Highly Dependable Systems

Sistemas de Elevada Confiabilidade
2019-2020

Project – Stage 2
Dependable Public Announcement Service

The goal of the second stage of the project is to extend the implementation of the first stage to tolerate the possibility that the system may be subject to Byzantine faults, affecting both the client and server processes.
More precisely, we will keep the same specification for the Dependable Public Announcement Server from the previous stage, but enhance the client and server implementations to make them resilient to Byzantine faults.

On the server side, the single server from the first stage must be replaced with a set of N replicas which collectively implement a Byzantine Fault Tolerant service. The size N of this set must be chosen in such a way that depends on a value f, which is a system parameter representing the maximum number of Byzantine faults, i.e., the maximum number of faults that may occur while preserving the correctness of the DPAS.

On the client side, we assume that a malicious user may arbitrarily alter the client library to attack the system, for instance, submit invalid announcements or submit on behalf of other clients, among others.

The design and implementation of the second stage should retain dependability guarantees despite such malicious clients. It is however outside of the scope of the project to integrate techniques that aim at protecting a legitimate user from a malicious client library (that may, e.g., leak the user's private key or produce fake replies for the user without even contacting the servers).

Design requirements
The basic change from the previous design is to replace the client-server communication between the library and the server with a replication protocol between the library and a set of server replicas. The Announcement Board of each user should behave as a (1,N) Byzantine Atomic register and the General Announcement Board should behave as a (N,N) Byzantine Regular register.

The `post` and `postGeneral` operations should be treated as write operations and the `read` and `readGeneral` operations should be treated as read operations.

Implementation Steps

To facilitate the design and implementation of the HDS DPAS, students are encouraged to break up the project into a series of steps, and thoroughly test each step before moving to the next one. Having an automated build and testing process (e.g.: JUnit) will help students progress faster. Here is a suggested sequence of steps:

- Step 1: Replicate the server, without implementing any scheme to synchronize the state of the various replicas and assuming non-byzantine clients. This will involve starting N server replicas instead of one, and replacing the client to server communication with a loop that repeats each communication step N times.
- Step 2: As for the next step, students are advised to first implement the (1,N) Byzantine Regular in Section 4.7 of the course book (Introduction to Reliable and Secure Distributed Programming, 2nd Edition)
- Step 3: Next, consider a transformation from (1,N) Byzantine Regular register to a (1,N) Byzantine Atomic register for each user's Announcement Board
- Step 4: Next, consider a transformation from (1,N) Byzantine Regular register to a (N,N) Byzantine Regular register for the General Board.
- Step 5: The Byzantine registers assume that only the server can be Byzantine and not the clients. Reason on what is the impact of having Byzantine clients and propose, if needed, solutions to cope with such an issue.
- Step 6: Implement a set of tests that demonstrates the various dependability features integrated in your system.

Submission

Submission will be done through Fénix. The submission shall include:

- a self-contained zip archive containing the source code of the project and any additional libraries required for its compilation and execution. The archive shall also include a set of demo applications/tests that demonstrate the mechanisms integrated in the project to tackle security and dependability threats (e.g., detection of attempts to tamper with the

data). A README file explaining how to run the demos/tests is mandatory.

- a concise report of up to 6,000 characters addressing:
    - explanation and justification of the design, including an explicit analysis of the possible threats and corresponding protection mechanisms.
    - explanation of the integrity guarantees provided by the system
    - explanation of other types of dependability guarantees provided

The deadline is **May 13 2020 at 17:00**. More instructions on the submission will be posted in the course page.