Brief Announcement: Accountability and Reconfiguration – Self-Healing Lattice Agreement

Luciano Freitas de Souza ✉
CEA LIST, Université de Paris-Saclay, France

Petr Kuznetsov ✉
LTCI, Télécom Paris, Institut Polytechnique de Paris, France

Thibault Rieutord ✉
CEA LIST, Université de Paris-Saclay, France

Sara Tucci-Piergiovanni ✉
CEA LIST, Université de Paris-Saclay, France

Abstract

An *accountable* distributed system provides means to detect deviations of system components from their expected behavior. It is natural to complement fault detection with a reconfiguration mechanism, so that the system could heal itself, by replacing malfunctioning parts with new ones. In this paper, we describe a framework that can be used to implement a large class of accountable and reconfigurable replicated services. We build atop the fundamental lattice agreement abstraction lying at the core of storage systems and cryptocurrencies.

Our asynchronous implementation of accountable lattice agreement ensures that every violation of consistency is followed by an undeniable evidence of misbehavior of a faulty replica. The system can then be seamlessly reconfigured by evicting faulty replicas, adding new ones and merging inconsistent states. We believe that this paper opens a direction towards asynchronous "self-healing" systems that combine accountability and reconfiguration.

2012 ACM Subject Classification Theory of computation → Distributed algorithms

Keywords and phrases Reconfiguration, accountability, asynchronous, lattice agreement

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.54

Related Version *Full Version*: https://arxiv.org/abs/2105.04909

1 Introduction

There are two major ways to deal with failures in distributed computing:

Fault-tolerance: we anticipate failures by investing into replication and synchronization, so that the system's correctness is not affected by faulty components.

Accountability: we detect failures *a posteriori* and raise undeniable evidences against faulty components.

Accountability in computing has been proposed for generic distributed systems [13, 14] as a mechanism to detect deviations of system nodes from the algorithms they are assigned with. It has been shown that a large class of deviations of a given process from a given deterministic algorithm can be detected by maintaining a set of *witnesses* that keep track of all *observable* actions of the process and check them against the algorithm [15].

The generic approach can be, however, very expensive in practice and one may look for a more tractable *application-specific* accountability mechanism. Indeed, instead of pursuing the ambitious goal of detecting deviations from the assigned algorithm, we might want to only care about deviations that violate the specification of the problem the algorithm is trying to solve.

35th International Symposium on Distributed Computing (DISC 2021).
Editor: Seth Gilbert; Article No. 54; pp. 54:1–54:5
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Application-specific accountability. The idea has been successfully employed in the context of Byzantine Consensus [6]. The accountable version of consensus guarantees correctness as long as the number of faulty processes does not exceed some fixed f. But if correctness is violated, e.g., honest processes take different decisions, then at least $f+1$ Byzantine processes are presented with undeniable evidences of misbehavior. This is not surprising: a decision in a typical f-resilient consensus protocol must receive *acknowledgements* from a *quorum* of processes, and any two quorums must have at least $f+1$ processes in common [20]. The fact that two processes took different decisions implies that at least $f+1$ processes in the intersection of the corresponding quorums *equivocated*, i.e., acknowledged conflicting decision values. Assuming that every decision is provided with a cryptographic *certificate* containing the set of signed acknowledgements from a quorum of processes, we can immediately construct a desired evidence. Polygraph [6, 7], a recent accountable Byzantine Consensus protocol, naturally builds upon the classical PBFT protocol [5]. One may ask – okay, we have detected a faulty process, but what should we do next? Ideally, we would like to *reconfigure* the system by evicting the faulty process and *reinitializing* the system state.

Reconfigurable replicated systems [11, 12, 16, 22] allow the users to dynamically update the set of replicas. It has been recently shown that reconfiguration can be implemented in purely *asynchronous* environments [1, 2, 11, 16, 17, 22]. The idea was first applied to (read-write) storage systems [1, 2, 11], and then extended to max-registers [16, 22] and more general *lattice* data types, first in the crash-fault context [17] and then for Byzantine failures [18].

Contribution. In this paper, we propose a framework that can be used to implement a large class of replicated services that are both accountable and reconfigurable. Following recent work on reconfiguration [16–18], we build atop the fundamental *lattice agreement* abstraction. Lattice agreement [3, 9] takes arbitrary inputs in a *lattice* (a partially ordered set equipped with a *join* operator) and returns outputs that are (1) joins of the inputs, and (2) ordered with respect to the lattice partial order. Lattice agreement is weaker than consensus and can be implemented in an asynchronous system.

Lattice agreement (LA) appears to be a perfect match for both desired features: accountability and reconfiguration. Indeed, a quorum-based LA implementation enables detection of misbehaving parties: as soon as two correct users learn two incomparable values, they also obtain a proof of misbehavior of all replicas that *signed* both values. Furthermore, the very process of reconfiguration can be represented as agreement defined on a lattice of *configurations* [16, 17]. These two observations inspire the design of our system.

We propose an accountable *and* reconfigurable implementation that reaches agreement on a *joint* lattice: an object lattice (defining the current *state* of the replicated object) and a configuration lattice (defining the current *configuration* of the replicas). Assuming that the number of failures is less than half of the system size, our implementation is *alive*. It is also *safe* if only benign (crash) failures occur. Once safety is violated, i.e., two correct users learn two incomparable object states, some Byzantine replicas are inevitably confronted with an undeniable proof of misbehavior. The system is then seamlessly reconfigured by evicting the detected replicas, adding new ones and merging inconsistent states. Once the state is merged, the system comes back to providing safety and liveness, as long as no new replicas exhibits Byzantine behavior. Eventually all Byzantine replicas are detected and the system maintains liveness and safety.

Outdated configurations are harmless. Our system prevents users from accessing outdated configurations with the use of *forward-secure digital signature scheme* [4, 8]. A member of each new configuration is assigned a new secret key. Furthermore, honest members of an old configuration are expected to destroy their old keys before moving to a new one. Thus, if they are later compromised, they will not be able to serve clients' requests, and the remaining Byzantine replicas will not constitute a quorum.

On Byzantine clients. Our solution assumes that service replicas are subject to Byzantine failures, but clients are *benign*: they can only fail by crashing. This hypothesis has already been done in designs of fault-tolerant storage systems [19]. In our case, this assumption precludes the cases when a Byzantine client brings the system into a compromised configuration or slows down the system by issuing excessive reconfiguration requests. In the full version of this paper [10], we also describe a *one-shot* version of accountable lattice agreement, without reconfiguration, in which *both* clients and replicas can be Byzantine. Marrying reconfiguration and accountability in a long-lived service that can be accessed by Byzantine clients remains an important challenge.

Message. Altogether, we believe that this paper opens a new area of asynchronous "self-healing" systems that combine accountability and reconfiguration. Such a system either preserves safety and liveness or preserves liveness and compensates safety violations with eventual detection of Byzantine replicas. It also exports a reconfiguration interface that allows the clients to replace compromised replicas with new, correct ones. In this paper, we show that both mechanisms, accountability and reconfiguration, can be implemented in a purely asynchronous (in the modern parlance – *responsive*) way.

2 Reconfigurable and Accountable Lattice Agreement

A *reconfigurable accountable (long-lived) lattice agreement* (*RALA*) abstraction must ensure, among others, the following properties:

- **Completeness.** If a correct client learns a value that is incomparable with a value learnt by another correct client then it eventually accuses some new replicas.
- **Liveness.** If the system reconfigures only finitely many times, every value proposed by a correct client is eventually included in the value learned by every correct client.

The properties above imply that either the safety property of the implemented object holds (the values learnt by correct processes are comparable) or some new Byzantine replicas are eventually detected. If from some point on, no more Byzantine faults take place, we ensure that all new learnt values are comparable. Our requirement of finite number of reconfigurations is standard in the corresponding literature [2, 17, 22] and, in fact, can be shown to be necessary [21]. In practice, we ensure liveness in "sufficiently long" time intervals without reconfiguration.

Notice that the choice of new configurations to propose is left entirely to the clients, as long as one condition is satisfied: if the system is in a configuration which is not eventually replaced, than this configuration must contain a majority of correct replicas. It is important to emphasize that **the system does not allow the accused replicas to affect the system's safety and liveness**. Please refer to the full version [10] for the complete specification and the matching implementation.

References

1 Marcos Kawazoe Aguilera, Idit Keidar, Dahlia Malkhi, and Alexander Shraer. Dynamic atomic storage without consensus. *J. ACM*, 58(2):7:1–7:32, 2011.

2 Eduardo Alchieri, Alysson Bessani, Fabíola Greve, and Joni da Silva Fraga. Efficient and modular consensus-free reconfiguration for fault-tolerant storage. In *OPODIS*, pages 26:1–26:17, 2017.

3 Hagit Attiya, Maurice Herlihy, and Ophir Rachman. Atomic snapshots using lattice agreement. *Distributed Comput.*, 8(3):121–132, 1995.

4 Mihir Bellare and Sara K Miner. A forward-secure digital signature scheme. In *Annual International Cryptology Conference*, pages 431–448. Springer, 1999.

5 Miguel Castro and Barbara Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.

6 Pierre Civit, Seth Gilbert, and Vincent Gramoli. Polygraph: Accountable byzantine agreement. *IACR Cryptol. ePrint Arch.*, 2019:587, 2019. URL: `https://eprint.iacr.org/2019/587`.

7 Pierre Civit, Seth Gilbert, and Vincent Gramoli. Brief announcement: Polygraph: Accountable byzantine agreement. In Hagit Attiya, editor, *DISC*, volume 179 of *LIPIcs*, pages 45:1–45:3, 2020.

8 Manu Drijvers, Sergey Gorbunov, Gregory Neven, and Hoeteck Wee. Pixel: Multi-signatures for consensus. In *29th USENIX Security Symposium (USENIX Security 20)*, Boston, MA, August 2020. USENIX Association. URL: `https://www.usenix.org/conference/usenixsecurity20/presentation/drijvers`.

9 Jose Faleiro, Sriram Rajamani, Kaushik Rajan, Ganesan Ramalingam, and Kapil Vaswani. Generalized lattice agreement. In *PODC*, pages 125–134, 2012.

10 Luciano Freitas de Souza, Petr Kuznetsov, Thibault Rieutord, and Sara Tucci Piergiovanni. Accountability and reconfiguration: Self-healing lattice agreement. *CoRR*, abs/2105.04909, 2021. `arXiv:2105.04909`.

11 Eli Gafni and Dahlia Malkhi. Elastic configuration maintenance via a parsimonious speculating snapshot solution. In *DISC*, pages 140–153, 2015.

12 Seth Gilbert, Nancy A. Lynch, and Alexander A. Shvartsman. Rambo: a robust, reconfigurable atomic memory service for dynamic networks. *Distributed Comput.*, 23(4):225–272, 2010.

13 Andreas Haeberlen and Petr Kuznetsov. The Fault Detection Problem. In *Proceedings of the 13th International Conference on Principles of Distributed Systems (OPODIS'09)*, December 2009.

14 Andreas Haeberlen, Petr Kuznetsov, and Peter Druschel. The case for byzantine fault detection. In *Proceedings of the Second Workshop on Hot Topics in System Dependability (HotDep'06)*, November 2006.

15 Andreas Haeberlen, Petr Kuznetsov, and Peter Druschel. PeerReview: Practical accountability for distributed systems. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP'07)*, October 2007.

16 Leander Jehl, Roman Vitenberg, and Hein Meling. Smartmerge: A new approach to reconfiguration for atomic storage. In *DISC*, pages 154–169, 2015.

17 Petr Kuznetsov, Thibault Rieutord, and Sara Tucci-Piergiovanni. Reconfigurable lattice agreement and applications. In *OPODIS*, 2019.

18 Petr Kuznetsov and Andrei Tonkikh. Asynchronous reconfiguration with byzantine failures. In Hagit Attiya, editor, *DISC*, volume 179 of *LIPIcs*, pages 27:1–27:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

19 Jean-Philippe Martin, Lorenzo Alvisi, and Michael Dahlin. Minimal byzantine storage. In Dahlia Malkhi, editor, *DISC*, volume 2508 of *Lecture Notes in Computer Science*, pages 311–325. Springer, 2002.

20 Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

21 Alexander Spiegelman and Idit Keidar. On liveness of dynamic storage. In *Structural Information and Communication Complexity - 24th International Colloquium, SIROCCO 2017, Porquerolles, France, June 19-22, 2017, Revised Selected Papers*, pages 356–376, 2017.
22 Alexander Spiegelman, Idit Keidar, and Dahlia Malkhi. Dynamic reconfiguration: Abstraction and optimal asynchronous solution. In *DISC*, pages 40:1–40:15, 2017.