Viewstamped Replication Revisited Implementation

Gleb Leonov
Stanford CS244B
2017-2018 Autumn

Project

The goal of this project is to build a clean, simple implementation of viewstamped replication based on the updated Liskov paper
Source code is located on Github
The project is implemented in Kotlin, modern and rapidly evolving programming language. Recently it became an official language for Android development.
The project uses Maven build system
Docker and docker-compose were used to create executable demo. To run it, one need to have Java, Maven and Docker installed. Do the following:
* run build.sh from project root folder to build sources and Docker containers
* run "docker-compose up" in vr-sample/docker-compose folder to start demo
To see how views are changing, one can stop and restart containers via docker-compose commands

Implementation

The project is divided into two parts - "vr-core" and "vr-sample" modules.
"Vr-core" directly contains implementation of VR protocol, "vr-sample" - simple demo application build upon "vr-core".

Classes in "vr-core" use scpd.vr.network.Channel abstraction for message transport and scpd.vr.scheduling.Scheduler abstraction for timed events such as messages resending. Such approach allows to write extensive single-threaded tests with full control of different events order.
Tests for "vr-core" contain scpd.vr.replica.TestNetwork which mocks network with a set of buffers and provide a way to manage them.

Concurrency

Concurrency isn't covered in the paper. From my point of view, VR protocol can hardly be parallelized since it's goal is to build full log of client requests. Also, executing requests can't be done asynchronously with internal VR protocol work since we need to execute requests sequentially.
On the other side, client doesn't need to be blocked when some request is in progress. To support it, scpd.vr.replica.Client request() method accepts a callback as parameter instead of returning result.

Issues and possible future work

The main issue in my project is that I didn’t have time to implement any optimization located in “Pragmatics” chapter of the paper. So, current implementation isn’t ready for real usages. Possible future work can be quite straightforward - implement log truncation, fast reads and reconfiguration.