EthDrive: A Peer-to-Peer Data Storage with Provenance

Xiao Liang Yu[1][2][3]✉, Xiwei Xu[1][4], and Bin Liu[1][4]

[1] Data61, CSIRO, Australia
[2] Carnegie Mellon University, Pittsburgh, USA
mr.y.xiaoliang@ieee.org
[3] The University of Melbourne, Melbourne, Australia
[4] School of Computer Science and Engineering, UNSW, Sydney, Australia
{firstname.lastname}@data61.csiro.au

Abstract. In this digitally connected world, cloud storage plays an important role in allowing users to store, share, and access their files anywhere. The conventional cloud storage is a centralised system that relies on a trusted party to provide all services. Thus, the cloud provider has the ability to manipulate the system including, for example, changing the user's data and upgrading the infrastructure software without informing the users. A centralised authority is also a single point of failure for the entire system from the software architecture perspective. In this paper, we demonstrate EthDrive, which is a peer-to-peer data storage that leverages blockchain to provide data provenance. A peer-to-peer architecture eliminates the centralised authority and achieves high availability since the data is normally replicated on multiple nodes within the network. The blockchain is used to provide a tamper-proof data provenance which could be used to check data integrity. We use an IoT scenario to show the feasibility of EthDrive[5].

Keywords: Information security · Data storage systems · Internet of Things · Distributed databases

1 Introduction

Cloud storage reduces the need for local storage and enables efficient file sharing. The conventional cloud storage is a centralised system that relies on a trusted party to provide all services. In such a centralised architecture, first, data on cloud storage is readable, modifiable, and deletable by the cloud provider. For example, on 31 Jan 2017, six hours of database data of the famous git repository manager gitlab.com was deleted by just a simple delete command[6]. Second, the availability of the data on cloud storage is questionable [5] while saving copies to

[5] The demo video for EthDrive is available at `http://video.ethdrive.org/latest`. The alpha version of EthDrive is now available at npm (`http://dist.ethdrive.org/latest`). The website for EthDrive is at `http://ethdrive.org`.

[6] The announcement from `gitlab.com` can access at `https://about.gitlab.com/2017/02/01/gitlab-dot-com-database-incident/`

X. Franch J. Ralyté, R. Matulavičius, C. Salinesi, and R. Wieringa (Eds.):
CAiSE 2017 Forum and Doctoral Consortium Papers, pp. 25-32, 2017.

multiple cloud providers to increase data availability needs a considerable effort due to vendor lock-in [1].

In this paper, we propose EthDrive solve those problems. EthDrive is a distributed storage client which allows users to upload, download, and share files. It utilizes content-addressable distributed file system as the underlying storage and blockchain as the distributed database. All files are stored in the distributed storage while the file records are registered on the blockchain. Files are indexed by a unique file ID which is associated with the content address, storage system used, last uploader, last update time, authorised editors, authorised administrators (people who can add and remove editors), immutability (whether the file can be updated or deleted), and comments. That information is stored in the blockchain via a smart contract, which is a program deployed and running across the blockchain network [7]. Blockchain is an emerging technology that allows participants in an industry ecosystem to transact with each other without relying on a central trusted authority to record and validate transactions [8].

Information in EthDrive is secured because only authorised personnel can mutate particular information. Information like the editor and time of the last update is assigned by smart contract. Any invalid attempt will be denied by the blockchain network. Read access control is enabled by encrypting the content address and/or the file content itself. From the nature of both blockchain and content-addressable distributed file system, high availability is easily achievable while data integrity is guaranteed. In addition, it provides data provenance driven by the distributed consensus which is difficult to achieve for conventional cloud storage without a trust [2]. EthDrive resolves users' most concerns when using cloud storage [4] - security, control, vendor lock-in, configurability, and speed to activate new services and expand capacity.

There are related work which aim to address those problems. Such as, Permacoin [6], Sia [7], and Storj [8]. However, they require currency in the respective blockchain valuable while EthDrive can leverage multiple blockchains with different mechanisms and benefit from pre-existing cryptocurrency ecosystem.

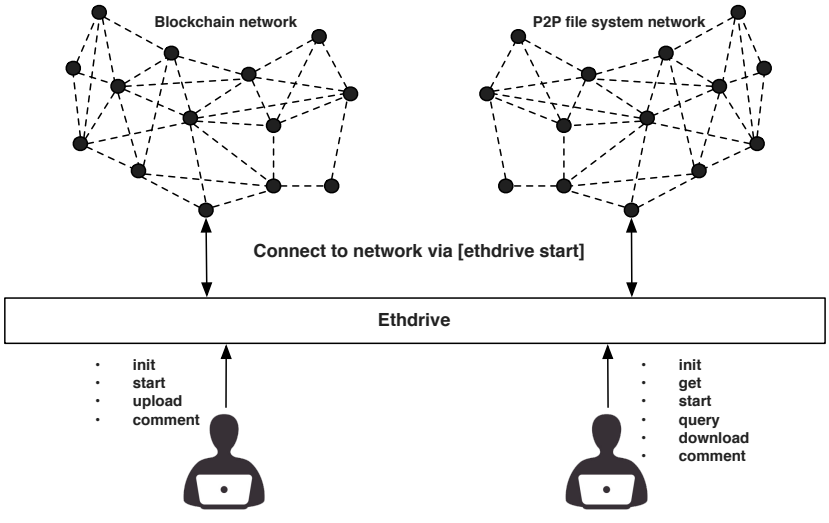
2 Background

Conventional Cloud Storage In the scenario of sharing files using conventional cloud storage, cloud provider stores the data uploaded by the file publisher. In this model, it is not transparent to the users about how the cloud provider would deal with those files. File integrity and availability don't often come with their services [3]. Even file integrity is included in their Service Level Agreement(SLA), file consumers are unable to verify it unless putting extra efforts. For provenance, we can only trust the cloud provider which can be unfavourable [2].

Peer-to-Peer Data Storage and File Sharing Peer-to-peer technology has been used for distributed data storage and file sharing. Such a system allows

[7] `https://sia.tech/`
[8] `https://storj.io/`

Fig. 1. Overview of EthDrive



Available commands of EthDrive

`init`	Configure the environment	`start`	Start daemon process of EthDrive
`new-account`	Create a new Ethereum account	`list-account`	List Ethereum accounts
`unprovide`	To NOT be a provider a file	`get`	Show the content of the file
`provide`	To be a provider of a file	`comment`	Get or add a comment to a file
`query`	Get metadata about a file	`upload/download`	Upload/download a file

users to access data that is stored in other computers connected to the same peer-to-peer network. A centralised server is not required. Existing products include BitTorrent [9], IPFS (InterPlanetary File System) [10] and etc. All these Peer-to-peer techniques use different mechanisms to share data with peers and to replicate data across nodes. IPFS is an open source content-addressable, globally (peer-to-peer) distributed file system for sharing a large volume of data with high throughput. IPFS has several limitations, including 1) originality of files cannot be verified, and 2) limited mutability.

Blockchain The blockchain data structure is a time-stamped list of blocks. *Blocks* are containers that aggregate *transactions*. The blockchain provides an immutable data storage where existing transactions cannot be updated or deleted. Cryptography and digital signatures are used to prove identity and authenticity. For a basic description and technical details of blockchain, please refer to [8]. *Smart contracts* are programs running on blockchain. All interactions with smart contracts are recorded immutably, and verified and accepted by the whole blockchain. Ethereum[11] is the most widely-used blockchain that supports general-purpose (Turing-complete) smart contracts at the time of this research being conducted.

[9] `http://www.bittorrent.com/`
[10] `https://ipfs.io/`
[11] `https://www.ethereum.org/`

3 EthDrive

By using blockchain as a reliable distributed database with content-addressable peer-to-peer storage, EthDrive guarantees that the files to be downloaded from storage providers are intact. EthDrive is currently based on Ethereum and IPFS. However, it does not couple with these specific implementations. EthDrive is known to facilitate scenarios with no trusted third party or the file content must be identical to the one uploaded by the indicated file uploader. Fig. 1 gives an overview of EthDrive with all the available commands. File publisher and file consumer need to configure the environment via `ethdrive init` and start daemon process of EthDrive via `ethdrive start` before other operations.

Users can upload and download files via `ethdrive upload` and `ethdrive download`. A user can use `ethdrive upload` to upload a file, then announce the file ID to allow others to locate the file. A user can also share a file without uploading it when he/she knows the IPFS address of the file via `ethdrive upload` with `--not_provide` as argument.

Every file is retrieved via `ethdrive download` based on a customised file ID, which is assigned by the file publisher who uploads the first version of the file. File ID is logical and understandable. Conceptually, a file ID is associated with information including the information publisher, a purpose, and the data related to that purpose. Using different naming space is achieved by using other smart contracts with the same interface.

Read access control is implemented by uploading an encrypted file content address and/or an encrypted file content using symmetrical encryptions. Only personnel having the correct key can download and/or decrypt the file content. Write access control is enforced by the smart contract, which implements a permission control based on the blockchain account addresses. When a user calls the smart contract to modify the file record for a particular file ID, the smart contract checks whether the user has the permission based on the blockchain account public key and the corresponding signature.

Users could add and get comments via `ethdrive comment` to/from a file record if the file publisher enabled this feature. Every comment has three information associated, including the file ID it comments on, the comment author and the comment content. Since all comments are stored in the blockchain, the provenance of the comments is assured in the same way as file records. The comment mechanism enables flexible interactions between the file author and the comment author. For example, a group of people uses EthDrive to publish an article. Comments are accepted. There are reviewers being invited to review this article. While everyone can post their comments on to the file record, the authors will only concern about the reviewers' reviews (in the form of a comment). By checking the comment authors, those reviews can be identified, authenticated, and confirmed to be intact without a centralised party.

Any node in the network can become a provider of a certain file via `ethdrive provide`. Also, people who download the data becomes a temporary provider. The Distributed Hash Table (DHT) of IPFS is able to connect them when a correct data content address is given.

3.1 Properties

Data Integrity Data integrity is achieved because 1) the content address of a file will change when the file is modified and 2) the content address is stored on the blockchain and can only be updated by authorised users.

Data Availability Both blockchain and content-addressable distributed file system are based on the peer-to-peer network. Users of EthDrive are in the Ethereum blockchain network and IPFS DHT network at the same time. There are two types of file provider in the IPFS network, including permanent providers and temporary providers. Permanent providers are nodes that provide the files. They might be the file publishers or storage provider who have some contracts with the file publishers to help them serve the files. Temporary providers are the ones that have downloaded the file. They will provide the file until the cache is cleaned. There are nodes providing other files which share some common blocks with the file. Such nodes are also providers of parts of the file.

Traceability The editor and the time of every update are recorded in the smart contract and accepted by the whole blockchain network. Thus, the transactions recorded on the blockchain provide the provenance of the file record. Using the provenance, file consumers are able to confirm that the content is the one that originally comes from the intended file publisher.

Customisability EthDrive is not coupled with the current implemented smart contract. It can work with the smart contract that implements the defined interfaces. This feature allows different logics to be used making EthDrive highly customisable to fit different situations while modification of the client program is not needed.

3.2 Limitations

Limitations of blockchain and IPFS apply to EthDrive. For example, Ethereum blockchain needs to be synchronised before using EthDrive. The database is growing without a bound - synchronisation can be time-consuming and the database might occupy a large space. This can be solved by deploying the light client protocol version of Ethereum which is under development at the time the research is being conducted. The download speed of IPFS can be slow down sometimes because the DHT routing of IPFS is still immature at the time the research is being conducted. However, the famous 51% attack[12] on the blockchain will only affect the data availability property of EthDrive while other main properties are intact.

4 IoT Use Case

This section gives a detailed example to show how EthDrive can improve the quality of data storing in the context of Internet of Things.

[12] The general term refers to a situation where malicious miners have more mining power than honest miners do

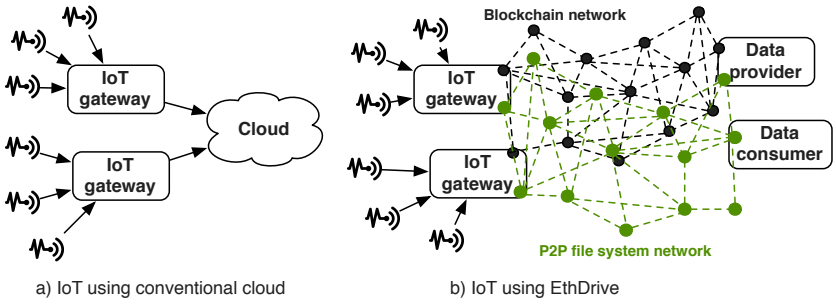


Fig. 2. IoT Data Storing Architecture

Fig. 2 a) shows IoT using conventional cloud storage, which includes sensors, IoT gateways, and a cloud storage. Sensors are used to collect data. IoT gateways are presented to aggregate sensor data, translate sensor protocols and process sensor data before data is sent to the cloud storage. We assume IoT gateways don't have enough storage for storing the data generated. Cloud storage providers are responsible for storing and distributing the data generated from the sensors.

Fig. 2 b) gives a conceptual architecture of IoT using EthDrive. Nodes of the peer-to-peer network are made up of IoT gateways, storage providers, and data consumers. EthDrive provides the provenance of the data and guaranteed data integrity to increase the confidence for the data from the data consumers' perspective. The following shows the steps of how EthDrive is used.

Registration on the network Storage provider registers itself to the network by uploading (`ethdrive upload`) an empty file named *IOT_IDS*. The file is used to store the ID of all the IoT gateways which the storage provider decides to provide the files they generate and upload using EthDrive. The file ID represents the storage provider ID on the network. IoT gateway registers itself to the network by uploading an empty file named *DATA_IDS*. The file is used to store the file ID of all the files the IoT gateway generates and uploads using EthDrive. The file ID represents the IoT gateway ID on the network.

Registration of a IoT gateway to a storage provider IoT gateway registers itself to a storage provider by adding a comment (`ethdrive comment`) on the file record whose file ID is the storage provider ID. The storage provider periodically checks if there is any new comment. If there is, the storage provider then decides whether accepts the request. If it accepts, the IoT gateway ID will be added to the file *IOT_IDS*. The modified file *IOT_IDS* is then uploaded (`ethdrive upload`) to replace the old file.

Data uploading IoT gateway packs data collected from the sensors into a file and uploads (`ethdrive upload`) it using EthDrive with a file ID. The file ID is added to the file *DATA_IDS*, then *DATA_IDS* is uploaded to replace the old file. The storage provider periodically checks for all the IoT gateway IDs registered to see whether they have uploaded any new file. If they do, the storage provider will call `ethdrive provide` to become a provider of the newly uploaded files.

Access data EthDrive uses a file ID to locate the data content address in Blockchain. If it is encrypted, data consumer will supply the correct key to EthDrive to get (`ethdrive get`) the data content address. Then, it downloads the data to the data consumer's local storage. If the data integrity is corrupted, the data will not be available since the IPFS component of EthDrive won't be able to locate it by the data content address stored in the blockchain. The time of upload and the data publisher can be verified by retrieving the file record information via `ethdrive query`.

5 Performance Data

Uploading files via `ethdrive upload` only creates a transaction for storing the file content address on to the Ethereum blockchain and uploads the file content to the *local* IPFS node. Thus, there is no network latency for `ethdrive upload`.

We conducted an experiment to evaluate the performance of downloading files using `ethdrive download`. In this experiment, we compared the download performance between EthDrive and a conventional cloud storage, AWS S3[13] in US standard region. We deployed four EthDrive nodes in four locations: Hong Kong, Tokyo (Japan), Sydney (Australia) and Pittsburgh (USA). The locations are selected so that they are geographically distinct to simulate a global peer-to-peer network. The test files are generated by a random data generator. We queried IPFS network and confirmed that there is no block in the test files shared with any pre-existing file on the network. All the test files are provided by three providers. An Ethereum private blockchain is used.

Fig. 3. The Experiment Result

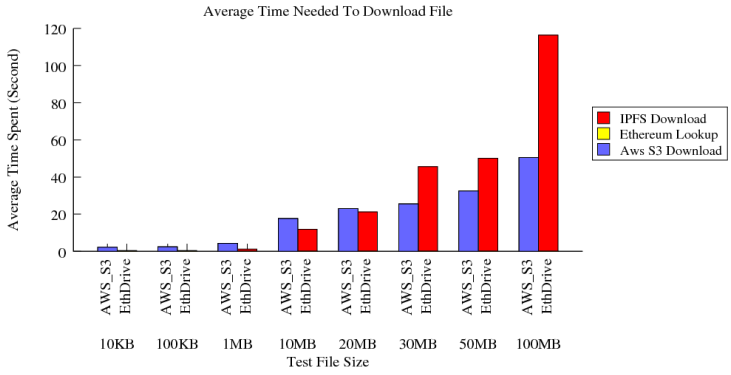


Fig. 3 shows the result of this experiment. All the results shown in the figure are the average values of ten identical tests. The Ethereum lookup time of the EthDrive for all the tests on average takes 0.038 seconds with a standard deviation of 0.000166. The Ethereum lookup time was almost constant in all tests

[13] `https://aws.amazon.com/s3/`

which is expected because the retrieval of the file content address is done on the local blockchain data. Most of the time EthDrive spent is for downloading the files from the IPFS network. The yellow column (Ethereum Lookup Time) in Fig. 3 is too small to be viewable. In this experiment, EthDrive is faster than AWS S3 when the file size $\leq$ 20MB and is slower when the file size $>$ 30MB. This shows that IPFS is not handling large files efficiently at the time of this experiment being conducted. This result suggests that when the file size is small, additional features provided by EthDrive doesn't come with a performance penalty.

6 Conclusion and Future Work

By utilizing blockchain and content-addressable distributed storage, the demonstrated EthDrive can have all the basic requirements for a reliable cloud storage fulfilled. It includes file integrity, reliable permission control, and high availability. In addition, EthDrive provides custom file ID as a means to locate the files to facilitate easy access, sharing and management, and comment mechanism to increase its versatility. For our future work, the following features/functions can be implemented on top of the current version: 1) An extension which allows users to pay to particular parties to have them provide certain files to ensure the minimum availability of those files. The payment will be done automatically when they are providing the file and terminated when they do not. 2) After integrating with other content-addressable distributed file systems, files can be downloaded from all the networks concurrently to boost the performance. 3) Mount files filtered by some conditions to the file system via Filesystem in Userspace (FUSE) to facilitate convenient use. We also plan to evaluate EthDrive more systematically through using additional evaluation criteria.

References

1. H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon. Racs: a case for cloud storage diversity. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 229–240. ACM, 2010.
2. M. R. Asghar, M. Ion, G. Russello, and B. Crispo. *Securing Data Provenance in the Cloud*, pages 145–160. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
3. I. Ion, N. Sachdeva, P. Kumaraguru, and S. Čapkun. Home is safer than the cloud!: privacy concerns for consumer cloud storage. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 13. ACM, 2011.
4. J. Ju, J. Wu, J. Fu, Z. Lin, and J. Zhang. A survey on cloud storage. *JCP*, 6(8):1764–1771, 2011.
5. B. Mao, S. Wu, and H. Jiang. Improving storage availability in cloud-of-clouds with hybrid redundant data distribution. In *2015 IEEE International Parallel and Distributed Processing Symposium*, pages 633–642, May 2015.
6. A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing bitcoin work for data preservation. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 475–490. IEEE, 2014.
7. S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI Matters*, 1(2), Dec. 2014.
8. M. Swan. *Blockchain: Blueprint for a New Economy*. O'Reilly, US, 2015.