HDS Ledger

Highly Dependable Systems

Alameda

Group 27

David Cruz	Catarina Beirolas	João Luís
89377	93034	95607

17th March 2023

1 Design

In our project, the system membership is static for the entire system lifetime, including a predefined leader process and the membership information is well-known to 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.

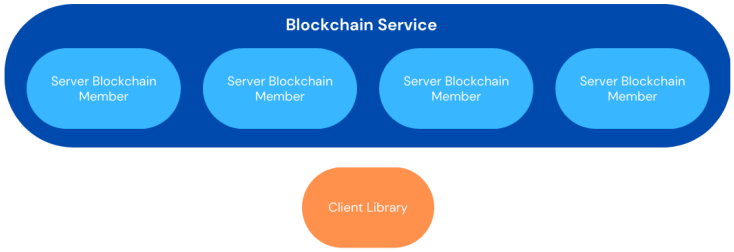

Figure 1: Design Overview

1.1 Puppet Master

A Puppet Master is used in our project when we want to manually instantiate both Server and Client processes (instead of the automated build and testing process using JUnit), mostly during the earlier stages of development.

1.2 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.

1.3 Server

1.3.1 Channels

As required in this project statement, the basic communication is done using **UDP** which allows the network to approximate the behavior of Fair Loss Links. TODO (isto faz sentido???) To simulate Perfect Channels, we implement the following:

- To avoid message loss, we use 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.

-¿ evitar duplicação de mensagens (envia-se na mensagem um id e quem recebe ve se não recebeu ja esse id) // FEITO -¿ protocolos de segurança: —¿integrity/authentication (assinar a mensagem com a chave privada e quem recebe verificar com a publica) // ta feito —¿ freshness (adicionar um nounce à mensagem e receber a mensagem com esse nounce) // ta feito

1.3.2 Broadcast

We chose to use Best Effort Broadcast in our project, as we didn't need stronger guarantees at this stage.

1.3.3 Command Processing

1.3.4 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.

1.3.5 IBFT

-

After the leader broadcasts a "PRE-PREPARE" command to all Servers, the Algorithm 1 from IBFT is complete and Algorithm 2 begins:

- Upon receiving a "PRE-PREPARE" command, Servers broadcast a "PREPARE" command;
- When a quorum (3, in our case) of "PREPARES" is received by the leader, it broadcasts a "COMMIT" command and when each Server receives this command, they update the HashMap which stores the number of Servers who accepted the "COMMIT" command;
- The moment the service reaches 3 Servers ready to "COMMIT", the "DECIDE" process takes place and the consensus is reached.
- TODO: explicar como a confirmação chega até ao cliente?

1.3.6 Leader

As already mentioned, the leader doesn't change during the system lifetime and is determined by the Server with the lowest port.

1.4 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.

1.5 Communication Technology

As required in this project statement, the basic communication is done using **UDP** which allows the network to approximate the behavior of Fair Loss Links.

1.6 Threats and Protection Mechanisms

In order to analyze our project and guarantee that it does provides the required security and design specifications, we designed a set of tests:

1.6.1 Functionality Test

Wrong Input

1.6.2 Replay Attack

1.6.3 Tamper Attack

1.6.4 Authentication Attack

1.6.5 Drop Attack

2 Dependability Guarantees and Security Attributes

In this section, we will discuss some of the essential characteristics related to security and dependability. We will explain the reasons why we included or excluded them in our project, and if they were included, we will describe how we achieved them.

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: as mentioned before, our program 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