IN5570/IN9570 – Distributed Objects
Home Exam 2

Department of Informatics
University of Oslo

April 22^{nd}, 2025

Introduction

The home exam is mandatory and will count $\frac{1}{4}$ to your final grade in the course.

Description

The assignment is about creating a framework for a simple replication of distributing objects with the help of Primary Copy Replication. The Framework should offer functionalities to replicate a given object to a given number of machines (nodes), and for a user program to access and update the replicated object via one or more replicas. To offer such function you could create an operation called `operation replicateMe[X,N]` where **X** is an object that should be replicated and **N** is the number of replicas to be maintained including the original object.

You are allowed to require that the given object has an operation `cloneMe[]` that returns a replica of the original object.

At the Primary Copy Replication, all updates must be performed on the Primary Copy. If a replica that is not the Primary Copy wants to do an update, the update should be delegated to the primary replica. How these objects address the Primary Copy and how a user program accesses a replica, is something you need to find out. Emerald has no built-in one-to-many reference mechanism.

In a classical Primary Copy Replication, when the primary replica has updated its state, make sure that any other replica will receive the updated information. You will be building a primary replica oriented framework that will keep track of replicas and which replica is the primary copy and what the state of the system is and what updates are currently being propagated. When new machines are detected, they become eligible to host replicas.
You are free to use any design pattern that you deem relevant. The Observer Design Pattern might be useful.
When a replica has crashed or disappeared, the framework should automatically generate a new replica. If the disappeared replica was the primary, the framework has to choose a new primary replica. How you exactly solve this is up to you; there are many solutions.
The framework should be able to tell an object which other replicas exist of a given object so that each replica does not need to keep track of all others.

Testing

You must also test the framework with at least two tests:

1. One where you will maintain a name server with a lookup operation with a given name, returns the matching object.

2. One where you will maintain a time server as the same as you did during your earlier assignment. The primary replica will maintain the time. If N is larger than the number of available machines, there should be a replica on each machine as far as it goes.

There will also be a requirement that the system be tested and run distributed using **PlanetLab**.

Additional Requirements for IN9570 Participants

IN9570 participants must additionally enable peers to access other peers in a order determined by the distance to the other peers. Distance must be measured as Round-Trip Time. You should discuss this issue briefly in your report.

Level of Ambition and Evaluation

It is important to note that a well-documented simple solution will be evaluated as better than an advanced solution that works only partially, or is poorly documented. Beware that main goal of the exam is building a distributed set of cooperating objects and that the replication is merely used here to illustrate the main goal. Thus you should not worry about creating the best or most fantastic replication scheme; simple solutions will suffice.

Delivery

1. A short (3 – 6 pages excluding illustrations and diagrams) report that contains:
    - An analysis of the most significant decisions you made regarding the program. First and foremost design decisions including any design patterns that you have chosen to use, if any.
    - A description of the main classes/objects in the program.
    - A description of how you have tested the program and why you have chosen to test as you do. For example, you should mention if you have tested the program in several parts and you should describe the resulting output briefly. Include an honest appraisal of how much of the program works.
2. The source code.
3. A Makefile, optional.
4. Output from your test runs.

The deadline for the assignment is **May** 21^{st}**, 2025 at 23:59:00 CEST (21:59:00 UTC)**.

Deliver your assignment in Inspera.

Last, but not Least

Good luck! And have fun! :)