for content service providers, their storage needs are far more complex than the simple block storage due to ensure data reliability and availability and reduce storage overhead. So we propose a new storage strategy which combines triple replication scheme and erasure codes storage scheme. As we all know, the three replication scheme is for frequently used data, and the erasure codes storage scheme is for infrequently used data, so we call the former hot data, the latter cold data. Combined with the BitSwap protocol, we propose a novel data storage scheme to distinguish between hot data and cold data, and also reduce storage overhead. Users do not need to trust the data in this layer because all the data can be verified in the first two layers.

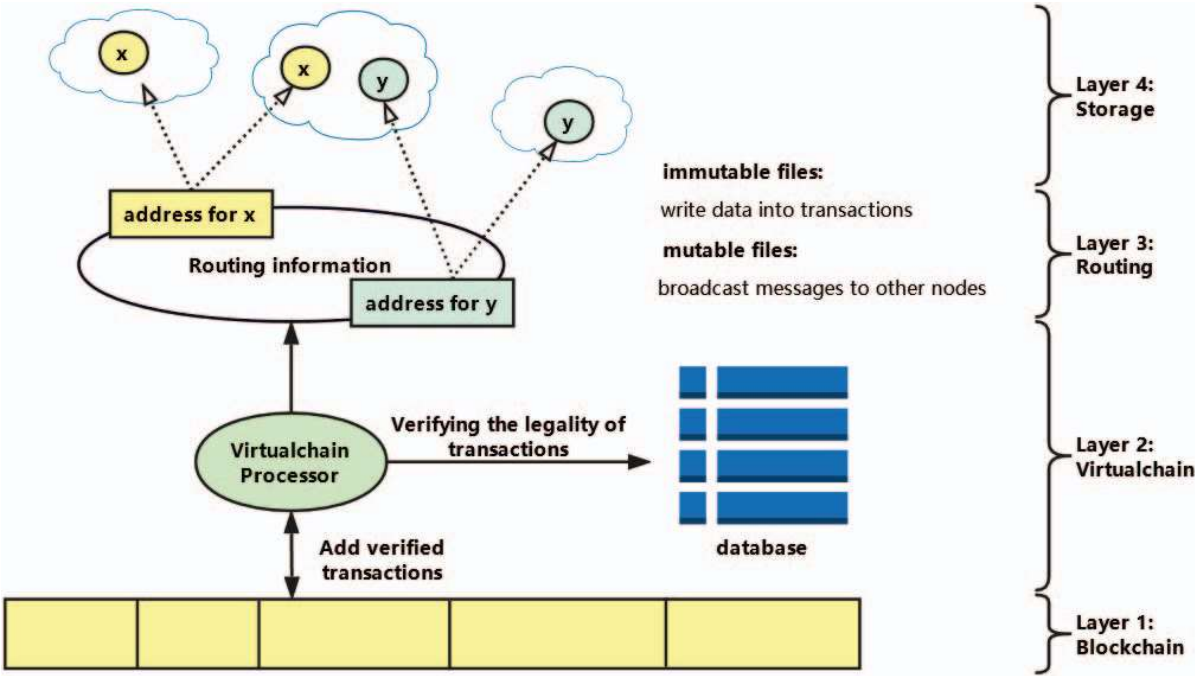

Figure 3. Overview of our scheme architecture

- Data Storage Scheme

BitSwap protocol is quite different from other data exchange protocol, such as BitTorrent. BitTorrent exchanges data in one torrent, but BitSwap peers are looking to exchange blocks in two lists and these blocks can be any part of any files. The two lists record what a node want and what a node possess. In a word, nodes seem to barter in a marketplace. We can find out that each data exchange implies the frequency of data usage. Based on the frequency of data usage, we can fairly give definitions of hot data and cold data. Hot data is what is read or written more than ten times per 90 days. Cold data is the opposite. This rule can also be modified according to the actual situation. Therefore, we add a data list to record every block's usage frequency in each node. If blocks are judged as hot data, they will be stored in three replication scheme, otherwise they will be stored in erasure codes storage scheme. In traditional erasure codes, we need to divide files into blocks and encode them. However, in our scheme, we only need to encode blocks directly because all kinds of files have already been divided into blocks in BitSwap protocol. In traditional (n, k) erasure codes, there are k data blocks and n-k parity blocks. It is data blocks that are exchanging in BitSwap protocol, so when one data block is lost, we can immediately repair that data block to continue the protocol. Base on this finding, we employ zigzag code to our scheme. The following figure shows the flow chart for storing data in (6, 4) zigzag code.

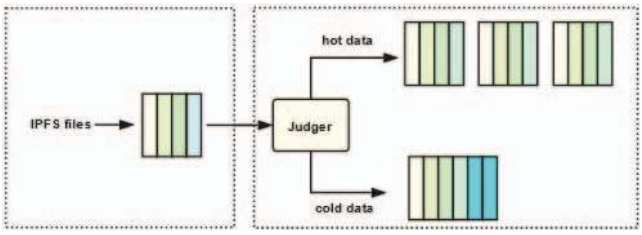

Figure 4. Flow chart of Data Storage Scheme

There are some differences between the original BitSwap protocol and the one in our scheme. The first step in the original protocol is to determine whether the other party is trustworthy according to each other's ledger. If they trust each other, they will advertise blocks that they want to (want_list) all connected nodes. Two peers will exchange data if both parties have their desired blocks. If one party does not have the data that the other party wants, it will look for a third party for data exchange, thus facilitating the success of data exchange finally. In our scheme, each node can maintain blocks information of all the other nodes in the local storage or obtain that directly from other nodes so that it does not need to confirm the want_list after peers have established a connection. They can send blocks straightforward to speed up data exchange. After the transmission of blocks, users can compute the blocks' hash to verify it matches the expected one regardless of whether there are malicious nodes or not. A mismatch will reduce the credit of the node on the ledger. Therefore, the next connection from that node is likely to be ignored.

IV. Evaluation

After the foregoing description, the process of exchanging data between nodes has been clear. Our scheme especially customized for large service providers is not only beneficial to them but also to the customers. Ordinary individual users can easily interact with the network of our scheme without having to maintain a fully functional node. This is because we introduce a blockchain and the information on each node can be saved to the blockchain. And the service provider node can quickly return this information to an individual user. In this way, individual users can interact with other nodes through service providers about content that they are interested in, thus avoiding communication at the local level with other nodes, and avoiding high throughput.

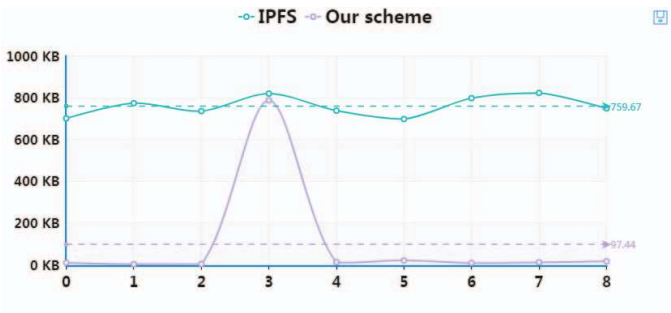

Figure 5. Bandwidth Occupancy

In our scheme, bandwidth occupancy will only increase if you interact with content service providers to download or upload something. Figure 5 shows a typical application scenario.

In addition, based on the characteristics of BitSwap protocol and the large amount of data that service providers want to store, we provide a novel storage scheme which combines three replication scheme and erasure codes storage scheme to replace what IPFS provides. Especially, we employ zigzag codes to store cold data. In terms of storage efficiency, zigzag codes which is one of MDS codes has the best storage efficiency in theory; In terms of repair time, experiments on a Hadoop cluster [16] shows that the repair time for zigzag code to repair one data block is the least compared to RS code and CRS code, which is exactly what we are concerned about; In terms of access latency, the access latency of the erasure code scheme is largely dependent on its repair time. Originally, IPFS provides a block storage model, but for content providers, this solution cannot guarantee data reliability and availability. We provide a novel storage scheme to ensure data reliability and availability, which is a qualitative change process. Therefore, our data storage scheme is more suitable for content service providers.

We introduce blockchain to combine IPFS with our storage model. Blockchain here is not just a 'bridge', but also give full play to its own advantages. Blockchain is a secure storage system. The underlying storage is replaced by our scheme and the data stored on the blockchain is exactly what the IPFS nodes need. We divide files into two kinds, long-term immutable and occasionally mutable, which is more appropriate with our storage scheme. Blockchain is also designed to support such functionality, including the preservation of permanent links.

As a result of the existence of the blockchain, the process of establishing a connection between nodes in the network becomes simple, because establishing a connection for the original IPFS nodes requires frequent communication to exchange lists from each other. With blockchain, communication becomes no longer frequent and the required data can be obtained directly from blockchain. So nodes in the network of our scheme can focus more on the data exchange than on the establishment of communication connections.

V. Conclusion and Future Work

In this paper, we introduce a novel scheme which have made some improvements to the IPFS architecture. With our scheme, individual users can no longer suffer from high throughput issues and content service providers can better interact with and benefit from the network of our scheme. We added a blockchain to the original IPFS so that each node's information can be saved to the blockchain. On this basis, the BitSwap protocol can work better and faster theoretically. We also optimize the large data storage scheme of IPFS for content service providers and propose a novel scheme which combines three replication scheme and erasure codes storage scheme. For the specific choice of erasure codes, we think that choosing zigzag is very reasonable. Our future work is to study other modules of IPFS and then do some depth customization optimization for content service providers.

ACKNOWLEDGMENT

This work is supported by National Keystone R&D Program of China (No. 2017YFB0803204, 2016YFB0800101), Natural Science Foundation of China (NSFC) (No. 61671001), GuangDong Key Program GD2016B030305005, Shenzhen Research Programs (ZDSYS201603311739428, JCYJ20170306092030521, JCYJ20150331100723974).

REFERENCES

[1] Benet J. Ipfs-content addressed, versioned, p2p file system[J]. arXiv preprint arXiv:1407.3561, 2014.

[2] B. Cohen. Incentives build robustness in bittorrent. In Workshop on Economics of Peer-to-Peer systems, volume 6, pages 68-72, 2003.

[3] BitTorrent, http://www.bittorrent.com/, [Online; accessed Jul. 28th, 2017].

[4] Fielding R, Gettys J, Mogul J, et al. Hypertext transfer protocol--HTTP/1.1[R]. 1999.

[5] Belshe M, Thomson M, Peon R. Hypertext transfer protocol version 2 (http/2)[J]. 2015.

[6] IPFS, https://ipfs.io/, [Online; accessed Jul. 28th, 2017].

[7] Nakamoto S. Bitcoin: A peer-to-peer electronic cash system[J]. 2008.

[8] Lee C. Litecoin (2011)[J]. 2011.

[9] Buterin V. Ethereum white paper[J]. 2013.

[10] Phoenix information, http://news.ifeng.com/a/20161029/50174092_0.shtml, [Online; accessed Jul. 28th, 2017].

[11] Ali M, Nelson J C, Shea R, et al. Blockstack: A Global Naming and Storage System Secured by Blockchains[C]//USENIX Annual Technical Conference. 2016: 181-194.

[12] Nameccoin, https://namecoin.org/, [Online; accessed Jul. 28th, 2017].

[13] Vorick D, Champine L. Sia: Simple Decentralized Storage[J]. 2014.

[14] Wilkinson S, Boshevski T, Brandoff J, et al. Storj a peer-to-peer cloud storage network[J]. 2014.

[15] Tamo I, Wang Z, Bruck J. Zigzag Codes: MDS Array Codes With Optimal Rebuilding[J]. IEEE Transactions on Information Theory, 2013, 59(3):1597-1616.

[16] Lu L, Li H, Chen J, et al. On the implementation of Zigzag codes for distributed storage system[C]// IEEE International Conference on Big Data.IEEE,2015:1791-1796.