HDS Ledger
Stage 1

Highly Dependable Systems

Alameda

Group 27

David Cruz 89377

Catarina Beirolas 93034

João Luís 95607

17th March 2023

1 Design

The system membership is static for the entire system lifetime, including a predefined leader process and the membership information is known by all participating processes before the start of the system. Because in this delivery we only cover Algorithms 1 and 2 from the IBFT [1], we don't handle leader changes.

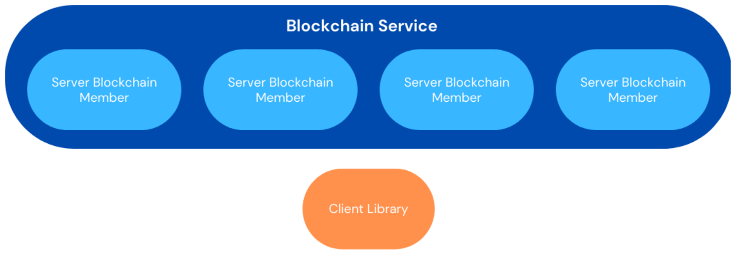

Figure 1: Design Overview

1.1 Client

The Client has a CLI through which it can submit a request (append a string) to the blockchain and receive a reply confirming if and when the request was executed.

1.2 Server

1.2.1 Channels

As required the basic communication is done using **UDP** which allows the network to approximate the behavior of Fair Loss Links. To simulate Perfect Channels, we implement the following:

- To avoid message loss, we set a 5 second timeout and re-send the message until a reply is received;
- To avoid message duplication, we added an ID to every message;
- To guarantee Integrity and Authentication, our messages are signed with the sender's private key and the receiver verifies it with the sender's public key;
- To guarantee freshness in the communication, the message's sender adds a nounce and then verifies if the nounce in the reply is the expected one.

1.2.2 Broadcast

We chose to use Best Effort Broadcast in our project. To simulate Perfect Channels and guarantees such as freshness, integrity, non-loss and non-duplication we implemented mechanisms which we explain later in the report.

1.2.3 Command Processing

We use threads to allow the Servers to still receive requests from the Clients while a Consensus instance is happening. When the consensus protocol has started, if a Server receives a message from the Client, it is added to a queue. When Servers are deciding, they remove the first string in the queue, so that all requests from the Client are only processed once and in order.

1.2.4 Command Translation

The Servers translate Client requests to invocations of the Istanbul BFT protocol [1], and also for the respective response to the Client. When a Client makes a request to the service, the leader server begins the Consensus Protocol with the START procedure of the Algorithm 1. At the end of the Consensus, after the DECIDE step of the Algorithm 2, the string sent by the Client is added to the blockchain and the Servers answer the Client with an "ACK" message. When the Client has two (f+1) "ACK" messages, it knows the command has been applied.

1.2.5 Leader

The leader doesn't change during the system lifetime and is determined by the Server with the lowest port.

1.3 Keys

We assume there is a Public Key Infrastructure in place. Blockchain members and blockchain clients use public/private keys that we previously generated, which are pre-distributed before the start of the system, and this information is also included in the static system membership.

1.4 Threats and Protection Mechanisms

In order to guarantee the required security and design specifications, we designed a set of demo tests (puppet masters) that create and run 4 servers and 1 client, they are described more in depth in our Readme:

- Correct Functionality
- Byzantine Server Attack
- Duplicate Message Attack
- Authentication Attack
- Correct Order Message Processing Demo
- Message Waiting in Queue Demo

2 Dependability Guarantees and Security Attributes

Confidentiality: based on the project requirements for this stage, it was decided that confidentiality wouldn't be necessary for this implementation.

Integrity: since we are using UDP, the network is unreliable and communication channels are not secured, every time any data is transferred from Client to Server and between Servers, we perform integrity checks through the usage of message signatures. This means that any time we send a message, we hash it (using SHA1) and encrypt it with the sender's private key. Then the receiver must decrypt the message with the sender's public key and verify if the sender is correct.

Availability: a subset of the blockchain members may be malicious and behave in an arbitrary manner. However, as long as the Client receives f+1 "ACK" messages, the service is available.

Reliability: this implementation has Servers with pre-determined functions and the algorithms that allow for a new leader election aren't yet implemented, so we can provide reliability over the period of time that the servers are active and correct, because we still need to reach a quorum.

Freshness: in the communication between the Client and the Servers and between Servers, the messages are sent with a message nounce (which is also signed with the sender's private key) and when a reply is received, the sender checks if the message nounce received corresponds to the expected value, guaranteeing it is not an old message being replayed.

Authentication: our project provides integrity and freshness, so it also guarantees authentication.

2.1 References

[1] Henrique Moniz. The Istanbul BFT Consensus Algorithm.

https://arxiv.org/pdf/2002.03613.pdf