Proceedings of the of the 17th Conference on Computer Science and Intelligence Systems pp. 563–566
DOI: 10.15439/2022F183
ISSN 2300-5963 ACSIS, Vol. 30

Increasing data availability and fault tolerance for decentralized collaborative data-sharing systems

Kamil Jarosz*, Łukasz Opioła†, Łukasz Dutka†, Renata G. Słota*, and Jacek Kitowski*†
* AGH University of Science and Technology,
Faculty of Computer Science, Electronics and Telecommunications,
Institute of Computer Science, Krakow, Poland
Emails: {kamil.jarosz, renata.slota, jacek.kitowski}@agh.edu.pl
† Academic Computer Centre CYFRONET AGH, Krakow, Poland
Emails: lopiola@agh.edu.pl, lukasz.dutka@cyfronet.pl

***Abstract*—In order to realize collaboration on a global scale, academic research requires often large quantities of data to be shared between geographically dispersed organizations. The requirement to protect and govern data in a network of loosely coupled, autonomous institutions is an incentive for decentralized solutions, where the participants are in full control of their data without trusting a third-party provider to store and process the data. In order to increase data availability and fault tolerance in decentralized collaborative systems, we propose a layer, which is based on replication and decentralized authority over the data. The solution consists of an idea of peer-sets, which are groups of peers implementing collective data management, a consensus protocol which synchronizes a distributed ledger between peers, and an atomic commitment protocol used to implement optional two-way references between documents. This architecture may be utilized in various decentralized collaborative data-sharing systems, such as Onedata.**

I. Introduction

DEMAND for global data access and data availability is growing due to increasing globalization and scale. It is especially evident in research, where often large quantities of data need to be shared upon request easily between geographically dispersed groups of people. High data availability and safety is usually a requisite in order to perform large scale computations. For instance, the eXtreme-DataCloud project [1] developed smart orchestration tools and building blocks of software, which provide storage federation with transparent or quasi-transparent data access for large and geographically distributed datasets. This project was dedicated to prominent research communities from various domains: LifeScience, Biodiversity, Clinical Research, Astrophysics, High Energy Physics, and Photon Science.

Collaboration between researchers may be backed by scientific organizations, which are autonomous and often do not have any written agreements between them. Each organization manages its own data, which may reside on various storage systems in different clouds. The requirement to protect and govern their data in a network of loosely coupled, autonomous institutions makes it difficult to constrain data sharing software to be centralized. Instead, decentralized solutions are more suitable in such cases, because the participants are in full control of their data and they may decide what is shared to whom, without trusting a third-party provider to store and process the data. Examples include ScienceMesh [2], which tries to connect existing storage systems and services using a common API, or Onedata [3], which delivers a service allowing to create a global network of independent storage providers.

High data availability in collaborative systems is crucial, but is also more difficult to accomplish in decentralized environments. When a party experiences a failure, all of its data becomes unavailable to others, whom parts of the data have been shared with. The problem is more complex, because not only may collaborators want to access the shared data, but also modify it—all of it during the outage of the original organization's infrastructure. These scenarios show a demand not only for decentralized architecture of data sharing systems, but also for decentralized authority over shared data, i.e. a possibility to modify it collectively by each one of its owners—collaboration between organizations often results in a joint work, and it is not uncommon for people to be parts of multiple organizations at once. In this paper, we propose an architecture of a layer of data-sharing systems, which ensures high availability and fault tolerance, and is based on a global, decentralized network of peers. The novelty of our proposal lies in dividing the global network into smaller peer-sets, which allow unrestricted global collaboration, at the same time limiting the flow of information between peers to the required minimum.

II. Related work

The concept of decentralized collaboration and data-sharing has been studied for years. Onedata is an important example, the goal of which is to achieve collaboration in global networks of autonomous organizations. It provides a service called Onezone, which implements a common layer between different storage providers and enables universal user authentication [4]. Other notable work include ScienceMesh [5], which provides an ecosystem—common interfaces and tools in order to connect existing heterogeneous services. Its goal is to allow collaboration using, for instance, share-with flows or data synchronization. It is relatively simple and does not implement more advanced solutions, such as complex organizational structures.

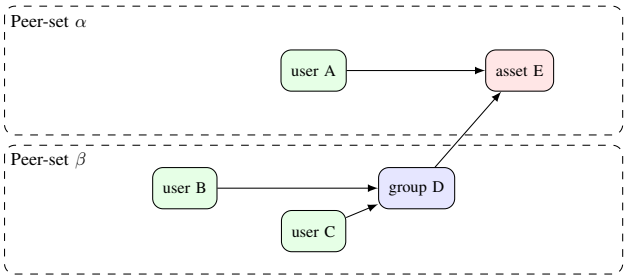

Fig. 1. Exemplary organizational structure spanned across two peer-sets.

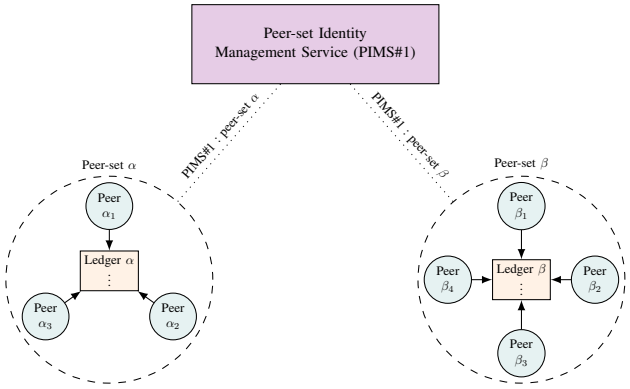

Fig. 2. Exemplary logical structure of peer-sets: each peer-set stores its own ledger, whereas PIMSes resolve peers inside peer-sets.

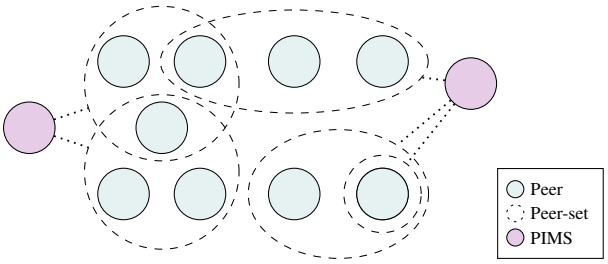

Fig. 3. Exemplary network consisting of PIMSes and peers divided into peer-sets; one peer may belong to an arbitrary number of peer-sets, peer-sets may consist of arbitrary numbers of peers.

It is possible to create a global network of independent peers using various DHT implementations, but they have inherent problems with security in trustless environments. Blockchain [6], on the other hand, ensures Byzantine Fault Tolerance and can be used both for decentralized storage [7], and for decentralized collaboration [8]. Blockchain is problematic due to its publicness—in situations where data is to be kept private its use is limited. In order to constrain possible participants and limit public access, permissioned blockchains may be used. However, they require a central authority responsible for authorization which undermines their decentralization. Irrespectively of the type of blockchain used, the primary issue is related to the commitment of organizations to handle a global chain and store data accessible to other blockchain participants regardless of their willingness to collaborate.

III. Our proposal

We propose a decentralized architecture of a layer of data-sharing systems with goals of increasing data availability and fault tolerance. It is achieved by utilizing decentralized authority over the data and replicating it over several peers. This section also includes a discussion on protocols and structures of the data used. We target the metadata used to describe organizational structures, users, groups, etc., but the architecture overall may be used for more generic document-like data with optional two-way references between them.

A. Architecture

We assume that the system stores and allows access to "objects", which represent arbitrary metadata. Objects may have relations between them, which are represented as two-way references. For instance, in case of organizational structures, an object may represent a user, a group, or an asset. Relations may represent memberships, in case of users and groups, or access grants, in case of assets (cf. Fig. 1).

Objects are decentralized by nature—each object is owned by some organization—a peer. For instance, an object representing a user may be owned by its university (an organization), along with his groups. Collaboration between different organizations happens when users belonging to each have common relations, e.g. they are both members of the same group.

We propose an idea of a "peer-set"—a group of peers which collectively manages a set of objects (cf. Fig. 2 and 3). This concept increases data availability and fault-tolerance, because objects are not managed in a centralized manner, but rather are managed collectively by multiple peers. This way, users with multiple affiliations may be naturally managed by a peer-set consisting of organizations they are part of. Groups and assets may be spanned across organizations which support the work they are used for, or they may be backed up to other peers provided by the organization. Every peer-set is a closed group, which may be created by a user, given permissions from each peer. We assume that every member of a peer-set is not malicious, and the peer-set creator trusts each peer to store and manage the data, thus we do not consider Byzantine faults.

In order to increase flexibility, our proposal also includes mutability of peer-sets. This allows users to dynamically add or remove a peer from the peer-set, which improves data availability.

B. Peer-set ledger structure and consensus protocol

A fundamental problem in distributed systems is coordinating multiple peers to agree on some common value, i.e. reach *consensus*. Consensus protocols—created to solve this problem—have to be fault tolerant, as an arbitrary set of participating peers may fail at any moment. The conventional consensus protocols include Paxos [9] along with its modifica-

TABLE I
PEER-SET α LEDGER CONTENTS

#	**Peer-set α**
4	Add relation "group D from β"→"asset E"
3	Add relation "user A"→"asset E"
2	Add object "asset E"
1	Add object "user A"

TABLE II
PEER-SET β LEDGER CONTENTS

#	**Peer-set β**
6	Add relation "group D"→"asset E from α"
5	Add relation "user C"→"group D"
4	Add relation "user B"→"group D"
3	Add object "group D"
2	Add object "user C"
1	Add object "user B"

tions, such as Multi-Paxos [10] or EPaxos [11]. There are also alternatives, such as Raft [12] or Zab [13]. Some consensus protocols may also be categorized as Byzantine fault-tolerant, which imposes a weaker condition on the type of faults that may happen—instead of only assuming crashes, they allow an arbitrary failure, including malicious misbehaving, such as forging counterfeited messages. Such consensus protocols include proof-of-work or proof-of-stake [14], both of which are used mainly in public blockchain networks [6]. There are also BFT versions of consensus protocols used outside of blockchain, such as PBFT [15].

In case of peer-sets, all peers inside them need to synchronize and decide on common state of the managed objects. Thus, we propose a peer-set history structure based on a distributed ledger, which consists of a list of incremental changes to the objects. The ledger is synchronized between peers using a consensus protocol based on examples mentioned above. Every peer may propose a change which extends the ledger by a new entry. Tables I and II depict exemplary ledger contents of respectively peer-set α and peer-set β, which are consistent with the state shown in figure 1. Solutions such as QLDB [16] may be used as an implementation for the ledger.

Our approach assumes not only data replication, but also a mechanism of data validation, where peers decide whether the common state is valid. This mechanism prevents one peer from suggesting a change that others may disagree with, and ensures that more than half of the peers accepted the change.

The proposed peer-set ledger structure is akin to a classic permissioned blockchain, however the global ledger is divided into multiple, separate ledgers maintained by a large number of peer-sets, which together participate in a global network.

C. Object and peer-set identification

One of the challenges in decentralized systems is object identification. A popular way of identifying objects in such environments is *content addressing*. This method is usually based on attaching a cryptographic hash function's digest of an object to its identifier. It addresses objects by their content instead of their location—it implies increasing data availability, but at the same time it restricts its modifications as every change generates a new identifier, and the old one becomes outdated. A well-known example is the BitTorrent protocol [17] which stores hashes inside torrent files or magnet links. Other uses include IPFS with CIDs [7], or Git with commit/tree/blob hashes [18].

When data is subject to change, content addressing becomes difficult to apply and other solutions are required. Standard identification methods, such as URLs, persistent identifiers, or handles [19], are content-agnostic, but in turn have fixed location, which undermines the decentralized nature of object storage and limits possibilities to migrate the data. Alternatives include *decentralized identifiers* (DIDs) [20], which are designed to identify and verify digital entities in decentralized web applications, and to provide a way of interacting with them. Blockchain also provides a possibility to store a pubic collection of peer-set identification records in a decentralized manner. However, all parties would have to agree to participate in a public network and to process the chain, which may discourage potential users.

Our architecture assumes both decentralized nature and possibility to modify data along with its owners. We propose a hybrid object identification system, which is based on a decentralized network of *peer-set identity management services* (PIMS), which are responsible for providing and managing information about peer-sets. An identifier to an object consists of three parts: 1) PIMS identification, 2) peer-set identification, and 3) object identification. The first part unambiguously identifies the service, which shares information about the peer-set upon request, using the second part of the identifier. This way, the user of the identifier may learn about the current state of the peer-set, without the need to change the identifier or waive the decentralized nature of the system (cf. f2). PIMSes are meant to be provided by organizations and create a decentralized network themselves.

Peer-set identity management services also have to implement specific identity management policies. Peer-sets are designed to be self-governing, so actions which modify a peer-set (such as adding a new peer, or removing an existing one) need to be established by the peer-set itself. Such requirement makes managing the data dependent on the data itself, which additionally justifies creating a dedicated service. For instance, adding a new peer D to a peer-set consisting of peers A, B, and C, may require approvals of D and majority of $\{A, B, C\}$.

D. Atomic commitment between peer-sets

Our proposal also takes into account atomic commitment between peer-sets. It may be used for instance to implement two-way relations between objects originating from different peer-sets. The atomic commitment ensures that two different peer-sets either both commit a change, or both ignore it (cf. entry 4 in table I and entry 6 in table II). We take into

consideration two ways of ensuring atomic commitment in our proposal.

Let us assume an atomic commitment between peer-sets A, B, and C, each one consisting of an arbitrary number of peers denoted A_0, A_1, etc. The first approach is to treat $\{A, B, C\}$ as one single database divided into three shards A, B, and C. Each shard contains a number of replicas, which are represented by peers. Using this interpretation we can use existing atomic commitment protocols which are designed for shards of replicas [21], but requires cooperation with the consensus protocol.

The second approach is to treat each peer-set as an entity and synchronize commitment between two representative peers. A special entry is added to the ledger, which represents an ongoing atomic commitment between peer-sets. This entry is used as a way of synchronizing state between all peers inside each of the peer-sets, and is used to implement a higher-level commitment protocol, which does not have to assume multiple replicas. Examples of such protocols include two-phase commit protocol (2PC), along with variations and improved versions such as 2PC* [22].

IV. Future work

The problem of increasing data availability and fault tolerance in decentralized collaborative systems is very broad and complex. This publication aims to designate a vision of our solution, which is based on replication and decentralized authority over data, and outline directions of its development. We propose an architecture of a layer of data-sharing systems, consisting of

- peer-sets, which implement a decentralized mechanism of collaboration between peers,
- peer-set identity management services, which allow discovery and identification of peer-sets, and
- atomic commitment protocol, which enables performing atomic changes between peer-sets in order to implement e.g. two-way relations between objects.

Discussions in section III are entry points to more detailed research in each subject. We plan to evaluate several consensus protocols in terms of usability and integrate them with the idea of voting and validating changes by peers. In case of atomic commitment, we want to implement and compare the two proposed solutions. Finally, we plan to create a proof-of-concept of the whole system along with detailed description and evaluation. We also intend to integrate our solution with Onedata by providing an integration layer with GraphSync—the metadata synchronization protocol.

V. Acknowledgements

This scientific work was published in part by an international project co-financed by the program of the Minister of Science and Higher Education entitled “PMW” in the years 2021–2023; contract No. 5193/H2020/2021/22. KJ, RGS and JK are grateful for support from the subvention of Polish Ministry of Education and Science assigned to AGH University of Science and Technology.

References

[1] D. Cesini *et al.*, “The eXtreme-DataCloud project solutions for data management services in distributed e-infrastructures,” *EPJ Web of Conferences*, vol. 245, p. 04010, 01 2020. doi: 10.1051/epjconf/202024504010

[2] ScienceMesh. [Online]. Available: https://sciencemesh.io/

[3] M. Wrzeszcz, Ł. Dutka, R. G. Słota, and J. Kitowski, “New approach to global data access in computational infrastructures,” *Future Generation Computer Systems*, vol. 125, pp. 575–589, 2021. doi: 10.1016/j.future.2021.06.054

[4] L. Opioła, L. Dutka, R. G. Słota, and J. Kitowski, “Trust-driven, decentralized data access control for open network of autonomous data providers,” in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, 2018. doi: 10.1109/PST.2018.8514209 pp. 1–10.

[5] Arora, Ishank, Alfageme Sainz, Samuel, Ferreira, Pedro, Gonzalez Labrador, Hugo, and Moscicki, Jakub, “Enabling interoperable data and application services in a federated sciencemesh,” *EPJ Web Conf.*, vol. 251, p. 02041, 2021. doi: 10.1051/epjconf/202125102041

[6] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, “Untangling blockchain: A data processing view of blockchain systems,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 7, pp. 1366–1385, 2018. doi: 10.1109/TKDE.2017.2781227

[7] J. Benet, “IPFS - content addressed, versioned, P2P file system,” *arXiv preprint arXiv:1407.3561*, 2014. doi: 10.48550/arXiv.1407.3561

[8] D. Ulybyshev, M. Villarreal-Vasquez, B. Bhargava, G. Mani, S. Seaberg, P. Conoval, R. Pike, and J. Kobes, “(WIP) Blockhub: Blockchain-based software development system for untrusted environments,” in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018. doi: 10.1109/CLOUD.2018.00081 pp. 582–585.

[9] M. Pease, R. Shostak, and L. Lamport, “Reaching agreement in the presence of faults,” *J. ACM*, vol. 27, no. 2, p. 228–234, apr 1980. doi: 10.1145/322186.322188

[10] L. Lamport, “Paxos made simple,” *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58, 2001.

[11] I. Moraru, D. G. Andersen, and M. Kaminsky, “There is more consensus in egalitarian parliaments,” in *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, ser. SOSP ’13. New York, NY, USA: Association for Computing Machinery, 2013. doi: 10.1145/2517349.2517350. ISBN 9781450323888 p. 358–372.

[12] D. Ongaro and J. Ousterhout, “In search of an understandable consensus algorithm,” in *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIX ATC’14. USA: USENIX Association, 2014. ISBN 9781931971102 p. 305–320.

[13] F. P. Junqueira, B. C. Reed, and M. Serafini, “Zab: High-performance broadcast for primary-backup systems,” in *2011 IEEE/IFIP 41st International Conference on Dependable Systems Networks (DSN)*, 2011. doi: 10.1109/DSN.2011.5958223 pp. 245–256.

[14] P. R. Nair and D. R. Dorai, “Evaluation of performance and security of proof of work and proof of stake using blockchain,” in *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, 2021. doi: 10.1109/ICICV50876.2021.9388487 pp. 279–283.

[15] M. Castro and B. Liskov, “Practical byzantine fault tolerance,” in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI ’99. USA: USENIX Association, 1999. ISBN 1880446391 p. 173–186.

[16] Amazon Quantum Ledger Database (QLDB). [Online]. Available: https://aws.amazon.com/qldb/

[17] The BitTorrent protocol specification. [Online]. Available: https://www.bittorrent.org/beps/bep_0003.html

[18] Git. [Online]. Available: https://git-scm.com/

[19] Handle.Net Registry. [Online]. Available: https://www.handle.net/

[20] World Wide Web Consortium. Decentralized identifiers (DIDs) v1.0. [Online]. Available: https://www.w3.org/TR/did-core/

[21] S. Maiyya, F. Nawab, D. Agrawal, and A. E. Abbadi, “Unifying consensus and atomic commitment for effective cloud data management,” *Proc. VLDB Endow.*, vol. 12, no. 5, p. 611–623, jan 2019. doi: 10.14778/3303753.3303765

[22] P. Fan, J. Liu, W. Yin, H. Wang, X. Chen, and H. Sun, “2PC*: A distributed transaction concurrency control protocol of multi-microservice based on cloud computing platform,” *J. Cloud Comput.*, vol. 9, no. 1, jul 2020. doi: 10.1186/s13677-020-00183-w