computers

MDPI

Article

An Architecture for Distributed Electronic Documents Storage in Decentralized Blockchain B2B Applications

Obadah Hammoud [1], Ivan Tarkhanov [2,3,*] and Artyom Kosmarski [2]

1 Department of Engineering Cybernetics, National University of Science and Technology "MISiS", 119071 Moscow, Russia; obadah.hammoud@gmail.com
2 Laboratory for the Study of Blockchain in Education and Science (LIBON), State Academic University for the Humanities (GAUGN), 119049 Moscow, Russia; kosmarski@gaugn.ru
3 Institute for Systems Analysis Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, 117312 Moscow, Russia
* Correspondence: itarkhanov@gaugn.ru

Abstract: This paper investigates the problem of distributed storage of electronic documents (both metadata and files) in decentralized blockchain-based b2b systems (DApps). The need to reduce the cost of implementing such systems and the insufficient elaboration of the issue of storing big data in DLT are considered. An approach for building such systems is proposed, which allows optimizing the size of the required storage (by using Erasure coding) and simultaneously providing secure data storage in geographically distributed systems of a company, or within a consortium of companies. The novelty of this solution is that we are the first who combine enterprise DLT with distributed file storage, in which the availability of files is controlled. The results of our experiment demonstrate that the speed of the described DApp is comparable to known b2c torrent projects, and subsequently justify the choice of Hyperledger Fabric and Ethereum Enterprise for its use. Obtained test results show that public blockchain networks are not suitable for creating such a b2b system. The proposed system solves the main challenges of distributed data storage by grouping data into clusters and managing them with a load balancer, while preventing data tempering using a blockchain network. The considered DApps storage methodology easily scales horizontally in terms of distributed file storage and can be deployed on cloud computing technologies, while minimizing the required storage space. We compare this approach with known methods of file storage in distributed systems, including central storage, torrents, IPFS, and Storj. The reliability of this approach is calculated and the result is compared to traditional solutions based on full backup.

Keywords: blockchain; distributed file storage; electronic document; reliability; Hyperledger

check for updates

Citation: Hammoud, O.; Tarkhanov, I.; Kosmarski, A. An Architecture for Distributed Electronic Documents Storage in Decentralized Blockchain B2B Applications. *Computers* **2021**, *10*, 142. https://doi.org/10.3390/computers10110142

Academic Editors: George Angelos Papadopoulos and Grigorios E. Koulouras

Received: 13 September 2021
Accepted: 1 November 2021
Published: 4 November 2021

1. Introduction

The need to exchange electronic documents between organizations has existed since the early 1970s. In the process of development of the EDI (electronic data interchange) concept, the first standard was elaborated in 1996 [1]. The first software solutions worked on the principle of p2p, using secure connections (VPN, etc.). Later, solutions based on VANs (value-added networks) emerged, organized as local intermediate nodes that manage document transmission using different transport protocols (FTP, AS2, etc.) [2]. This configuration is still the most popular and is usually managed by a large telecommunications company, a consortium of providers, or government agencies (e.g., Peppol, EDF+) [3].

Large corporations develop their own systems for distributed and secure data exchange between branches which mostly are based on centralized data storage architecture. Systems of this kind—i.e., based on a centralized architecture (a single database)—are vulnerable to hacking and untraceable modifications [4]. Still, they are popular because they present a relatively rapidly deployable solution, and they are efficient in cost and speed of implementation. Decentralized systems, on the other hand, are difficult to deploy and expensive, but in general, they are more reliable, even though individual nodes of such a system are still vulnerable. One promising technology for improving the reliability of information systems is blockchain (or distributed ledger technology, DLT) [5].

It is no secret that the popularity of DLT in the last 10 years has led to a heightened interest in such solutions in the enterprise market and the emergence of new DLT platforms customized for it [6,7]. Applications that communicate with each other not through a common centralized database and use blockchain to store data are called DApps [8]. They attract more and more attention from the developers of enterprise solutions. As mentioned above, the main obstacle for using DApps is the high cost of their implementation, which significantly reduces the range of potential users [9].

In order to reduce the cost of the system (with the data stored in the DLT), the latter could be combined with a distributed file storage that operates on the principles of p2p or similar ones. In this case, only the files will be stored in the storage, while the metadata of these files is kept within the DLT. Many researchers consider this approach promising [10,11]. If it is possible to solve the problem of increasing the efficiency of data storage—i.e., to reduce the cost of implementing DApps—it may serve as a driver for the introduction of such systems in companies. Therefore, this study focuses on the development of the method of effective data storage in DApps. It is important to understand how much a system with such an architecture is suitable for a B2B model, therefore it is necessary to calculate the reliability of such a system as a whole and compare it with traditional centralized solutions.

Another serious problem that interferes with the implementation of DApps is attached to this insufficient elaboration of the storage of large data in DLT and the use of horizontal scaling approaches. Famous DLT platforms (Hyperledger, Ethereum) by themselves solve the problem of scaling data stored in DLT, but the issue of connecting them with external file storages has not been resolved. Therefore, another important aim of this study was to conduct a series of experiments and choose the optimal DLT platform for the proposed document storage approach and resolve scaling problem.

The article is structured as follows. Section 2 considers the shortcomings of existing projects. Section 3 presents the basic architecture and approaches for organizing file storage. Section 4 describes a prototype performance testing experiment that justifies the choice of a DLT platform. Section 5 is devoted to assessing the reliability of the presented architecture and Section 6 discusses the limitations of the presented approach. At the end, a summary and possible directions for future research are summarized.

2. Related Works

Well-known projects are used for file exchange without DLT—BitTorrent file system [10] and Storj [12]. Both projects are aimed at end-users (b2c) and do not consider handling electronic documents. A user requests to save his data on other users' computers. To do this, the user has to pay those who host his data using cryptocurrency (Storj or BT token). Our goal is to develop an approach which focuses on enterprises with the required level of reliability and allows them to store files in a distributed and secure way without using cryptocurrency.

Numerous torrents built on peer-to-peer principles work as follows. When a node requests a file, the node makes a copy of the file and becomes able to share it. Thus, it is possible to have a file with thousands of copies (if thousands of nodes have requested the same file). At the same time, there is no node availability control and no control of remote files—i.e., a situation may arise that the file is no longer available. Such a solution cannot be considered as a reliable b2b solution.

In the last decade, many projects and works have been devoted to the development of DApps frameworks and DLT platforms. According to a study [13], hybrid systems which combine on-chain and off-chain data (systems which combine blockchain networks with other storage methods like IPFS) overcome scalability issues, but have main challenges, as data aggregation and integration. This means accessing data from several nodes can be problematic. DADS [14] is an application deployment system, which uses IPFS to store applications instead of using dedicated servers. The system in general is designed to handle the storage of different applications, with different users and wallets.

HyperBSA [15] is a solution that suggests caching files which are accessed the most, using a new caching algorithm LBN which manages memory space using HashMap. Additionally, it splits data into two categories, state and continuous data, and then it handles each type separately. In our research, we focus on file storage. According to the original HyperBSA paper, the system itself is not effective in the matter of searching for archived files, and that is why this solution has a mode to connect to other storage solutions like IPFS.

Another solution [16] discusses a new method for data storage and recovery in industrial blockchain networks which tries to increase the data repairing speed, improve the way data is monitored, etc. Nevertheless, this method does not consider files storage.

Li et al. [17] presented a model to use blockchain to store large volume of data in IoT networks. The contribution was to make blockchain miners in charge of accessing data instead of using servers. In order to communicate with IoT devices, the authors suggested using an edge device which can be a mobile phone or a computer. However, this model cannot be used by companies for data storage, as companies cannot trust random miners for data permanent storage.

The review of the projects showed that there is no optimal b2b solution for the reliable storage of electronic documents based on the blockchain, which does not require duplicating the data of all files. This paper presents a novel method which allows companies to store data in an efficient and trusted distributed structure. The provided method suggests having a special data structure, which has clusters managed by blockchain to prevent data tampering, and each cluster includes a group of data servers in addition to a dedicated server which stores the according Erasure code, and the distribution is built in such a way that data can be recovered if one of the servers gets damaged, while the storage is reduced by 25% compared with a full backup system. These clusters are managed by a load balancer, which handles all the internal processes, and ensures data access transparency in such a way that the end user is not included in all internal operations. The load balancer also defines rules for deleting and modifying data, so if a hacker could access the system, no damage would be made. To assess the feasibility of creating such a b2b system, it is necessary to calculate its reliability and compare it with traditional systems

3. Materials and Methods

In this section, we present the proposed architecture and the required processes to handle electronic documents operations.

3.1. Conceptual Architecture

Let us consider the task of creating a reliable platform for the distributed storage of legally significant electronic documents. This platform should be efficient in terms of access speed (first of all, reading) and resources (disk space) utilization. Additionally, the platform must scale without loss of speed, through the use of cloud computing. As noted above, the approach of combining two technologies—DLT and distributed file storage (e.g., P2P, torrent, etc.)—looks promising.

An electronic document consists of the following:

- Metadata (as a rule, this is props and technical information in XML format).
- Files associated with the document (in text or binary formats—DOCX, PDF, etc.).
- Links to other documents (a link type and link, or identifiers to other documents are specified).
- Electronic signature—the result of signing with the private key (as a rule, it contains the result of encryption with the private key in the form of hash, and the public key, which is necessary for decryption).

In this way, all document metadata, links between documents, linked file metadata, and electronic signature components may be stored on a blockchain. The files themselves may be stored in a distributed file repository (Figure 1).

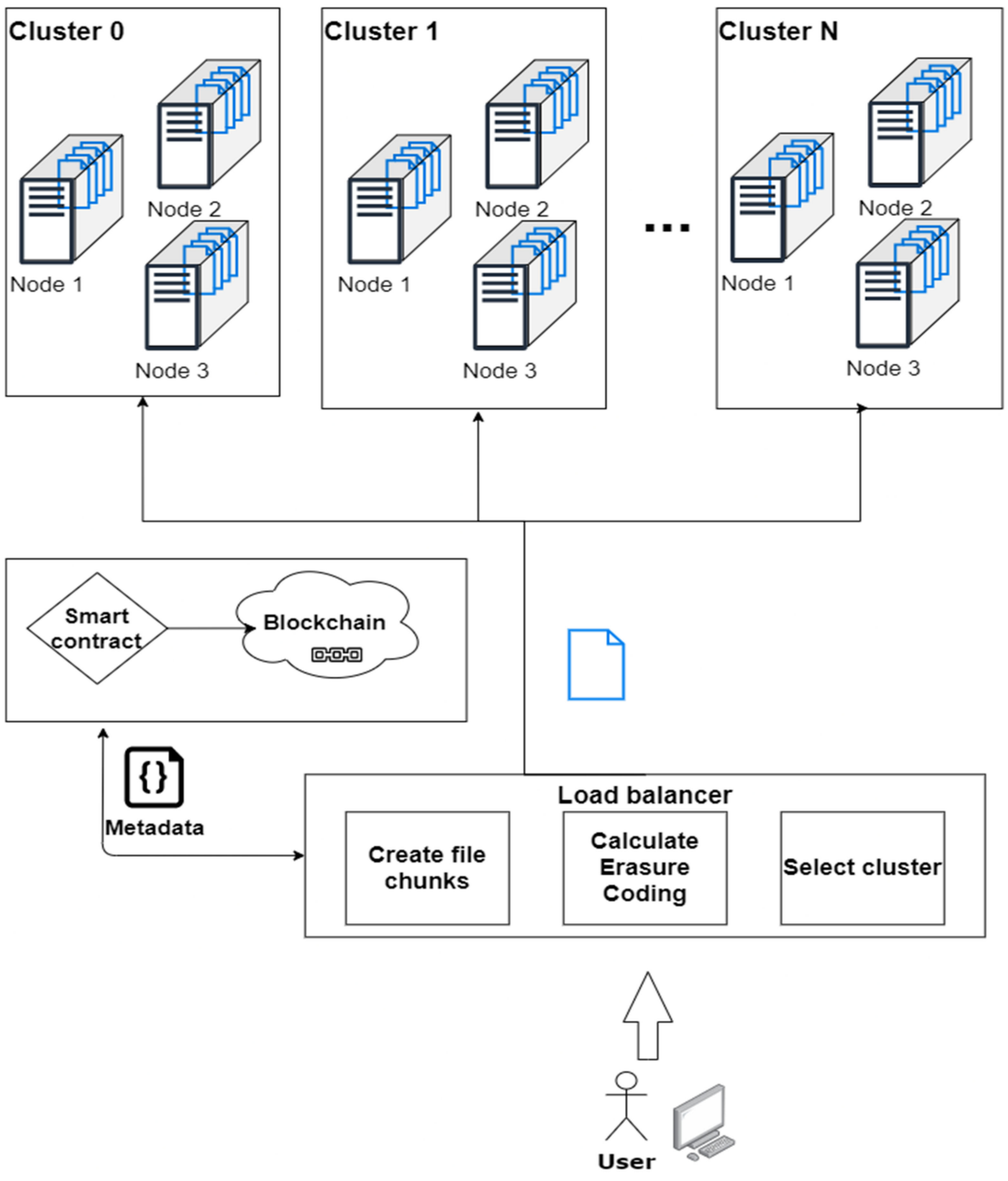

Figure 1. Conceptual architecture of the electronic document exchange system.

When a user requests a file, the file data is checked for integrity and a user's rights to access it. When a new version of a file is uploaded, its new metadata is filled in and a record of the file is added to the blockchain.

3.2. Block-Based File Storage

The system under consideration may store various types of files, including binary files of large size. To solve the problem of optimizing storage space, each file should be divided in half. Then it could be stored on two servers, and a well-known Erasure coding approach may be applied [18]. In the case of storing large files, those could be divided into blocks to increase the synchronization speed. However, splitting files into large blocks limits I/O operations. In case of network errors, it is necessary to resume loading from the last block.

Thus, we need to provide an option for indexing the files (for example, using their path) and for storing information about the files. To achieve this, we propose the following file placement structure in the DLT (Table 1), so that we can subsequently operate on it through a smart contract. Each node stores information about the files it stores, the hash of the file in the whole system, and the path to it on the given server.

Table 1. Structure of the file storage table in the system.

Index	Hash	Timestamp	Creator	Cluster
/var/data/myfile1.pdf	29ce8a685d966da960645068b 0b36c99eb2ae3c2	1613824311	0xacF5dc45172202D2Ed9721AA 48030A9d0FB0f0b8	1
/var/data/myfile2.docx	46303c376da30655cd481db2f 7accdc0e7b617d6	1613029375	0xacF5dc45172202D2Ed9721AA 48030A9d0FB0f0b8	2

There is also a common array in the DLT with information about the blocks within each cluster (see Table 2).

Table 2. Location of the files' parts.

Index	Node	Part
/var/data/myfile1	0	1
/var/data/myfile1	1	2
/var/data/myfile1	2	0
/var/data/myfile2	2	1

In Table 2, the Node is the node ID, and the Part specifies which part of the file is stored in the node. In the example from Table 2, myfile1 has parts 1 and 2. Part = 0 means that this location stores its erasure code.

Erasure code has several implementations [18]. Here we consider the simplest form, which works by calculating the XOR of two files. Suppose we have two files: file1 and file2, and they have the same length (L) (length in bits) (size). The XOR of both files is a third file (file3), which has the same length (L). Now suppose that each of these files is on a different server.

If the server containing file1 fails, this file can be computed from files file2 and file3 using XOR table. Thus, instead of saving a complete duplicate of both files (which would result in a total length of 4L), only one file would be saved instead of two duplicates, so the total length would be 3L. This means that we can save 25% of the total space needed on all the servers in our system.

3.3. File Versions Management

It is logical to assume that the file storage used to store electronic documents should support file versioning. Updating a file means adding a new version of that file, not replacing the contents of an existing file. When deleting/updating a file, its record is not deleted but moved to another array (OldFiles). This array has the same structure as files but additionally stores the date of insertion into the table.

Since different versions of binary files may require a large amount of storage, old versions may be deleted when two conditions are met:

- The version is older than a certain amount of time defined in the blockchain network (e.g., a year).
- This is not the latest version of the file.

When these two conditions are combined, storage space is not wasted on unnecessary files, and if an authorized user for some reason decides to destroy a file by adding several new versions, it will not be permanently deleted. The same storage rules described in Section 3.2 apply to each version of a file.

3.4. File Deletion

The file deletion operation should result in the complete deletion of the file. There must be a reliable way to ensure that if someone hacked a user with editing rights, they would not be able to remove files from the system. This goal can be achieved by adding a new array for the files to be deleted, with the structure of Table 3.

Table 3. Deleted files table.

File ID	Timestamp	User ID	Decision
47	1613029375	13	1
47	1613029375	28	1
391	1203393411	14	1

Therefore, instead of deleting a file, its identifier gets inserted into the table, and a temporary deletion mark is recorded.

In addition, all users in the system who already have access to the file are notified about it (possibly by a GUI). In this way, users who do not approve of the operation can vote against its deletion.

3.5. Downloading and Updating Files Algorithm

An important element of the proposed methodology is the implementation of a control algorithm for splitting files into blocks and accounting for stored blocks. Blockchain supporting smart contracts [8] can be used to determine in which cluster the necessary data will be located.

1. When adding or updating a file, the client application calls the load balancer, which polls existing clusters and finds out where the most free space is or which one has the fastest connection speed to the current application (the selection algorithm can be parameterized on the server-side in the cluster).
2. Next, the balancer, having identified the desired cluster id, calls the AddFile method of the smart contract.
3. Next, the balancer side calculates the erasure code and splits the file into blocks.
4. Then the first cluster server writes the first block to the predefined path formed by the file id (for example, /files/var/data/articles/248.pdf/1)
5. The second server, in a similar way, saves the second part of the file (for example, /files/var/data/articles/248.pdf/2)
6. The same thing happens when adding erasure code with part 0 to the third server.
7. When updating, the previous entries are moved to the same table of old versions—oldFiles
8. The conditions for the array oldFiles from point 1.3 are checked. If it is not fulfilled for some versions, then the files on the path are deleted, and these versions are removed from the array.
9. The data in the files array is updated via a smart contract, the state of the new transaction is recorded, and all DLT nodes are updated.

Figure 2 illustrates the process in the case of file retrieval.

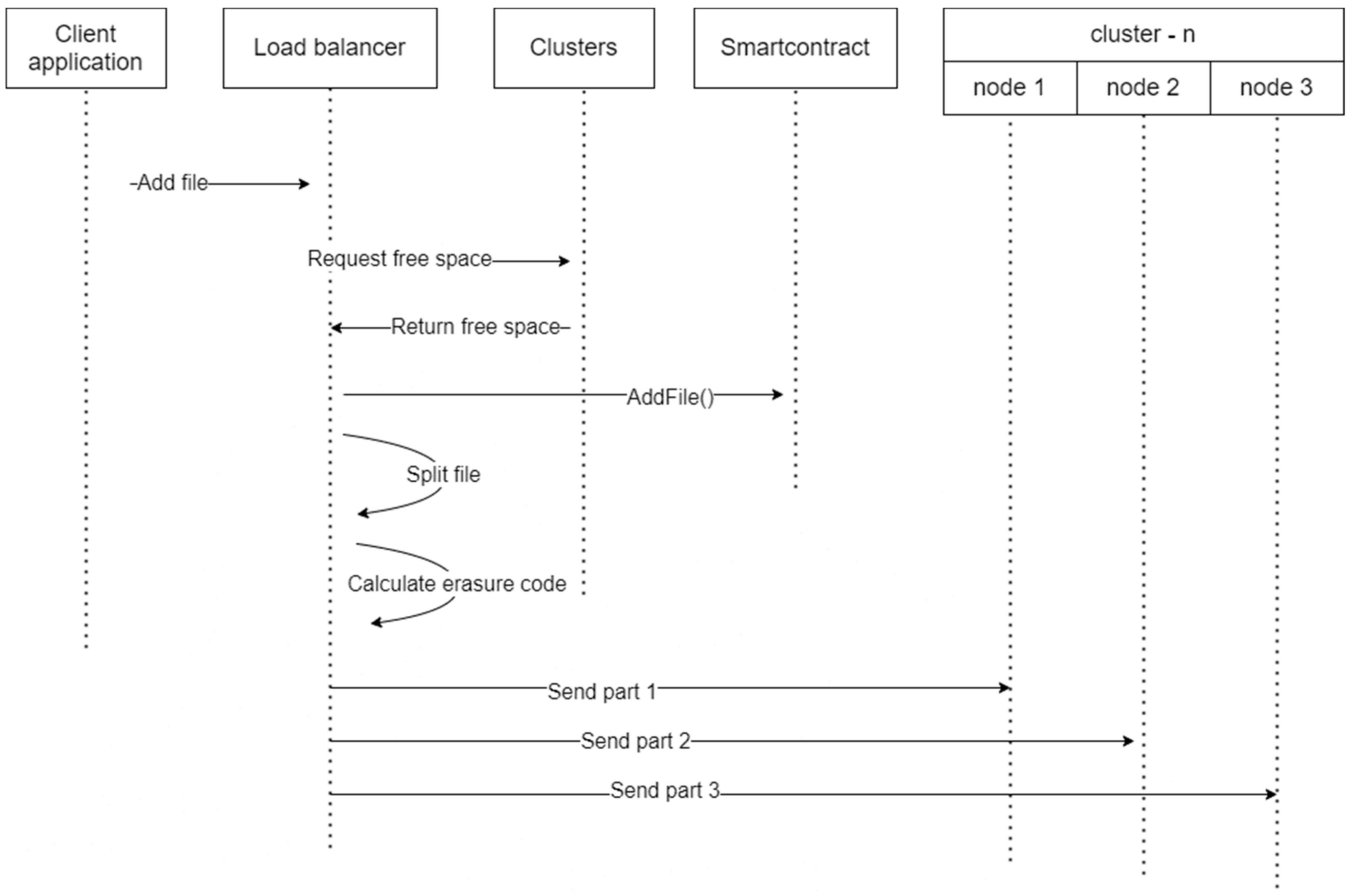

Figure 2. Requesting file algorithm.

The described algorithm has been implemented in two forms:

1. As a smart contract, which is written in Solidity 0.8.0 for Ethereum (Supplementary Material: https://github.com/Obadah-H/DistributedFileSystem/blob/main/Smartcontract/MainContract.sol—accessed on 2 November 2021), and web3.js was used to connect with Ethereum, with the help of an Infura node.

2. As a Go chaincode on Hyperledger Fabric (Supplementary Material: https://github.com/Obadah-H/DistributedFileSystem/blob/main/Smartcontract/Hyperledger/smartcontract.go—accessed on 2 November 2021). The Hyperledger Fabric node was deployed on a virtual server.

As seen in Figure 2, the end-user does not directly access the smart contract. To do so, one needs to implement a special interface (e.g., through a REST API web service) on the server where the blockchain node is running, and to address it. This is a well-known feature of current DLT implementations [19].

4. Reliability Calculation

In this section, we calculate the reliability of the proposed system, and we compare it with the reliability of full-redundancy systems. To assess the reliability of the proposed conceptual architecture using erasure code, it is necessary to calculate the reliability scheme [20].

Each element (component) of our system is located on a separate server. Thus, the file can be retrieved in one of the following ways:

- Through part 1 and part 2 of the file;
- Through erasure coding and part 1 of the file;
- Through erasure coding and part 2 of the file.

The OR case is calculated as follows:

$$R = 1 - [(1 - R_1)(1 - R_2) \ldots (1 - R_n)] \tag{1}$$

where R is the reliability of each component in parallel. The AND case is calculated as

$$R = R_1 \times R_2 \times \ldots \times R_n \tag{2}$$

If we have the following case, then this is the AND case for R_1 and R_1. Therefore, $R = R_{12}$. As this is the same component, so R_{12} could be replaced by R_1. Thus, the overall reliability of the system can be calculated as

$$\begin{aligned} R &= 1 - (1 - R_1 \times R_2) \times (1 - R_1 \times R_3) \times (1 - R_2 \times R_3) = \\ &= R_2 \times R_3 + R_1 \times R_3 + R_1 \times R_2 - 2 \times R_1 \times R_2 \times R_3 \end{aligned} \tag{3}$$

Assuming that each component has a reliability of 0.9, this means that the overall reliability is equal to

$$R = 0.9 \times 0.9 + 0.9 \times 0.9 + 0.9 \times 0.9 - 2 \times 0.9 \times 0.9 \times 0.9 = 0.972 \tag{4}$$

Solutions based on full duplication of all components of the system are definitely the most reliable solution. However, they are not cost-effective (if eight servers are required to deploy the system, then another eight servers are required for backups).

In this case, the total reliability of one component will be equal to

$$R = 1 - (1 - R_1)(1 - R_2) = 0.99 \tag{5}$$

Reliability calculation in a scheme without backup servers

$$R = R_1 \times R_2 = 0.81 \tag{6}$$

Therefore, the reliability calculation with full redundancy

$$R = [1 - (1 - R_1)(1 - R_1')] \times [1 - (1 - R_2)(1 - R_2')] = 0.9801 \tag{7}$$

Figure 3 shows the reliability scheme of the proposed system against full-backup systems. In the proposed scheme, R_1 is the server where the first part of the file is saved, R_2 is the server where the second part of the file is saved, R_3 is the Erasure coding server. In the full-backup system scheme, R_1 and R_2 are files storage servers, while R_1' and R_2' are the servers where the backup is stored.

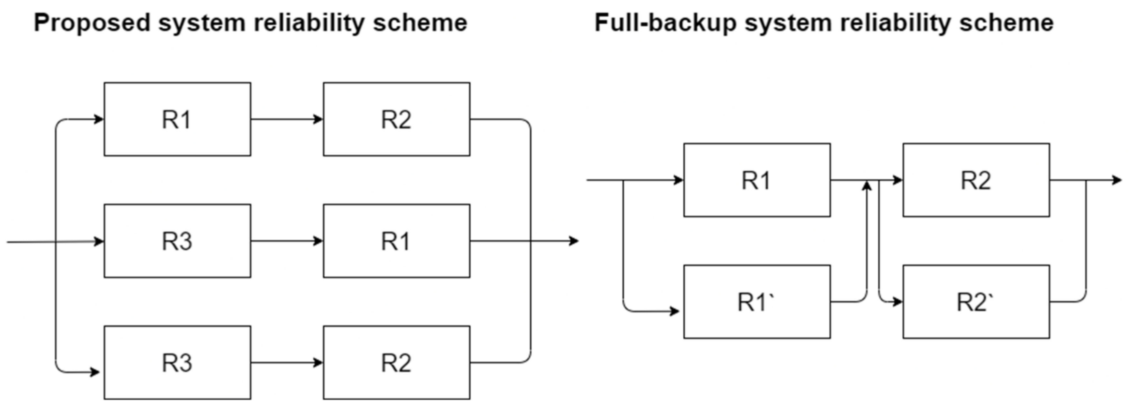
Proposed system reliability scheme
Full-backup system reliability scheme
R1
R2
R3
R1
R3
R2
R1
R2
R1`
R2`

Figure 3. Proposed and full-backup systems reliability schemas.

To sum up, the proposed approach has a reliability of 0.972 instead of 0.9801 (in the case of full duplication). The calculation results are quite comparable, but the total amount of required storage space would be 25% less.

5. Performance Test Results

To test the approach outlined above, we deployed a testbed on the local network, which includes a balancer server (see Section 3.1), a cluster of three virtual file machines, and a 100 Mbit/s network.

The purpose of the experiment is to choose a suitable blockchain network which performs the best under the proposed requirements, and to analyze the performance of the system. To do that, a basic system which handles files uploading and downloading was designed. When uploading a file, it calculates its hash, sends it to the blockchain network, and then splits it into two halves and calculates erasure code. After that, it uploads each half (and the erasure code) into a server. This experiment was repeated on three blockchain networks: Hyperledger, Ethereum Enterprise (Hyperledger Besu), and Ethereum Ropsten (a public Ethereum network).

Hyperledger and Ethereum Enterprise seem to be the most suitable blockchain solutions for implementing our approach, for the following reasons:

- These platforms are designed to create corporate solutions in the b2c model;
- There is no need to use cryptocurrency;
- Availability of smart contracts and their programming language—Solidity (Ethereum Enterprise) or Chaincode (Hyperledger);
- Support for confidential transactions (the ability to configure access to certain transactions for certain users).

The specific blockchain architecture was hosted in Yandex Cloud service. A virtual server (2 GB RAM, 2 vCPU Intel Xeon Gold 6230, 13 GB HDD) was used for Hyperledger, with 30 Mbit/s network bandwidth.

For Ethereum Enterprise, the Hyperledger Besu client was used on a virtual server (6 GB RAM, 2 Intel Xeon Gold 6230 vCPU, 20 GB), 30 Mbps network bandwidth.

The overall speed of the following tests was measured in this infrastructure:

- Upload—uploading from the balancer 1000 new documents, including XML with metadata and a 10 MB file with random content;
- Download—downloading 1000 documents (XML + files) through the balancer.

We measured the test start and end speed on the balancer in order to exclude client traffic delays from the measurements.

As can be seen from the results (Figure 4 and Table 4), both implementations are quite comparable in time with known p2p solutions running on torrent protocols. This is confirmed by performance tests [21,22].

Table 4. Hyperledger/Hyperledger Besu file transfer delay (sec).

Hyperledger upload test	min:	0.535	**Hyperledger Besu upload test**	min:	2.21
	max:	21.762		max:	12.214
	avg:	1.242023		avg:	3.203192
	median:	0.585		median:	3.294
Hyperledger download test	min:	0.27	**Hyperledger Besu download test**	min:	1.937
	max:	16.615		max:	26.012
	avg:	0.927483		avg:	2.986525
	median:	0.432		median:	2.902

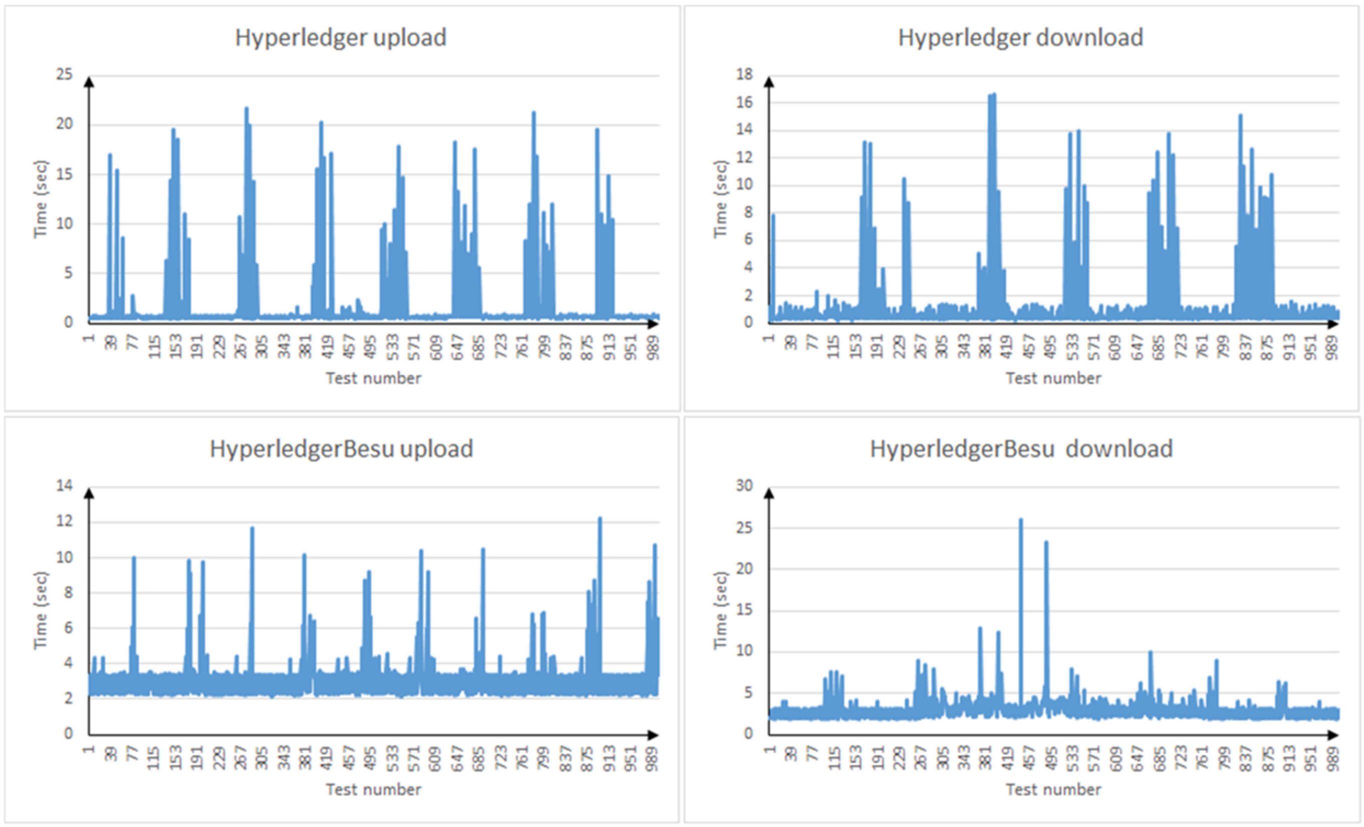

Figure 4. File upload/download delay in Hyperleger and Ethereum Enterprise (Hyperledger Besu).

In addition, a test was conducted using a smart contract placed in the public Ethereum test network Ropsten.

In the case of Ropsten (Figure 5 and Table 5), while the downloading files is quite fast, we can see that uploading files is several times slower. This is due to the load of the test network and its use of the PoW consensus algorithm, in contrast to Hyperleger Fabric [23] and Hyperledger Besu [24], which use less resource-intensive algorithms by default (PBFT or PoA). Primary problem of public DLT solutions that medium transaction processing time is too big [25,26]. Obtained results show that public blockchain networks are not suitable for creating such a system.

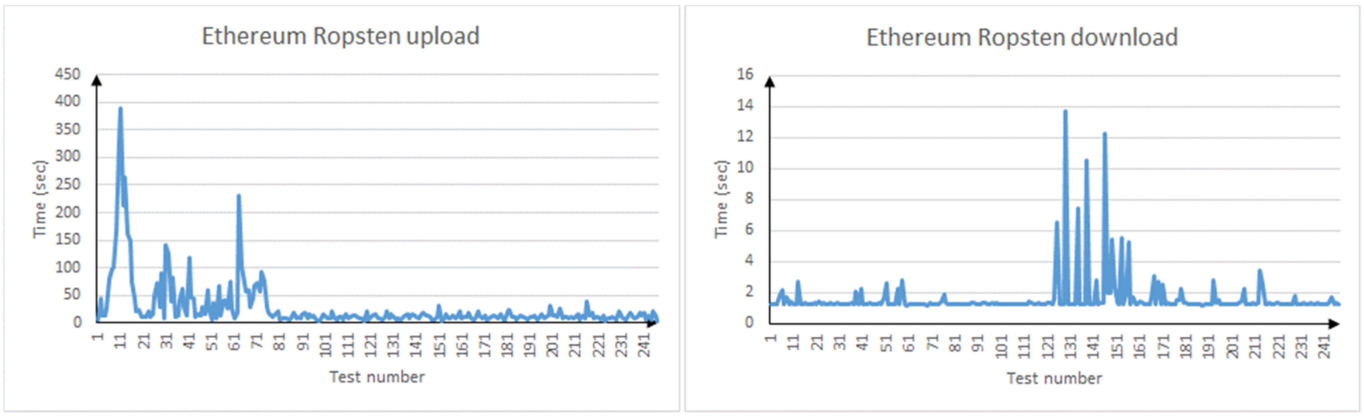

Figure 5. File upload/download delay in Ethereum Ropsten.

Table 5. Ropsten file transfer delay (sec).

Ropsten upload test	min:	3.512	**Ropsten download test**	min:	1.142
	max:	388.917		max:	13.673
	avg:	27.27899		avg:	1.597874
	median:	11.935		median:	1.258

Our experiment showed that such a system can be quickly scaled horizontally using inexpensive servers. At the same time, for storing files of 1000 e-documents, a total volume of no more than 15 GB is sufficient (10MB file + xml metadata in DLT). Thus, the system can store up to 1,000,000 electronic documents using relatively cheap nodes with a total capacity of 15 TB. For this purpose, it is possible to organize file storage on SAS disks with IOPS of 10,000 rpm.

6. Discussion

The ideas presented here to a certain extent follow the argument laid out in [27]. That paper also assumes the use of Erasure coding to optimize the storage location and Hyperledger Fabric as a blockchain. However, experimental results are missing from that paper, and the data is stored in a peer-to-peer network (not in a cluster). The authors of [28] also discuss document storage in the blockchain, and the 'bottleneck server' is used as an interface to communicate with the nodes. In their implementation; however, the nodes store a complete copy of the data, which is clearly redundant. The problem of storing files in the blockchain is not discussed, either.

The ideas discussed here are partly similar to the way CDN technology works [29]. CDN is widely used to download files by placing the most frequently used files (which are determined mainly using the LRU algorithm) on different servers in different geographical locations. For this reason, CDN can be considered a more expensive solution, which contributes to the speed of file downloads by browsers, rather than limiting the size of the storage.

IPFS [30] was studied as a file storage system in this study. It is considered by many researchers as a promising addition to DLT [4]. However, IPFS is built on the same principles as torrent solutions and is not suitable for companies in terms of network bandwidth, and the lack of backup methods. For example, unlike [13] in the proposed system, data access and processing are made transparent, as the end-user does not know about all the processes required to split files or to combine files' blocks, or how to get the file from the erasure code if a part is lost, etc. Additionally, unlike HyperBSA [15], in this research it is assumed that the system treats data equally, as there's only one type of data to be processed. Unlike DADS [14], our solution combines companies' servers with a blockchain network in order to store files, and it provides control over data accessing methods (the way data is updated or deleted, etc.) while minimizing the required storage space.

Compared to projects that use DLT to store IOT data [17], in our model, blockchain is used to securely manage files operations, and the storage nodes are the companies' servers, with a system designed specifically to prevent attacks and eliminate the possible actions if such attacks occur.

When calculations of the reliability of DLT-based systems are performed, they, as a rule, are not compared with traditional ones [31,32].

The main difficulties we encountered in the implementation process are as follows:

- System scaling: While the system is flexible regarding managing clusters, the process of adding a new server is not. A single server cannot be added (three servers should be added together).
- Choosing file storage location: If we assume that there are already three existed clusters, the proposed system should decide which cluster to use to store the new file based on defined rules.

- Managing storage capacity: In the proposed storage model, we assumed that all the servers in the same cluster have the same storage capacity. This assumption is not necessarily correct in real life. According to the way the proposed protocol works, the common storage capacity of a cluster is considered based on the server with the smallest storage capacity. An example would be: if a cluster contains three servers with the according storage capacity: 10 TB, 10 TB, 5 TB, the system will work as if each server has only 5 TB (the smallest capacity among them). That is because the file is split into two equal parts, and the Erasure coding has the same length of a single part of those parts. In other words, 10 TB would be wasted.

Clusters of the proposed repository can be horizontally scaled on the basis of cloud computing [33,34]—the blockchain nodes may be scaled, as well (taking into account the problems described above).

7. Conclusions

We have discussed the architecture of electronic document storage in DLT and distributed file structure. The main contribution of this research that we are first who combine enterprise DLT with distributed file storage, which can be scalable by applying cluster approach and in which the availability of files is controlled. Unlike other b2b solutions, we use DLT as primary storage for documents metadata and simultaneously, unlike b2c projects, proposed file storage where file availability is monitored and managed.

Considered DApps architecture reduces storage space by 25% using Erasure coding, what can be a significant step for the implementation of such a system in b2b. The results of performance testing allow us to draw conclusions about the required total amount of HDD volume and justify the use of inexpensive drives for this.

We were among the first to make an attempt to evaluate the reliability of DLT systems, in order to justify the feasibility of creating such a system in enterprise level. Here we did not try to create a new method for assessing reliability, but only tried to compare to traditional systems. Calculation of the reliability of such a system shows that, in general, it is more reliable than a centralized system and is not significantly inferior to systems with full duplication of the same major components (0.972 instead of 0.9801).

The experiment that we had undertaken showed that the results of using the most well-known blockchain platforms are quite acceptable for end-users. The implementation of the proposed approach on public networks, such as Ethereum, is difficult because of the unpredictable speed of transaction processing and redundant calculations in the PoW consensus operation. According to our results, Hyperledger can be considered as a good blockchain network for the required task, as it performs well compared with other networks, and it does not require cryptocurrency. We did not discuss the upload scenario in detail. In the upload test, no time is spent searching the index and it is assumed that the user already knows the index of the document and file needed. The implementation of the file and document retrieval scenario would be the task of our future research.

In addition, future work will be devoted to access control implementation, as well as testing and debugging the horizontal scaling algorithm of the described architecture. The considered DApps storage methodology easily scales horizontally in terms of distributed file storage, if we consider the whole system as a set of clusters of three servers (two main and one for storing erasure code). Therefore, the algorithm used to choose which cluster to save a new file can be updated. In the current model, choosing the cluster which has the most free space is used. Other parameters can become useful, such as network traffic, etc. Other possible file system related tasks must be considered—i.e., searching for files.

Finally, it should be noted that this article only describes the concept of our methodology. Additional functions of file and block encryption, integrity checking, and data transfer protocols are needed to create a reliable and secure system.

Supplementary Materials: The following smart contracts code is available online at https://github.com/Obadah-H/DistributedFileSystem/blob/main/Smartcontract/MainContract.sol, Code: Ethereum smart contract for managing filesystem processes, https://github.com/Obadah-H/DistributedFileSystem/blob/main/Smartcontract/Hyperledger/smartcontract.go, Code: Hyperledger smart contract for managing filesystem processes. Both GitHub links are mentioned in Section 3.5.

Author Contributions: Conceptualization, I.T. and O.H.; Methodology, I.T.; Software, O.H.; Validation, I.T. and A.K.; Writing—original draft preparation, O.H.; Writing—review and editing, A.K. All authors have read and agreed to the published version of the manuscript.

Funding: The article was prepared at the State Academic University for Humanities within the state assignment of the Ministry of Science and Higher Education of the Russian Federation (topic No. FZNF-2020-0014).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, Ivan Tarkhanov, upon reasonable request, as the number of data entries is large to fit in this section.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. FIPS PUB 161-2. Electronic Data Interchange (Supersedes FIPS PUB 161-1—19 April 1993). 1993. Available online: https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub161-2.pdf (accessed on 31 May 2021).
2. Smith, K. Implications of value added network services. *Data Process.* **1985**, *27*, 41–45. [CrossRef]
3. Kemp, B.; Olivan, J. European data format plus (EDF+), an EDF alike standard format for the exchange of physiological data. *Clin. Neurophysiol.* **2003**, *114*, 1755–1761. [CrossRef]
4. Golosova, J.; Romanovs, A. The Advantages and Disadvantages of the Blockchain Technology. In Proceedings of the IEEE 6th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), Vilnius, Lithuania, 8 November 2018; pp. 1–6.
5. Wu, K.; Ma, Y.; Huang, G.; Liu, X. A First Look at Blockchain-based Decentralized Applications. *arXiv* **2019**, arXiv:1909.00939.
6. Hamida, E.B.; Brousmiche, K.; Levard, H.; Thea, E. Blockchain for Enterprise. Overview, Opportunities and Challenges. In Proceedings of the ICWMC 2017, Nice, France, 23 July 2017.
7. Meng, Y.; Nazir, S.; Guo, J.; Uddin, I. A decision support system for the uses of lightweight blockchain designs for P2P computing. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 2708–2718. [CrossRef]
8. Johnston, D.; Yilmaz, S.O.; Kandah, J.; Bentenitis, N.; Hashemi, F.; Gross, R.; Wilkinson, S.; Mason, S. The General Theory of Decentralized Applications, DApps. 2014. Available online: https://cryptochainuni.com/wp-content/uploads/The-General-Theory-of-Decentralized-Applications-DApps.pdf (accessed on 31 May 2021).
9. Chen, Y.; Li, H.; Li, K.; Zhang, J. An improved P2P file system scheme based on IPFS and Blockchain. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 2652–2657.
10. Xu, J.; Figueiredo, R. GatorShare: A file system framework for high-throughput data management. In Proceedings of the HPDC'10, New York, NY, USA, 21 June 2010; pp. 776–786.
11. Li, J.; Jigang, W.; Chen, L. Block-secure. Blockchain based scheme for secure P2P cloud storage. *Inf. Sci.* **2018**, *465*, 219–231. [CrossRef]
12. Storj. A Decentralized Cloud Storage Network Framework. 2018. Available online: https://www.storj.io/storj.pdf (accessed on 31 May 2021).
13. Paik, H.Y.; Xu, X.; Bandara, H.D.; Lee, S.U.; Lo, S.K. Analysis of data management in blockchain-based systems: From architecture to governance. *IEEE Access* **2019**, *7*, 186091–186107. [CrossRef]
14. Altamimi, F.; Asif, W.; Rajarajan, M. DADS: Decentralized (Mobile) Applications Deployment System Using Blockchain: Secured Decentralized Applications Store. In Proceedings of the 2020 International Conference on Computer, Information and Telecommunication Systems (CITS), Hangzhou, China, 5–7 October 2010; pp. 1–8.
15. Chen, X.; Zhang, K.; Liang, X.; Qiu, W.; Zhang, Z.; Tu, D. HyperBSA: A High-Performance Consortium Blockchain Storage Architecture for Massive Data. *IEEE Access* **2020**, *8*, 178402–178413. [CrossRef]
16. Liang, W.; Fan, Y.; Li, K.C.; Zhang, D.; Gaudiot, J.L. Secure data storage and recovery in industrial blockchain network environments. *IEEE Trans. Ind. Inform.* **2020**, *16*, 6543–6552. [CrossRef]
17. Li, R.; Song, T.; Mei, B.; Li, H.; Cheng, X.; Sun, L. Blockchain for Large-Scale Internet of Things Data Storage and Pro-tection. *IEEE Trans. Serv. Comput.* **2019**, *12*, 762–771. [CrossRef]
18. Babu, B.S.; Krishnan, M.N.; Vajha, M.; Ramkumar, V.; Sasidharan, B.; Kumar, P.V. Erasure coding for distributed storage. an overview. *Sci. China Inf. Sci.* **2018**, *61*, 1–45.

19. Khan, S.N.; Loukil, F.; Ghedira-Guegan, C.; Benkhelifa, E.; Bani-Hani, A. Blockchain smart contracts. Applications, challenges, and future trends. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 2901–2925. [CrossRef] [PubMed]
20. Menčík, J. *Concise Reliability for Engineers*; Intech: Rijeka, Croatia, 2016; Chapter 16. [CrossRef]
21. Guo, L.; Chen, S.; Xiao, Z.; Tan, E.; Ding, X.; Zhang, X. A performance study of BitTorrent-like peer-to-peer systems. *IEEE J. Sel. Areas Commun.* **2007**, *25*, 155–169. [CrossRef]
22. Pawar, M.K.; Patil, P.; Sharma, M.; Chalageri, M. Secure and Scalable Decentralized Supply Chain Management Using Ethereum and IPFS Platform. In Proceedings of the 2021 International Conference on Intelligent Technologies (CONIT), Hubli, India, 25–27 June 2021; pp. 1–5. [CrossRef]
23. Cachin, C. Architecture of the hyperledger blockchain fabric. In Proceedings of the Workshop on Distributed Cryptocurrencies and Consensus Ledgers, Chicago, IL, USA, 25 July 2016; Volume 310.
24. Besu Enterprise Ethereum Client. Available online: https://besu.hyperledger.org/en/stable/ (accessed on 31 May 2021).
25. Zhang, L.; Lee, B.; Ye, Y.; Qiao, Y. Ethereum transaction performance evaluation using test-nets. In *European Conference on Parallel Processing*; Schwardmann, U., Wissenschaftliche, G., Eds.; Springer: Cham, Switzerland, 2020; Volume 11997, pp. 179–190. [CrossRef]
26. Ethereum Testnet. Available online: https://teth.bitaps.com/ (accessed on 8 October 2021).
27. Ali, S.; Wang, G.; White, B.; Cottrell, R. A Blockchain-Based Decentralized Data Storage and Access Framework for PingER. In Proceedings of the 2018 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), New York, NY, USA, 1 August 2018; pp. 1303–1308.
28. Ghanghoria, A.S.; Raja, A.S.A.; Bachche, V.J.; Rathi, M.N. Secure E-Documents Storage using Blockchain. *Int. Res. J. Eng. Technol. (IRJET)* **2020**, *7*, 1972–1974.
29. Peng, G. CDN: Content Distribution Network. *arXiv* **2004**, arXiv:cs/0411069.
30. Benet, J. IPFS—Content Addressed, Versioned, P2P File System. *arXiv* **2014**, arXiv:1407.3561.
31. Zheng, P.; Zheng, Z.; Chen, L. Selecting reliable blockchain peers via hybrid blockchain reliability prediction. *arXiv* **2019**, arXiv:1910.14614.
32. Metcalfe, W. Ethereum, Smart Contracts, DApps. In *Blockchain and Crypt Currency*; Springer: Singapore, 2020; pp. 77–93.
33. Wu, J.; Ping, L.; Ge, X.; Wang, Y.; Fu, J. Cloud storage as the infrastructure of cloud computing. In Proceedings of the 2010 International Conference on Intelligent Computing and Cognitive Informatics, Kuala Lumpur, Malaysia, 22 June 2010; pp. 380–383.
34. El-Gazzar, R.F. A literature review on cloud computing adoption issues in enterprises. In Proceedings of the International Working Conference on Transfer and Diffusion of IT, Aalborg, Denmark, 2–4 June 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 214–242.