Instituto Superior Técnico

Highly Dependable Systems

Group 13 - Taguspark

Project - Stage 2
Highly Dependable Location Tracker

Sara Machado
86923

Rafael Figueiredo
90770

Ricardo Grade
90774

1. Design

Our system is implemented in Java, the communication channels are in gRPC, and data persistence and corruption prevention is guaranteed in Servers by using PostgreSQL databases.

Our system is divided into 5 modules:

- A Server which is responsible for answering the clients.
- A User and a HA which are the clients of the system.
- A Library that contains cryptographic functions used by all modules.
- A Byzantine which contains misbehaviours of the clients and Servers.

1.1. Abstraction Layers

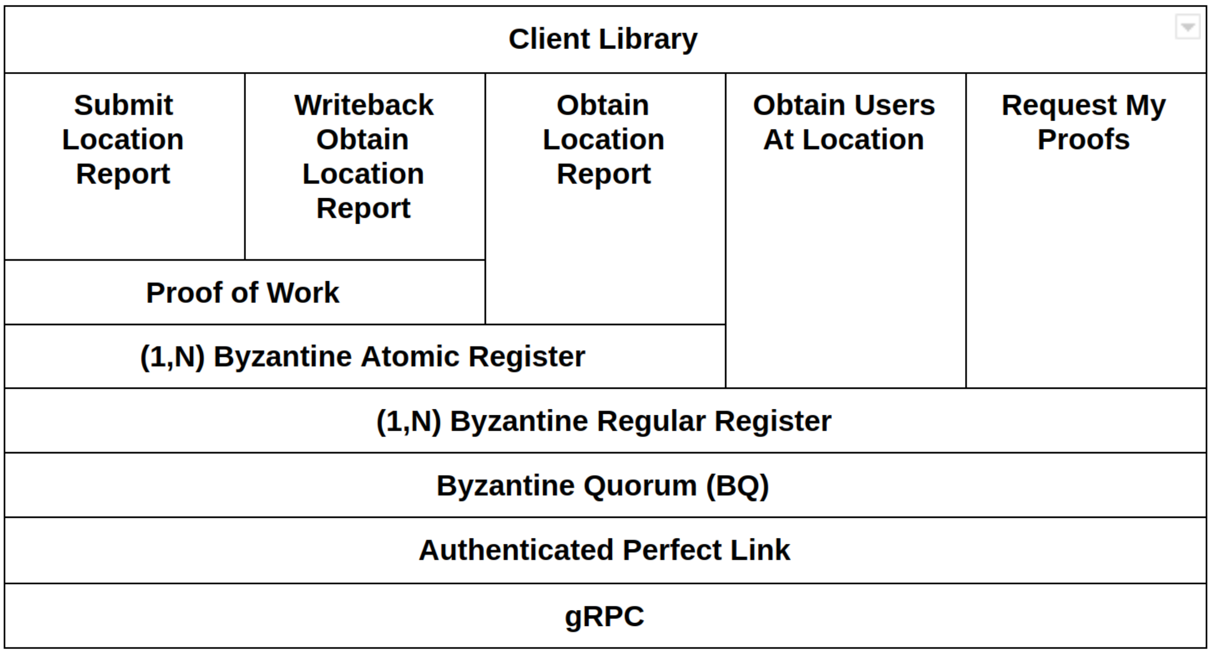

Figure 1: Client (User and HA)

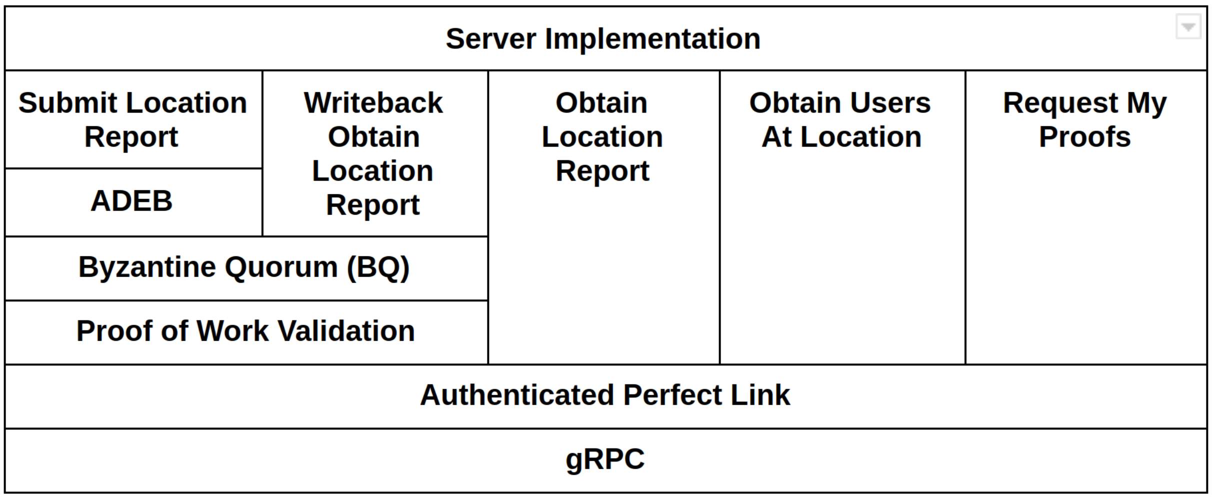

Figure 2: Server

2. System Processes

2.1. Base Considerations

- In a scenario where there are fs byzantine Servers, the number of Servers that are created is $N = 3fs + 1$.
- Our system is asynchronous. Therefore, a User can request its location at any epoch.
- Communication with a Server uses HMAC for authenticity and AES-CBC for confidentiality after a session is established. Communication between Users uses signatures.
- All exceptions thrown across modules are signed along with the received nonce, with the emitters Private Key, to prevent spoofed message rejections.
- All Private Keys are kept confidential, being stored in the Java Key Store.

2.2. Diffie-Hellman

Our system ensures Perfect Forward Secrecy. Therefore, whenever a system entity wants to request a Server, they establish a Session Key, to do so they execute the DH. This key has a short-term period validity. In the session establishment the Server keeps a nonce sent by the User in order to ensure its freshness. Furthermore, to ensure freshness in the subsequent messages the Server sends a nonce to the system entity, which should be incremented and sent back in every message within the given session.

2.3. User Location Proofs Gathering

When a User needs to submit its *LocationReport* at a given epoch to the Servers it requests *UserProofs* to the nearby Users. The respondents will assert the closeness of the User and respond accordingly.

When the requester is indeed nearby, the respondent will generate a *UserProof* and return it to the requester. A *UserProof* is a signature of the requester username, location, and epoch.

To assure the Servers can verify the correctness of its *LocationReport* in a scenario where it can be tolerated $fu' < fu$ byzantine Users in the proximity of each User, the requester gathers fu valid *UserProofs* to insert in the *LocationReport*.

2.4. (1,N) Byzantine Regular Register [1]

A correct client makes a request to all Servers and waits for $2fs + 1$(BQ) number of replies. Guaranteeing that if a submission of a *LocationReport* already finished, that result will always be returned by at least one Server to the client.

2.5. (1,N) Byzantine Atomic Register [2]

When a client obtains a *LocationReport*, which is not present in a BQ of Servers, then a writeback is made in order to ensure that, after it returns, itself, and every other client will obtain this *LocationReport*.

2.6. ADEB [3]

In order to prevent the correct Servers from becoming inconsistent between each other, prior to a *LocationReport* submission finishing, the Servers execute the ADEB, in order for all the Servers to write the same *LocationReport* in its database. This guarantees Servers consistency even in a scenario when a byzantine User submits different *LocationReports* to different Servers.

Moreover, a modification to the *ReadyMessage* of the ADEB was made, all of them carry the emmitter's *ServerProof* of each *UserProof* present in the *LocationReport*. Each *ServerProof* consists of a signature of a *UserProof* made by a Server. Doing so, when the ADEB finishes all correct Servers possess a BQ of *ServerProofs* for each *UserProof* which are required for the clients to validate its correctness.

2.7. Spam Combat Mechanism

In order to mitigate an attack where a client could flood the system with computational expensive requests, a Proof of Work mechanism has been implemented. The protected requests are the following:

- *LocationReport* submission, where all Servers must execute ADEB, exchanging a quadratic number of messages among themselves, verifying and signing each *UserProof*, and verifying the *ServerProofs* received in the *ReadyMessages*.
- *LocationReport* writeback, which although less heavy than its submission, each Server must verify both *UserProofs* and *ServerProofs* contained in the *LocationReport*.

Each client must calculate an integer such as when appended to the message, its combined hash has an established amount of leading zeros. Due to the lifetime of the session and the generation of its nonce being made by a Server it is impossible for a client to calculate this integer prior to its establishment.

2.8. Queries

- *ObtainLocationReport*: Returns a valid *LocationReport* if exists.
- *RequestMyProofs*: Returns the valid response with the largest amount of *UserProofs*.
- *ObtainUsersAtLocation*: Returns the valid response with the largest number of Users.

In order to verify the validity of a *LocationReport*, it must contain fu number of valid *UserProofs*.
To verify the validity of a *UserProof* it must contain a BQ number of valid *ServerProofs*.

3. Guarantees

- **Dropping Messages Prevention**:
  - o If the request times out, the requester retries it.
- **Rejecting Messages Prevention**:
  - o If the exception is not correctly signed by its emitter, the requester retries the request.
- **Replay Attacks Resistant**:
  - o A nonce is always appended to every request. The signature or HMAC of the reply must contain this nonce.
- **Authenticity**:
  - o Signatures (RSA-4096) or HMACs (SHA-512) are generated and appended in every message over its payload and nonce.
- **Perfect Forward Secrecy**:
  - o The Diffie-Hellman (DH-3072) is used to establish Session Keys between the Server and its clients with a short-term period validity.
- **Confidentiality**:
  - o Messages are encrypted by Session Keys (AES-128) using the CBC Mode between the Server and its Clients.
- **Byzantine Users Safeness**:
  - o The Server only accepts a *LocationReport* submission when N *UserProofs* issued by different Users are deemed valid, being $N \geq fu$.
- **Byzantine Server Safeness**:
  - o A *LocationReport* submission is only written after executing the ADEB.
- **Byzantine Server-Client Safeness**:
  - o All *UserProofs* are signed by a BQ of Servers, preventing that a byzantine Server can inject by itself additional valid *UserProofs*.
- **Data Persistency and Non-Corruption**:
  - o The Server keeps its data in a database, which survives to system crashes, keeping it consistent even if it happens in the middle of an operation.
  - o Moreover, transactions are used when an operation requires modifying multiple tables.

4. References

- [1]: Section 4.7 in Course Book
- [2]: Section 4.8 in Course Book
- [3]: Section 3.11 in Course Book