Distributed Systems
Autumn semester 2016
Prof. Peter Kropf, prof. Pierre Kuonen,
Assistants: Veronica Estrada, Andrei Lapin

Project 3

Message exchange

Due: 20 December 2016, 13:00

Description

We provide you a simple coin exchange program that has been implemented in Java (see file *CoinExchanger.java*). The program uses *JGroups* as a toolkit for reliable messaging. When you launch multiple instances of the program on a single machine they will automatically discover each other, group up and start exchanging coins. The program terminates in two cases:

1. It does not have any coins left;
2. It collected coins of all members previously connected to the group.

Preparation

For this project, there is no requirement for the execution environment. You can use any available environment of your choice (e.g., your local machine, remote cluster). The source code of the project (i.e. *project3.zip*) is available on *Ilias* and inside the *shared_public* directory on the cluster machines. Copy the archive to your working directory and unzip.

Executing the program

From the unzipped directory run the following command:

$java -cp .:jgroups-3.6.6.Final.jar CoinExchanger

Work to do

1. Implement the Snapshot algorithm [1] to determine how many coins are in each process at a given time.

2. In some program runs, a process decides to terminate but prints ”I have 1 coins”. How is this possible to happen, if the program only decides to end when it has zero coins? Implement a distributed termination algorithm to fix this.

3. Write a separate test program, in which a given member multicasts a fixed number of messages. Run it with different numbers of members and different numbers of messages sent. Plot a graph and estimate the complexity of the multicast algorithm used (regarding the number of messages sent). You may want to look into JGroups and its documentation to see if you can find the algorithm used, to corroborate your experimental results.

This time, you are not provided with a strictly defined structure for your reports. However, you are expected to organize the content in meaningful sections yourself. Do not forget to include all the relevant details of your work in the report (e.g., implementation details, brief algorithm description, execution results, plots, a small README or HOWTO with clear execution instructions for your programs).

Submission

The following documents must be uploaded on ILIAS:

1. A pdf version of your report (max. 12 pages!). Each student has to implement his/her own version of the report . This report should contain all your interesting results (in table and/or plot formats), as well as a pertinent analysis and evaluation.

2. A zip file containing all the source files of your programs.

Please respect the following point:

- Make sure that your compressed document is of reasonable size (no need for images of the highest resolution) and upload it on ILIAS by strictly following the naming conventions.

Note that the fact that you strictly followed all the instructions concerning the way to provide your report and results (formatting, type of file, deadline,...) will be taken into account to establish your mark.

References

[1] Chandy, K. Mani and Lamport, Leslie, *Distributed Snapshots: Determining Global States of Distributed Systems*, ACM Trans. Comput. Syst., Feb. 1985,