INF5510 - Distributed Objects
Assignment 3
Home Exam 1

Department of Informatics
University of Oslo

March 2015

Introduction

The assignment is graded and will count ⅓ of your grade on this course. The assignment is also a requirement to fulfill to qualify for the final exam.

Description

The assignment is about creating a simple form of Napster in Emerald - our version will be called Nopester.

- Napster used music files (.mp3) - We will use Strings.
- Like in Napster, it requires that the files are immutable.
- The files must be localized on different peers

Implement Nopester using Emerald Programming Language.

Assumption

The program should have a central server that indexes where the different files are located. (Which Peers that has these given files) Each file has a name, but the system will identify the files with the help of one secure digest of the files content. You are free to implement whatever secure digest you want (MD5 or SHA-1(1) which is quite challenging to make). You can implement a secure digest, but it has to be either FNV-hash, BERNSTEIN or ADDITIVE-hash (you will not get any bonus for implementing MD5 or SHA-1, choose what you want to implement). It is a requirement that the hashing algorithm is implemented with the help of a **typeobject** to handle the so called «loose coupling ». Having typeobject makes sure that the hashing implementation can easily be changed out.

Peers

You will also need to implement the peers

- Each peer should be able to insert new files into the system. It shall do so by registering that it has a given file by sending name and the secure digest to the central server which then makes sure to insert data into a table (unless the file already exists) Moreover, the peer is inserted in the list of peers that has the specific file.
- A peer should be able to ask to be given the list of the peers that has the file with a given name. Moreover, peers may deliver the file on request.
- A peer will also be able to tell the central server that a given file no longer exists among peers.
- It is not necessary to store the files on disk, it is sufficient that the files are objects that lies in the memory.

You should also write a test object that initializes the system and generating a number of peers. Each peer must create a number of files and not least request files from other peers. Finally, dump out the entire system state. You will have to build some suitable example.

In principle, each peer and the central server should be running on different machines, but it is sufficient to use fewer machines. So you must have at least peers distributed over two machines, while the server runs on a dedicated machine. Nopester will also need to have a notification implemented where it notifies if a peer goes down. Tips here would be to use unavailable and/or additional processes that keep an eye on which machines are up by repeated calls of `getActiveNodes`

Planetlab

You will need to run your program using Planetlab.

Tests

Write a suitable test example. Use for example 10 files and 5 peers with a suitable number of replications.

Delivery

1. A short (3 - 6 page) report that describes:
   - An analysis of how your program is and why you have designed it (create some figures to illustrate your idea of your program).
   - A description of the most important classes/objects in the program.
   - A description of what you have tested and why you have chosen to test it. You should for example mention that you have tested the program in several parts and how you run the program

  - If possible, create a Makefile if you choose to have several files for your assignment

2. source code

3. output file

The deadline is **16th April 2015 23:59**. Where you will deliver your assignment in devilry - let the zip/tar file have the name <username>.<zip/tartype>

We wish you good luck!