HDS Serenity Ledger

- Report -

Contributors: Paulo Bolinhas - ist1110976 | Rui Martins - ist110890 | Nuno Palma - ist86903

1. Introduction

Serenity Ledger is a simplified blockchain system with high dependability guarantees, that uses the Istanbul BFT Consensus Algorithm protocol as its core. In this report, we will present our initial implementation of this application, a codebase that handles several threats, and arbitrary behaviour and guarantees to the client

2. Design

2.1. Main design

Entities: As an assumption, every public key is pre-distributed (published) to every Client and Node using config files that are globally known. Client and node interactions in the blockchain are defined in a *ClientLibrary* class and a *NodeService* class, respectively.

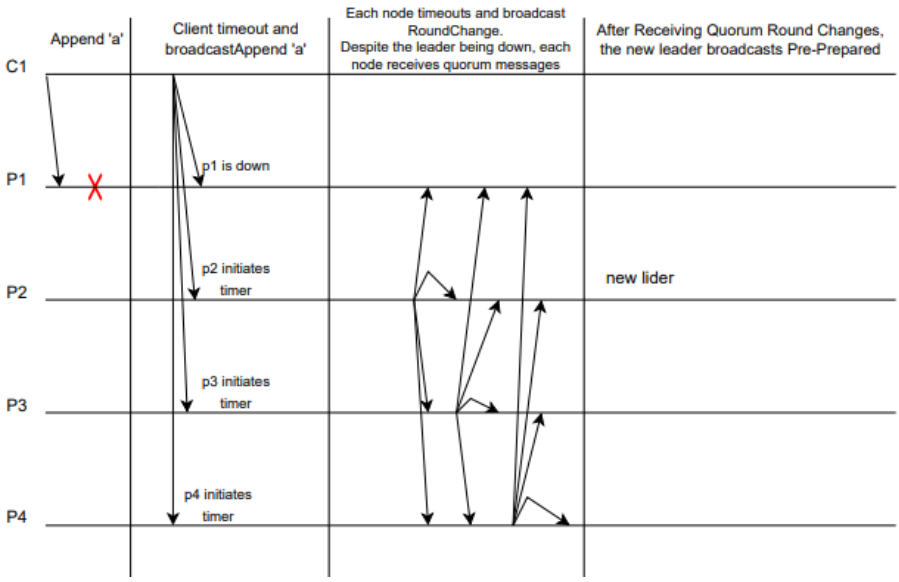
Links, Signatures and Keys: The connection, previously implemented using perfect links, was now over-secured with authenticated perfect links. To achieve that, we use digital signatures. To achieve that, each entity signs on the *sending* and the receiver checks it. The signatures are created with keys generated with OpenSSL, and after loaded on Java. We did not implement confidentiality since was out of our scope.

Algorithm implementation: The algorithm that we implemented has the IBFT as its core but with some differences. To begin with, the similar parts relate to the node's timeouts. Each node starts/resets its timer on the functions *startConsensus* and *uponPrePrepare* and stops it on *uponCommit* after receiving *quorum commit* messages. About the differences, these timers are extremely important to trigger *Round-Change* and, despite that IBFT implements the upon rule at line 5 of algorithm 3 and resets the timer on *uponRoundChange*, we did not implement them to promote simplicity. This fact does not compromise our implementation since their main purpose is to accelerate the algorithm. Lastly, our timer does not increment exponentially because our teacher said that it was not strictly necessary.

2.2. Possible Threats and corresponding Protection Mechanisms

Malicious client: A cliente that does not have its public key published is rejected by the network. Also, if the malicious cliente tries to replay a message of a legitment node, he won't be able.

Leader down: The leader crashes after receiving the first *append* message from the client. Because of that, the client will timeout and broadcast the same message, triggering the timer on each node that will start a *Round-Change* after it expires. After that, the blockchain *Decides* as usual.



Malicious leader: If the leader receives *Append* ***'a'*** he will not be able to either broadcast *Pre-prepare* **'b'** or use the message to create a replay attack. This is because the client signs the message and each message has a timestamp that is checked too.

Network Delay or Packet drop after Prepare: If the consensus instance already prepared a value and a round but the network delays or drops, the timer of each node will trigger and start a round change. The prepared value and round are passed to the next round. As long as the network problems stop, the consensus will finish as usual.

3. Dependability Guarantees

Liveness: The round changes and timers are pivotal for ensuring liveness as they facilitate the rotation of leader roles among different processes. Always eventually a new leader is defined which will eventually *Decide* the consensus (something good happens).

Safety: The use of message justification, with the arrival of a quorum of *Round-Change* messages validations, always guarantees that the proposal value chosen by the leader of a new round is safe. Quorum validations and the carrying of the prepared values through rounds ensure safety across rounds (nothing bad should happen).

Agreement: We achieve this by ensuring message integrity. If a node sends a message with value v, no other node will process messages that have been altered to v' it would fail the integrity check of the signature.

Validity: A correct process only accepts a message if it deems it valid, requiring evidence of integrity and sender authentication, such as a digital signature, for validation, as we use when we share messages.

Termination: Every correct process eventually decides since it’s a correct process, that never goes down.

4. Conclusion

In this report, we introduced our initial implementation of a simplified blockchain system with high dependability guarantees, emphasizing secure interactions and adherence to IBFT algorithms. The consistency of our implementation allows us to have a solid foundation to keep enhancing and growing the network, every dependability guarantee and to start the next phase related with the actual blockchain.