TÉCNICO LISBOA

Highly Dependable Systems

2018-2019

HDS Notary – Stage 2

Group 21

Diogo Fernandes	76534
Jose Canana	82039
Diogo Cardoso	94024

May, 2019

1. System Architecture and Functionality

In this Stage we remain with the previous requisites and main functionality structure and system architecture but implemented a replicated system for the notary servers with measures in place to tolerate byzantine faults. Instead of only one Notary server we now start several servers that are known to the clients at the moment of their initialization. Due to this alteration, the communication protocol is now different from the previous stage. The clients now send the request to all known servers and receive an agreed response in order to proceed correctly (the used communication protocol and consensus algorithm will be explained in detail in the following sections). We have also updated the previous server functionality where now one entry per transaction is created and stored in a server file for future proofs of purchases. This new feature enforces the previously implemented transaction Integrity requisite preventing non-repudiation of messages / transactions.

A new security requirement was introduced in this stage in order to prevent DOS attacks and it’s implementation will also be explained in the next sections.

1.1. Notary Server

In order to simplify the test cases it is possible to start the server with or without the previous Citizen Card implementation (a user prompt appears at the beginning of the server run in order to set this condition).

It is imperative that all running servers have the same initial condition, this is, all servers need to use a citizen card or none can in order for the application to run correctly.

1.2. Notary Client

The client also starts with a new resource file were the names and ports of the Notary servers are obtainable and a new module (called Round-Robin) used to send and analyze the received responses from the requests made. In order to prevent malicious clients from attacking the servers in a denial of service fashion, an anti spam mechanism was implemented and activates before any request.

2. Security Measures

2.1. Replicated System

We have implemented a (1,N) Byzantine Regular Register that tolerates up to f failures. The f value is known to the clients for them to correctly judge the majority of the received responses.

When a client calls a remote method the request is sent to all active servers and the client waits for at least N-f acks from the servers (being N the number of active servers), the ack works as a promise from the server that it can and will perform the task if asked again. After the number of acks is acceptable the client calls the method again, this time for real, and waits again for a byzantine majority of results which are then analyzed in order to end with a consensus from all the received responses and present the client with the correct one.

For the Write time stamps (wts) and the Read ids (rid) we have used the previously implemented message-ids due to the fact that they are sequential and were already in place in the request and response SOAP messages.

2.2. Anti Spam Implementation

In order to prevent DOS attacks from malicious clients, before a remote method call, a new anti spam mechanism is activated. This simple function asks the user for an addiction of two random numbers and takes about 10 seconds to verify the user input. In case of success the operation proceeds as normal and in case of wrong response from the user the function runs again with a higher difficulty.

3. Final Notes

Due to lack of time we were unable to implement a communication protocol between the Notary servers and as a result the Atomic register was not implemented.

A solution for byzantine clients was not needed due to the fact that there is no information ever stored on the clients and therefore nothing for them to wrongly evaluate or send to the server. Also, all possible attacks from malicious clients were already prevented on the previous Stage.

In the .zip file there is also a Readme with a set of tests that demonstrate all the dependability and security features of the system.