Atomic Appends in Asynchronous Byzantine Distributed Ledgers*

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: Atomic Appends · Asynchrony · Blockchain · Byzantine process · Cooperation · Distributed Ledger Object, Synchronization

Acknowledgement This work has been partially supported by the Regional Government of Madrid (CM) grant EdgeData-CM – P2018/TCS4499 (cofunded by FSE & FEDER) and the Spanish Ministry of Science and Innovation grants ECID – PID2019-109805RB-I00 and PRX18/000163 (cofunded by FEDER), and the NSF of China grant 61520106005

1 Summary

There has been a great interest recently in the so-called crypto-technologies (e.g., blockchain systems [5]), and distributed ledger technology (DLT) in general [8], which are becoming very popular and are expected to have a high impact in multiple aspects of our everyday life. Although such a recent popularity is primarily due to the explosive growth of numerous crypto-currencies, there are many applications of this core technology that are outside the financial industry. These applications arise from leveraging various useful features provided by distributed ledgers, such as a decentralized information management, immutable record keeping for possible audit trail, robustness, availability, security, and privacy (see, for instance, [4, 6, 7]). However, there are many different DLT-based systems, and new ones are proposed almost everyday. Hence, it is extremely unlikely that a single DLT will prevail. This is forcing the DLT community to accept that it is inevitable to come up with ways to make DLTs interconnect and interoperate.

In that direction, a formal definition of a reliable concurrent object, termed *Distributed Ledger Object* (DLO), which endeavours to convey the essential elements of the many DLTs, was proposed in [3]. In particular, a DLO maintains

* Presented as Regular Paper at 16^{th} European Dependable Computing Conference (EDCC 2020)

a sequence of records, and has only two operations, APPEND and GET. The APPEND operation is used to add a new record at the end of the sequence, while the GET operation returns the whole sequence.

Using the above-mentioned formalism, the study of systems formed by multiple DLOs that interact among each other was pursued in [2], where the *Atomic Appends problem* was introduced. In this problem, 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 [2], the clients were assumed to be selfish and rational and could have different incentives for the different outcomes. Additionally, any client could fail by crashing. The authors 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.

1.1 Contributions

While [2] and [3] assume that clients and servers can only fail by crashing, several DLT-based systems assume both some servers (e.g., miners) and some clients (e.g., users) can act maliciously. To this respect, this article presents implementations where *both* the clients and the servers can be Byzantine, i.e., it presents implementations of *Byzantine-tolerant* linearizable DLOs. More precisely, the following contributions are presented.

- A formalization of the *Byzantine-tolerant Distributed Ledger Object*, in short BDLO.
- Algorithms (with their correctness proof) that implement a linearizable BDLO in an asynchronous setting (enriched with an underlying Byzantine Atomic Broadcast service) in which up to f servers can be Byzantine, and (i) an unbounded number of clients can be Byzantine, or (ii) only a bounded number t of clients can be Byzantine. In the second case it is possible to prevent spurious records to be appended by Byzantine clients without adding a specific mechanism.
- A definition of the *Atomic Appends* problem in a system with Byzantine failures.
- Algorithms (with their correctness proof) that combine BDLO implementations to solve the Atomic Appends problem. We provide two solutions.
  - Following the Smart DLO presented in [2] for process crash failures, a Smart version of BDLO (SBDLO) is first introduced, which aggregates and coordinates the append of multiple records. The SBDLO is implemented with a set N of $n \geq 2t+1$ servers up to which at most t can fail. The BDLOs on which the Atomic Appends is applied are implemented

as BDLOs with a bounded number t of Byzantine clients, so it is guaranteed that only if at least one correct process in N appends in them, the append takes place.

- Then, it is shown how the problem can be solved by replacing the SBDLO with a "classical" BDLO and the use of a set N of at least $2t+1$ "helper" processes, of which up to t can be Byzantine. These processes monitor (by periodically invoking GET operations) the BDLO for new Atomic Appends operations. Once "matching" Atomic Appends records are observed, the helper processes perform the APPEND operations to the corresponding BDLOs.

1.2 Experimental evaluation

In order to validate the adherence of our formalism to the currently available Blockchains and Distributed Ledger Technologies, we implemented the BDLO algorithms presented in our work. For the Byzantine Atomic (Total-order) Broadcast, we relied on the Tendermint [1] consensus protocol and its Golang SDK that easily allow to define custom messages' semantic for any application. Using a public cloud provider, we applied workloads with a growing number of clients and operations to BDLOs made up of 4, 7, and 10 servers. Finally we created an atomic append scenario in which 2, 3, and 4 different clients performed atomic append operations, through an SBDLO, targeting respectively 2, 3, and 4 different BDLOs. We compared the outcome in terms of overall throughput of the system to a similar problem setting solved using the HashLock/TimeLock solution and we showed how, in spite of the number of involved clients and BDLOs, the time to complete an atomic append operation remains constant in our approach while it grows linearly with the number of target BDLOs with the HashLock/TimeLock solution.

References

1. Buchman, E.: Tendermint: Byzantine fault tolerance in the age of blockchains. Ph.D. thesis (2016)
2. 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)
3. Fernández Anta, A., Konwar, K.M., Georgiou, C., Nicolaou, N.C.: Formalizing and implementing distributed ledger objects. SIGACT News **49**(2), 58–76 (2018)
4. Kuo, T.T., Kim, H.E., Ohno-Machado, L.: Blockchain distributed ledger technologies for biomedical and health care applications. Journal of the American Medical Informatics Association **24**(6), 1211–1220 (2017)
5. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf (2008), [Online; accessed 22-February-2020]
6. Namecoin: Namecoin. https://www.namecoin.org/, [Online; accessed 22-February-2020]

7. Spielman, A.: Blockchain: Digitally Rebuilding the Real Estate Industry. MS dissertation, Massachusetts Institute of Technology (2016)
8. Zago, M.G.: 50+ Examples of How Blockchains are Taking Over the World. Medium (2018), https://medium.com/@matteozago/50-examples-of-how-blockchains-are-taking-over-the-world-4276bf488a4b