

Algoritmos e Sistemas Distribuídos- Project Phase 2

João Carlos Antunes Leitão

NOVA Laboratory for Computer Science and Informatics (NOVA LINCS)

and

Departamento de Informática

Faculdade de Ciências e Tecnologia

Universidade NOVA de Lisboa

Preliminary (incomplete) first version

20th November 2020

1 Overview

This document discusses the second phase of the ASD project for 2020/21. The project has two phases which are fully independent between them. The primary goal of the project is to build and compare two state machine replication [6] solutions (that will be applied to replicate a simple application whose state is essentially a hash map) that differ on the agreement protocol being employed. One of them will use Paxos [3, 4, 5, 1] (but since multiple Paxos instances will be required, you will have to rely on some simple meta-protocol that wraps the different Paxos protocol) and Multi-Paxos. You are going to compare these two different design alternatives by conducting an experimental evaluation in a system with three replicas that are subjected to different operational load generated by different number of clients. Clients are going to be emulated by YCSB [2]. Your evaluation will observe the resulting throughput and latency as observed by clients. Notice that in the experiments we will aim at having read operations that have strong consistency semantics, which means that they have to be executed in a slot of the state machine (and ordered using the agreement protocol used in that solution, either Paxos or Multi-Paxos). While you are not required to evaluate the system under faults or when adding replicas to it, your implementations should be correct if any of these events happen.

In the following the target protocol architecture (and their interfaces) for solving this phase of the project are presented (Section 2). The programming environment for developing the project is briefly discussed next (Section 3). Finally, the document concludes by providing some information on the delivery rules for this phase of the project (Section 4).

2 Solution Architecture

Figure 1 illustrates the layering of protocols that you will have to have to for phase 1 of the project¹. The figure also describes the interactions between the protocols, from which the **interfaces** of the protocols can be derived. These

¹for evaluation purposes this layering is *mandatory*.

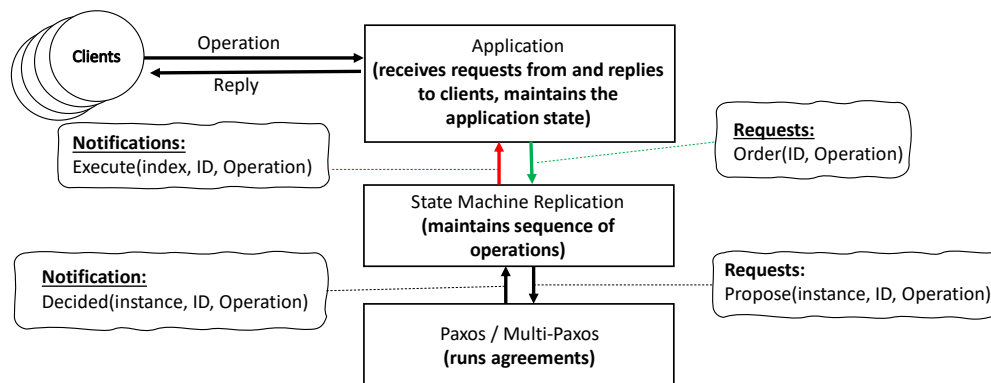


Figure 1: Architecture of a System Replica (and client interactions).

interfaces must be respected by your implementations, and the java project that you will receive will already provide the Babel events that are used to materialize these interfaces.

Every protocol that you will implement should have an equivalent interface to communicate with the other local protocols (this is materialized by Requests and Indications). This is essential to ensure that you can swap one protocol implementation for another and that your process can still execute correctly. Notice that the messages exchanged by each protocol that you develop with the (equivalent) protocol on other processes is not fixed. You should model these messages following the specification of the protocols that you are implementing. Notice that if you are careful in your implementation you can implement a single State Machine Replication protocol that is consistent with the use of either Paxos or Multi-Paxos.

Below the options for each of the relevant protocols is provided (with the references to papers when applicable).

2.1 State Machine Replication

TO BE COMPLETED.

2.2 Paxos

TO BE COMPLETED.

2.3 Multi-Paxos

TO BE COMPLETED.

2.4 Application

The application layer is going to be provided to you. It will have the code necessary to support all interactions to clients and it will use the prescribed interface to interact with the state machine replication layer.

2.5 Clients (YCSB)

To inject load in the system (i.e., execute operations) we are going to use the popular YCSB system. We will provide a driver that allows for YCSB to interact with the replicated application described above. Note that YCSB might send operations to any of the active replicas. Also YCSB executed multiple “client threads” where each one emulated an individual client. This is important because in the experiments we will want to study the differences between the latency (measured in mili-seconds) and throughput (measured in operations per second) as the total number of clients in the system increases. YCSB already computes and outputs both of these metrics for you.

You might need to run more than one instance of YCSB such that you can distribute the load of clients across multiple machines (a single physical machine has limits to the number of client threads it can execute concurrently without the clients becoming bottlenecked. The evaluation should saturate the servers and not the clients, as that would yield incorrect experimental results. If you rely on multiple instances of YCSB to run your experiments, you will need to (manually) combine the results outputed by each instance. To that end you should compute the average of the latency observed on each instance, and you should compute the sum of the throughput reported by each YCSB instance.

2.6 Evaluation

TO BE COMPLETED.

3 Programming Environment

The students will develop their project using the Java language (1.8 minimum). Development will be conducted using a framework developed in the context of the NOVA LINGS laboratory² written by Pedro Fouto, Pedro Ákos Cost, João Leitão, whose internal code-name is (still) **Babel**³.

The framework resorts to the Netty framework⁴ to support inter-process communication through sockets (although it was designed to hide this from the programmer). The framework will be discussed in the labs, and example protocols will be made available to students. Moreover the top layer that is responsible for injecting load in the system (i.e., propagate messages and receive them) is offered.

The javadoc of the framework can be (temporarily) found here: <https://asc.di.fct.unl.pt/~jleitao/babel/>.

The framework was specifically designed thinking about two complementary goals: *i*) quick design and implementation of efficient distributed protocols; and *ii*) teaching distributed algorithms in advanced courses. A significant effort was made to make it such that the code maps closely to protocols descriptions using (modern) pseudo-code. The goal is that you can focus on the key aspects of the protocols, their operation, and their correctness, and that you can easily implement and execute such protocols.

²<http://nova-lings.di.fct.unl.pt>

³We are still not happy with the name – for different reasons – so we are still accepting suggestions, although we might submit a paper about this on early December.

⁴<https://netty.io>

The course discussion board in Moodle can (and should be) used to ask for support in using or clarify any aspect on the operation of Babel.

While this is the second year that Babel is being used in this course, and the current version has also been used to develop several research prototypes, the framework itself is still considered a prototype, and naturally some bugs (or idiotic interfaces) can be found. We will address these as soon as possible. The development team is open to suggestions and comments. You can make such comments or improvement suggestions either through the Moodle discussion board or by e-mail to the course professor.

4 Operational Aspects and Delivery Rules

Group Formation

Groups should remain the same as in the first phase of the project.

The DI and NOVA LINCS research cluster

The technical specification of the cluster as well as the documentation on how to use it (including changing your group password and making reservations) is online at: <https://cluster.di.fct.unl.pt>. You should read the documentation carefully. The cluster is accessible from anywhere in the world through ssh. Be careful with password management. Never use a weak password to avoid attacks and intrusions on our infrastructure. Incorrect or irresponsible use of the department resources made available to students will be persecuted through disciplinary actions predicted in the School regulations.

All groups will have their quota in the cluster increased appropriately for the expected complexity of the project. Additional quota can be requested to the course professor.

Evaluation Criteria

The project delivery includes both the code and a written report that should have the format of a short paper. Considering that format, this document refers from this point onward as simply *paper*.

The paper should be at most 10 pages long, including all tables, figures, and references. It should be written with font size no bigger than 10pts and be in two column format. It is highly recommended (for your own sanity) that you use L^AT_EX to do this.

The paper must contain clear and readable pseudo-code for each of the implemented protocols, alongside a description of the intuition of these protocols. A correctness argument for protocol that was devised or adapted by students will be positively considered in grading the project. The written report should also provide information about all experimental work conducted by the students to evaluate their solution in practice (i.e., description of experiments, setup, parameters) as well as the results and a discussion of those results.

The project will be evaluated by the correctness of the implemented solutions, its efficiency, and the quality of the implementations (in terms of code readability).

The quality and clearness of the report of the project will have an impact the final grade. Notice however that students with a poorly written report might (accidentally) penalized on the evaluation of the correctness the solutions employed. This phase of the project will be graded in a scale from 1 to 20.

Deadline

Delivery of phase 2 of the project is due on 13 December 2020 at 23:59:59.

The delivery shall be made by e-mail to the address jc.leitao@fct.unl.pt with a subject: "ASD Phase 2 Delivery - #student1 .. #studentn".

The e-mail should contain two attachments: *i*) a zip file with the project files, without libraries or build directories (include src, pom.xml, and any configuration file that you created) and *ii*) a pdf file with your written report. If you have an issue with sending the zip file you can do one of the following: *a*) put a password on the zip file that should be 'asd2020' with no quotation marks; or *b*) put the zip file in a google drive, and send the public access link.

Project deliveries that do not follow these guidelines precisely may not be evaluated (yielding an automatic grade of zero).

References

- [1] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: An engineering perspective. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, page 398–407, New York, NY, USA, 2007. Association for Computing Machinery.
- [2] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, page 143–154, New York, NY, USA, 2010. Association for Computing Machinery.
- [3] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.
- [4] Leslie Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column)* 32, 4 (Whole Number 121, December 2001), pages 51–58, December 2001.
- [5] Robbert Van Renesse and Deniz Altinbuken. Paxos made moderately complex. *ACM Comput. Surv.*, 47(3), February 2015.
- [6] Rubbert van Renesse. *State Machine Replication with Benign Failures*, page 83–102. Association for Computing Machinery, New York, NY, USA, 2019.