Byzantine-tolerant Distributed Grow-only Sets: Specification and Applications

Vicent Cholvi[1], Antonio Fernández Anta[2], Chryssis Georgiou[3], Nicolas Nicolaou[4], Michel Raynal[5,6], and Antonio Russo[2]

[1] Universitat Juame I, Spain
[2] IMDEA Networks Institute
[3] University of Cyprus, Cyprus
[4] Algolysis Ltd., Cyprus
[5] IRISA, France
[6] PolyU, Hong Kong

Keywords: Grow-only Sets · Distributed Ledgers · Blockchains · Atomic appends

1 Summary

Blockchains (as termed by Nakamoto in [14]) or Distributed Ledger Technologies (DLTs) (as used in [7] and [15]) became one of the most trendy data structures following the introduction of crypto-currencies [14] and their recent application in finance and token-economy. Despite their early wide adoption, little was known initially about the fundamental construction and semantic properties of DLTs. A number of research groups attempted to provide rigorous definitions to characterise the fundamental properties of DLTs as those used in Bitcoin and beyond [1, 7, 8]. Among those, Fernández Anta et al. [7], was the first to identify and provide a formal definition of a reliable concurrent object, termed *Distributed Ledger Object* (DLO), which conveys the essential building block for many DLTs.

With a growing amount of works dedicated to the Distributed Ledger formalization, it was shown in [10] that cryptocurrencies do not need consensus to be implemented. From a theoretical point of view, it was shown in [9] that, assuming one process per account, the consensus number of cryptocurrencies is 1. A non-sequential specification of money transfer was introduced in [2]. It follows that Byzantine transactional systems do not necessarily need consensus, but rather can be implemented on top of less powerful data structures. That's one of the reason why we focused on providing a lighter tool that gets rid of the strict ordering information of the elements that are added to the system.

In the DLO context, the introduction of many different DLT systems have led multiple studies [3, 6, 11, 12] to investigate the possibility of DLT interoperability, i.e., the ability for an action to be applied over a set of DLTs, rather than in a single DLT at a time. Using the DLO formalism, [6] introduced the *Atomic Appends problem*, in which several clients have a "composite" record (a set of semantically-linked "basic" records) to append. Each basic record has to

be appended to a different DLO, and it must be guaranteed that either all basic records are appended to their DLOs or none of them is appended.

In the work presented in [6], the authors assumed that clients may fail by crashing and showed that for some cases the existence of an intermediary is necessary. They materialized such an intermediary by implementing a specialized DLT, termed *Smart* DLO (SDLO). Using the SDLO, the authors solved the Atomic Appends problem in a client competitive asynchronous environment, in which any number of clients, and up to f servers implementing the DLOs, may crash. A subsequent work solved the problem assuming Byzantine failures [3], by introducing the notion of *Byzantine Distributed Ledger Objects* (BDLO). Solutions for implementing BDLOs were presented, with each solution relying on an underlying Byzantine Total-order Broadcast Service (BToB) [4, 5, 13]. Using BToB and an intermediary SBDLO the authors demonstrated how Atomic Appends may be achieved in systems that suffer Byzantine failures. However, BToB is a strong primitive, and requires consensus to be solved. So one may ask: *Is it possible to implement Atomic Appends without solving consensus?*

Similarly to the crypto-currencies finding, in this work, we observe that intermediary S(B)DLOs and strong primitives like BToB [13], may not be necessary to allow interoperability between multiple DLOs. Note that the goal of the intermediate S(B)DLO is to collect the records to be appended atomically, so that when all the records involved are in the S(B)DLO, then the actual records are appended in their respective DLOs. It is apparent that, for *Atomic Appends*, the order of the records in the intermediary data structure is not important, but rather the membership property required redirects to a *set* data structure.

A relevant distributed set data structure was presented by Shapiro et al. in [16] with the introduction of Conflict-Free Replicated Data Types (CRDTs). A CRDT is a data structure that can be replicated in multiple network locations. CRDTs have the property that each replica can be updated independently and concurrently, but it is always mathematically possible to resolve any inconsistencies between any pair of replicas, leading eventually all the replicas to a consistent converged value when the communication between the replica hosts is stabilized. A *Grow-Only Set* (G-Set) is such a CRDT, that supports operations add and lookup only. The add operation modifies the local state of the object by a union of the value of the set with the element we want to insert. Since add is based on union, and union is commutative, the G-Set implementation converges. In [16] (and other subsequent works), implementations of G-Sets where given in a crash-prone environment. In order to utilise a G-Set in more practical setups (like the ones in cryptocurrencies) we need to examine whether such data structure is possible when Byzantine failures are present in the system.

1.1 Contributions

In this work we examine whether G-Sets can be implemented when Byzantine processes are assumed in the system, without using consensus. We show that an implementation of an eventually consistent [17] G-Set is possible, and we demonstrate how such data structure can be used to solve Atomic Appends

and other related problems. In particular, our itemized contributions are the following:

- Provide a formal definition of a Byzantine Grow-only Set Object (BDSO).
- Provide an implementation for an eventually consistent BDSO in an asynchronous message passing system[7]. We consider such a consistency model since, although it provides weaker guarantees than other consistency models, it is easier and more efficient to implement, while being powerful enough to be used in the type of applications we consider (described next).
- Use BDSOs to implement:
  - Consensus-free Byzantine Atomic Appends.
  - Consensus-free Byzantine *Atomic Adds*. This is the analogous problem of atomic appends where records must be added in an atomic way to different BDSOs. This problem could be applicable in blockchain-like systems in which the ordering of the records is not important; what is important is that the records are added in the corresponding unordered blockchains (G-Sets). An example could be a system of G-Sets that implement personal calendars, so the records in the sets are meetings. Then, fixing a two-person meeting would imply an Atomic Add of the meeting data in the calendar of both persons.
  - Consensus-free single-writer BDLOs. This data structure can be suitable to implement whatever system that requires total order among data produced by a single writer. A punch in/out system for a company is an example of such an application in which a single writer, the employee, appends records only to his/her own ledger of presences. A cryptocurrency can be another suitable application, with one BDLO per account, because of the need to order transactions in relation to money transfers issued by the only transaction signer.

References

1. Anceaume, E., Pozzo, A.D., Ludinard, R., Potop-Butucaru, M., Tucci Piergiovanni, S.: Blockchain abstract data type. In: 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, Phoenix, AZ, USA, June 22-24, 2019. pp. 349–358. ACM (2019). https://doi.org/10.1145/3323165.3323183, https://doi.org/10.1145/3323165.3323183
2. Auvolat, A., Frey, D., Raynal, M., Taïani, F.: Money transfer made simple: a specification, a generic algorithm, and its proof. Bull. EATCS **132**, 23–43 (2020), http://eatcs.org/beatcs/index.php/beatcs/article/view/629
3. Cholvi, V., Fernandez Anta, A., Georgiou, C., Nicolaou, N., Raynal, M.: Atomic appends in asynchronous byzantine distributed ledgers. In: 2020 16th European Dependable Computing Conference (EDCC). pp. 77–84 (2020). https://doi.org/10.1109/EDCC51268.2020.00022
4. Coelho, P., Junior, T.C., Bessani, A., Dotti, F., Pedone, F.: Byzantine fault-tolerant atomic multicast. In: DSN 2018. pp. 39–50. IEEE (2018)

[7] Note that in such a system deterministic consensus can't be solved.

5. Cristian, F., Aghili, H., Strong, R., Dolev, D.: Atomic broadcast: From simple message diffusion to byzantine agreement. Information and Computation **118**(1), 158 – 179 (1995)
6. Fernández Anta, A., Georgiou, C., Nicolaou, N.: Atomic appends: Selling cars and coordinating armies with multiple distributed ledgers. In: International Conference on Blockchain Economics, Security and Protocols, Tokenomics 2019, Paris, France. pp. 39–50 (2019)
7. Fernández Anta, A., Konwar, K.M., Georgiou, C., Nicolaou, N.C.: Formalizing and implementing distributed ledger objects. SIGACT News **49**(2), 58–76 (2018)
8. Garay, J.A., Kiayias, A., Leonardos, N.: The bitcoin backbone protocol: Analysis and applications. In: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT 2015, Sofia, Bulgaria, April 26-30, 2015, Part II. pp. 281–310 (2015). https://doi.org/10.1007/978-3-662-46803-6_10, https://doi.org/10.1007/978-3-662-46803-6_10
9. Guerraoui, R., Kuznetsov, P., Monti, M., Pavlovic, M., Seredinschi, D.: The consensus number of a cryptocurrency. In: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019. pp. 307–316. ACM (2019). https://doi.org/10.1145/3293611.3331589, https://doi.org/10.1145/3293611.3331589
10. Gupta, S.: A non-consensus based decentralized financial transaction processing model with support for efficient auditing. Arizona State University (2016)
11. Herlihy, M.: Atomic cross-chain swaps. In: Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018. pp. 245–254 (2018)
12. Koens, T., Poll, E.: Assessing interoperability solutions for distributed ledgers. Pervasive and Mobile Computing **59**, 101079 (2019). https://doi.org/https://doi.org/10.1016/j.pmcj.2019.101079, https://www.sciencedirect.com/science/article/pii/S1574119218306266
13. Milosevic, Z., Hutle, M., Schiper, A.: On the reduction of atomic broadcast to consensus with byzantine faults. In: SRDS 2011. pp. 235–244 (2011)
14. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf (2008), [Online; accessed 22-February-2021]
15. Raynal, M.: Fault-Tolerant Message-Passing Distributed Systems - An Algorithmic Approach. Springer (2018). https://doi.org/10.1007/978-3-319-94141-7, https://doi.org/10.1007/978-3-319-94141-7
16. Shapiro, M., Preguiça, N., Baquero, C., Zawirski, M.: Conflict-free replicated data types. In: 13th International Symposium Stabilization, Safety, and Security of Distributed Systems, SSS 2011, Grenoble, France. pp. 386–400. Springer (2011). https://doi.org/10.1007/978-3-642-24550-3_29, https://hal.inria.fr/hal-00932836
17. Vogels, W.: Eventually consistent. Commun. ACM **52**(1), 40–44 (2009). https://doi.org/10.1145/1435417.1435432, https://doi.org/10.1145/1435417.1435432