Confiabilidade de Sistemas Distribuídos – 2018/19

Project 1

Byzantine fault-tolerant system for storing wallets

Introduction

The goal of this project is to create a system that maintains information about wallets. This system could be used, for example, to maintain the information about some virtual currency or to maintain the information about wallets in some application (e.g. game).

Architecture and specification

The system exports two operations (the parameters in the operations are indicative – you may add additional parameters if necessary):

- createMoney(id, amount)
    - Add *amount* to the wallet identified by *id*.
      You should define under which conditions this operation succeeds or fails.

- transfer(id_from, id_to, amount)
    - Transfer *amount* from the wallet identified by *from_id* to wallet identified by *to_id*.
    - This operation should fail if balance of the *from_id* wallet is smaller than *amount*.

- get(id)
    - Return the amount of money associated to the wallet identified by *id* or 0 if that wallet does not exist.

The system should be implemented using a client/replicated server architecture and expose a REST API (or a SOAP API).

The system should be able to tolerate Byzantine faults in the servers, using the traditional assumption that for tolerating `f` faults, the number of replicas should be `3f+1`. For tolerating Byzantine faults, the server should implement some form of Byzantine Fault-tolerant replication.

Suggestion: use library BFT-smart to achieve this goal.

Your solution should guarantee that:

- The client is connecting to a server of the systems and avoid replaying.
    - Suggestion: use TLS connections between the client and the server.
- The client should verify that the replies from the server are correct.
    - Suggestion: the server must send to the client the replies received by the BFT-smart client, so that the client of your system can verify that the result was returned by f+1 replicas.
- The systems must guarantee that the client has permissions to execute the transfer

operations.
    - Suggestion: use Bitcoin approach to avoid having to implement authentication mechanisms in the servers.

For the client of your system, create a class that encapsulates the interaction with the servers, and maintains the client private information – e.g. if using Bitcoin approach for identities, the private keys should be kept in the state of this class.

Evaluation

The system must be evaluated experimentally using, preferably, a distributed setting with 3 machines connected to a local network, running the following software:

- 2 of the machines should run two replicas of the server each;
- The third machine should run a simple benchmark, with multiple threads. The benchmark should start by creating a set of wallet with some initial money, followed by, in different threads, a sequence of transfer operations. Suggestion: create 100 * num_threads wallets; run threads by at least 3 minutes.

Compare a configuration of the system with BFT-smart configure to tolerate 0 byzantine faults and to tolerate 1 byzantine faults.
The report should present the throughput of the system and latency of the operations.

Optional: compare with an additional configuration where there is a byzantine replica. Suggestion: this can be implemented by having one replica that returns incorrect values randomly.

Report

The report should present your work and have the following structure:
Introduction : presents the objectives of the project;
System design : presents the architecture of the systems and the algorithms used;
Evaluation : presents the evaluation (note: do not forget to present the experimental setup);
Conclusions.

Dates

8/April – delivery of the code;
TBD – delivery of the report.

Sobre a biblioteca BFT-Smart:

BFT-SMaRt is a replication library written in Java. It implements state machine replication. It is designed to tolerate Byzantine faults, while still being highly efficient -

even if some replicas are faulty. In this page we describe how this library works, in a high level of abstraction. More details can be found in the technical report describing the system.

- **https://github.com/bft-smart/library**
- https://github.com/bft-smart/library/wiki/How-BFT-SMaRt-works
- http://bft-smart.github.io/library/
- State Machine Replication for the Masses with BFT-SMART, DSN 2014