IN5570 – Distributed Objects
Home Exam 1

Eric Jul
Department of Informatics
University of Oslo

March 11th, 2021

Introduction

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 – which Strings are.
- The files must be located on different peers.

Implement a simple version of Nopester using the Emerald Programming Language.

Assumption

The program should have a central server that indexes where the different files are located, i.e., which Peers that have the files. Each file has a name, but the system will identify the files with the help of one secure digest of the files content. The server should be able to return a list of files that match (or contain) a given name. It should also be able to return a list of locations for a given file where the file is identified by its hash. It should also allow the peers to provide an update with a file has, an pointer to the peer, and a name (a string) for the file.

You can implement a simple digest, for example, FNV-hash, Bernstein's djb2 or ADDITIVE-hash (you will not get any bonus for implementing MD5 or SHA-1). It is a requirement that the hashing algorithm is implemented with the help of a **typeobject** to enable *loose coupling*, so that the hash function can be replaced easily.

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 is available from the peer.
- It is not necessary to store the files on disk, it is sufficient that the files are objects that lie in the memory.

You should write a test object that initializes the system and generating a number of peers. Each peer must create a number of files and not the least request files from other peers. Finally, dump out the entire system state. You will have to build some suitable example. Make sure that they are large enough to show that your system works, but no larger.

Each peer and the central server should be running on different 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. Tip; use unavailable and/or additional processes that keep an eye on which machines are up by calls to `getActiveNodes`.

Planetlab

You must run your program on Planetlab.

Tests

Write a suitable test example. Use, for example, 10 files and 5 peers with a suitable number of replications. Peers must be located on physically different Planetlab machines at least 50 km apart.

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.

Delivery

1. A short (3 - 6 page) report that describes:
    - An analysis of your program design and why you have designed it (create some figures to illustrate the main ideas 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 from you most important test cases.

Deadline

The deadline is **14th of April 2021 23:59:00 CEST - local Oslo time**. When you deliver your assignment in devilry—let the zip/tar file have the name <username>.<zip/tartype> You can als add a version number, if you want, as to be able to upload different versions.

Grading

The assignment is graded and will count ¼ of your grade for this course. The assignment is compulsory and thus also a requirement to qualify for the final oral exam.

And finally do not forget ...

... have fun!