BGWS: A Dependable Decentralized Ledger for a Blockchained Global Wallet Service

Diogo Cebola, 52718 and Gonçalo Areia, 52714

Abstract—

Index Terms—Dependable Distributed Systems, Blockchain, Byzantine Fault Tolerance

1 Introduction

This work was proposed by professor Henrique Domingos as the course project of "Dependable Distributed Systems" 2020-21.

From cloud and world-wide e-commerce platforms to banking replacements, distributed systems are widely used nowadays for various tasks. Such tasks can range from simple consultations (e.g. querying products available in an online store) to critical operations (e.g. money transfers). Frequently these operations need to be ordered and the processes involved are required to reach an agreement. To solve this problem distributed systems often rely on state machine replication and a consensus algorithm.

With the increasing number of distributed systems being used for critical tasks, it is imperative that they are dependable and ensure high degree of reliability, availability, safety, confidentiality, fault tolerance, integrity and maintainability. When designing dependable systems it is necessary to consider the execution environment and the correct adversary model.

Inspired by the rise of blockchain technologies and the way such technologies deal with consensus, in the realm of financial operations, we will present the implementation of a dependable decentralized ledger for a blockchained global wallet service. From this point onwards we will refer to the system using BGWS. BGWS will operate in an asynchronous environment, with no timing assumptions, over authenticated channels and consider byzantine faults.

We will study how the number of replicas impact the overall performance of the system, how the size of blocks affect the latency of successful mining operations and how the system tolerates faults.

The remaining of the document is organized as follows. In Section 2, we introduce important concepts related to our work, more specifically, State Machine Replication, Consensus under the byzantine fault model and Blockchain technologies. In Section 3, we present BGWS system model and architecture. Internal mechanisms, design assumptions and the service planes are covered in section 4. In Section 5, we explain the technology stack used and development issues. Section 6 will present the experimental setting and discussion of the obtained results. In Section 7 we evaluate the current state of the BGWS system, its correctness, limitations and trade-offs. Finally, in Section 7 we conclude the document and mention future work.

2 Background

2.1 State Machine Replication

We can define a state machine as a set of states, a set of transitions, and a current state. When a user issues an operation to the machine, that operation triggers a transition from its current state to a new one, producing an output.

In State Machine Replication, multiple replicas of a system are created, each one being a deterministic state machine. If they are given an input of the same operations in the same order, all the replicas will transition to the same state and produce the same output. The ordering of the operations will have to be decided through an agreement protocol.

2.2 Consensus

Consensus algorithms are used by processes to reach agreement. There are various forms of consensus and algorithms that solve them [1] [2]. From binary consensus, multi-valued consensus, to probabilistic consensus that relies on randomization algorithms [3] [4]. Let us consider the following specification of consensus:

> **C1 (Termination)**: Every correct process eventually decides a value.
> **C2 (Validity)**: If a process decides v, then v was proposed by some process.
> **C3 (Integrity)**: No process decides twice.
> **C4 (Agreement)**: No two correct processes decide differently.

In an environment with byzantine such specification is not enough. Byzantine processes can have an unpredictable and arbitrary behavior, so we need to restrict all properties to correct processes and the validity property must require that every value decided by a correct process must have been proposed by some process. We also need to adopt the notion of *weak* and *strong* validity, respectively, for execution scenarios where all processes are correct and execution scenarios where there are byzantine processes.

In byzantine consensus values can not be invented arbitrarily and must have been proposed by a correct process. In *strong* validity, if there is no agreement in the decided value, an arbitrary $\perp$ is decided, that symbolizes "not decided".

To implement byzantine consensus in practice, the PBFT [5] algorithm uses public-key cryptography, and digital signatures. The system needs quorums (majorities) of $3n + 1$ nodes to tolerate n faults.

2.3 Blockchain Technology

First we will define the concepts of *transaction*, *smart-contract* and *block*.

A transaction represents a contract between peers, such contract can be as simple as a transfer of money from peer A to peer B or can be as complex as a programmable "smart-contract" with a custom set of rules, inputs and outputs.

A block is a record that contains n transactions, transactions within the block must be exclusive to the block.

A blockchain can be defined as an expandable list of blocks that are logically ordered with the use of hash functions. Each block contains the hash of the previous block in the list, forming a logical chain.

Consensus in blockchains is solved with a *proof of finality*. Each peer must propose a block with this proof. A decision protocol is applied over all proposed blocks, and the block with the best proof is added to the chain. Normally, a block can only be considered definitive, in the chain, after a certain amount of new blocks have been added. This is due to the irreversibility of the chain being derived from the cost of reverting all the previous blocks. The cost is directly associated with the type of proof used.

Bitcoin [6] uses a *proof of work*, where the cost of the proof is the computational difficulty to complete a cryptographic puzzle that consists in generating a block hash with a target number of leading 0s. To execute a *Sybil attack* on the such network, not only the attacker would have to possess the computational power superior to 51% of the peers but, to revert the order of the chain, it would have to re-compute all previous *proofs of work*. In practice, this is almost impossible to execute, and it results in being more profitable to just contribute as a well-behaved peer.

Bitcoin, with this technique to solve consensus, was the first decentralized virtual currency to solve the *double spending problem*. Since its appearance new blockchains have been proposed, some with a different modular and permissioned architecture, like the Hyperledger Fabric [7] [8], others with more scalable and efficient proofs of finality [9], like Ethereum with its Casper protocol [10] [11] and a *proof of stake*.

3 System Model and Architecture

4 Mechanisms and Service Planes

5 Implementation

5.1 Technology Stack

5.2 Issues

6 Experimental Evaluation

6.1 Experimental Setup & Methodology

6.1.1 Validation

6.1.2 Evaluation

6.2 Experimental Results

In this section we present the results as an average of the three runs executed, and describe the patterns observed.

6.3 Discussion

In this section we will discuss the variation in the results obtained for the different experiments.

7 Coverage of Requirements

8 Conclusion & Future Work

References

[1] L. Lamport *et al.*, "Paxos made simple," *ACM Sigact News*, vol. 32, no. 4, pp. 18–25, 2001.

[2] 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, p. 305–320.

[3] H. Moniz, N. F. Neves, M. Correia, and P. Verissimo, "Ritas: Services for randomized intrusion tolerance," *IEEE transactions on dependable and secure computing*, vol. 8, no. 1, pp. 122–136, 2008.

[4] H. Moniz, N. F. Neves, M. Correia, and P. Veríssimo, "Experimental comparison of local and shared coin randomized consensus protocols," in *2006 25th IEEE Symposium on Reliable Distributed Systems (SRDS'06)*. IEEE, 2006, pp. 235–244.

[5] M. Castro, B. Liskov *et al.*, "Practical byzantine fault tolerance," in *OSDI*, vol. 99, no. 1999, 1999, pp. 173–186.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2019.

[7] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the thirteenth EuroSys conference*, 2018, pp. 1–15.

[8] J. Sousa, A. Bessani, and M. Vukolic, "A byzantine fault-tolerant ordering service for the hyperledger fabric blockchain platform," in *2018 48th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2018, pp. 51–58.

[9] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer *et al.*, "On scaling decentralized blockchains," in *International conference on financial cryptography and data security*. Springer, 2016, pp. 106–125.

[10] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.

[11] O. Moindrot and C. Bournhonesque, "Proof of stake made simple with casper," *ICME, Stanford University*, 2017.

[12] A. Bessani, J. Sousa, and E. E. Alchieri, "State machine replication for the masses with bft-smart," in *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2014, pp. 355–362. [Online]. Available: https://github.com/bft-smart/library