Raft consensus algorithm

(Addendum)

Author: Joan Manuel Marquès

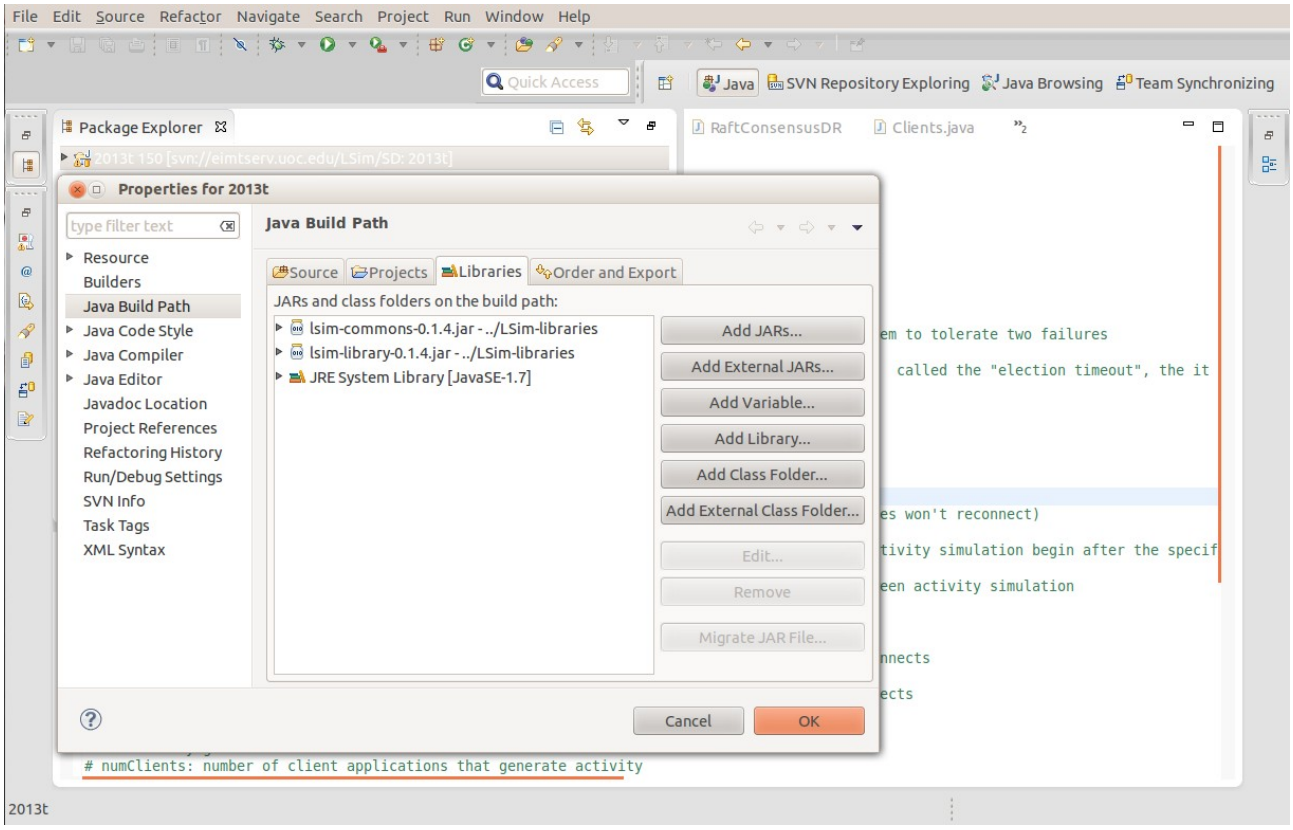
Distributed Systems course

Autumn 2013

1. New version of the practical assignment code

1.1. Download the new version of the code of the practical assignemtn

- Download the new version of the provided code of the practical assignment.
- Substitute `RaftConsensus.java` class included in the downloaded code for your implementation of this class.
  - Add any other auxiliary class that you created.
- Download LSim libraries (`lsim-commons-0.1.4.jar` and `lsim-library-0.1.4.jar`) from http://dpcs.uoc.edu/projects/lsim/files/sdlab-libraries
- Add LSim libraries to your eclipse project
  - Select the project of the practical assignment.
  - Go to menu *Project -- Properties -- Libraries – Add External JARs …*
  - Add following libraries:
    - `lsim-commons-0.1.4.jar`
    - `lsim-library-0.1.4.jar`



1.2. Changes on the signature of methods `requestVote` and `appendEntries` of `RMIsd` class

The signature of methods `requestVote` and `appendEntries` from `RMIsd` (package `communication.rmi`) has changed: destination type is `Host` instead of `String`. New signatures are:

```
public AppendEntriesResponse appendEntries (
            Host destination,
            long term,
            String leaderId,
            int prevLogIndex,
            long prevLogTerm,
            List<LogEntry> entries,
            int commitIndex
            ) throws Exception{


public RequestVoteResponse requestVote(
            Host destination,
            long term,
            String id,
            int lastLogIndex,
            long lastLogTerm
            ) throws Exception {
```

1.3. RaftConsensus class

Add the following two methods to your implementation of the `RaftConsensus` class (package `recipesService.raft`): necessary to check the correctness of the Server state.

```
public long getCurrentTerm() {
      return persistentState.getCurrentTerm();
}

public String getLeaderId() {
      return /* id of the leader in current term; “” if no leader */
}
```

1.6 Phases

We added phase `4.1`, `only4.1` and `4.2`:

> **`4.1`** : generates standard simulated log replication activity (phase `3`) and, in addition, specific activity to test phase `4.1`.: Formal evaluation will be done in this mode.
>
> **`Only4.1`**: simulated log replication activity only generates speTcific activity to test phase `4.1`.
>
> **`4.2`**: like phase `4.1` but you are allowed to modify running parameters to test your solution under different conditions.

Specific activity to test phase `4.1`

(According to section **8 Client Interaction** of the paper) We do two specific tests to check that *if it receives a command whose serial number* (timestamp in our case) *has already been executed, it responds immediately without re-executing the request*:

Test 1. (First) a `recipe` is inserted by an `AddOperation` with timestamp `timestamp0`; (second) this `recipe` is removed by the `RemoveOperation` with `timestamp1`; (third) a new `AddOperation` with `timestamp2` adds again the same `recipe;` (finally) the `RemoveOperation` with `timesamp1` is re-issued again to

remove the `recipe`. Raft shouldn't re-execute it (i.e. even though the `recipeTile` of the `RemoveOperation` with `timestamp1` refers to the `recipe` inserted by the AddOpeartion with `timestamp2` it shouldn't be removed because the `RemoveOperation` with `timestamp1` refers to `AddOperation` with `timestamp0` and not to the `AddOperation` with `timestamp2`)

```
hardSend(new AddOperation(recipe, timestamp0));
hardSend(new RemoveOperation(recipeTitle, timestamp1));
hardSend(new AddOperation(recipe, timestamp2));
hardSend(new RemoveOperation(recipeTitle, timestamp1));
```

Test 2. Re-sending the `AddOperation` of a removed recipe. Second add shouldn't be executed.

```
hardSend(new AddOperation(recipe, timestamp0));
hardSend(new RemoveOperation(recipeTitle, timestamp1));
hardSend(new AddOperation(recipe, timestamp0));
```

The whole code implementing the generation of code for phase 4.1 is in class `Clients` (package `recipesService.test.client`)

1.5. Renamed classes

`SimulationData` was renamed to `ActivitySimulation`.

2. Testing the practical assignment interacting with teacher's implementation

Teacher's implementation is running in an **Amazon** server. Use it to **test your solution interacting with teacher's implementation**.

Remember that the **formal evaluation** of your practical assignment will be done using **sdlab** (explained in next section). Before sending your practical assignment to sdlab you can test your implementation interacting with the teacher's implementation using the testing service deployed in Amazon environment.

These are the steps to run your implementation interacting with teacher's implementation deployed in Amazon:

Step 1. Upload your implementation in your Amazon machine

Step 2. Run your implementation from the scripts folder of your Amazon machine:

```
./start.sh 3 -phase 2 -p 20000 -h 10.164.23.174 --remoteTestServer
```

Change phase for the appropriate phase (values are: `2`, `3`, `4.1` , `only4.1` or `4.2`)

Number of Servers must always be 3

Check that the `groupId` field in `config.properties` contains your groupId.

Valid values for argument `phase`: `2`, `3`, `4.1`, `only4.1` and `4.2`

Step 3. Check the results of your executions in the following web pages:

`http://54.204.42.83:8080/`<groupId>

Summary information of all your executions

`http://54.204.42.83:8080/`<groupId>`.data`

Detailed information of all your executions

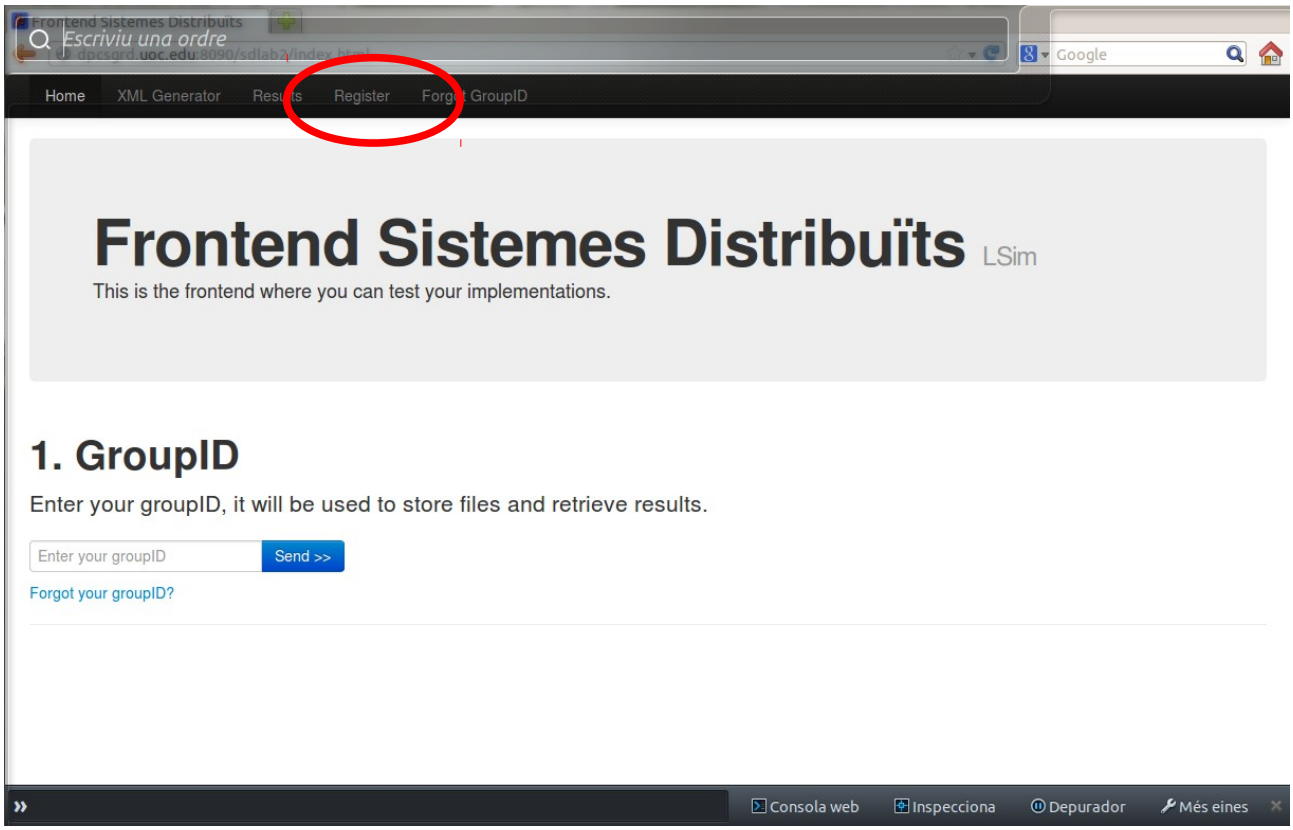
3. Evaluation of the practical assignment

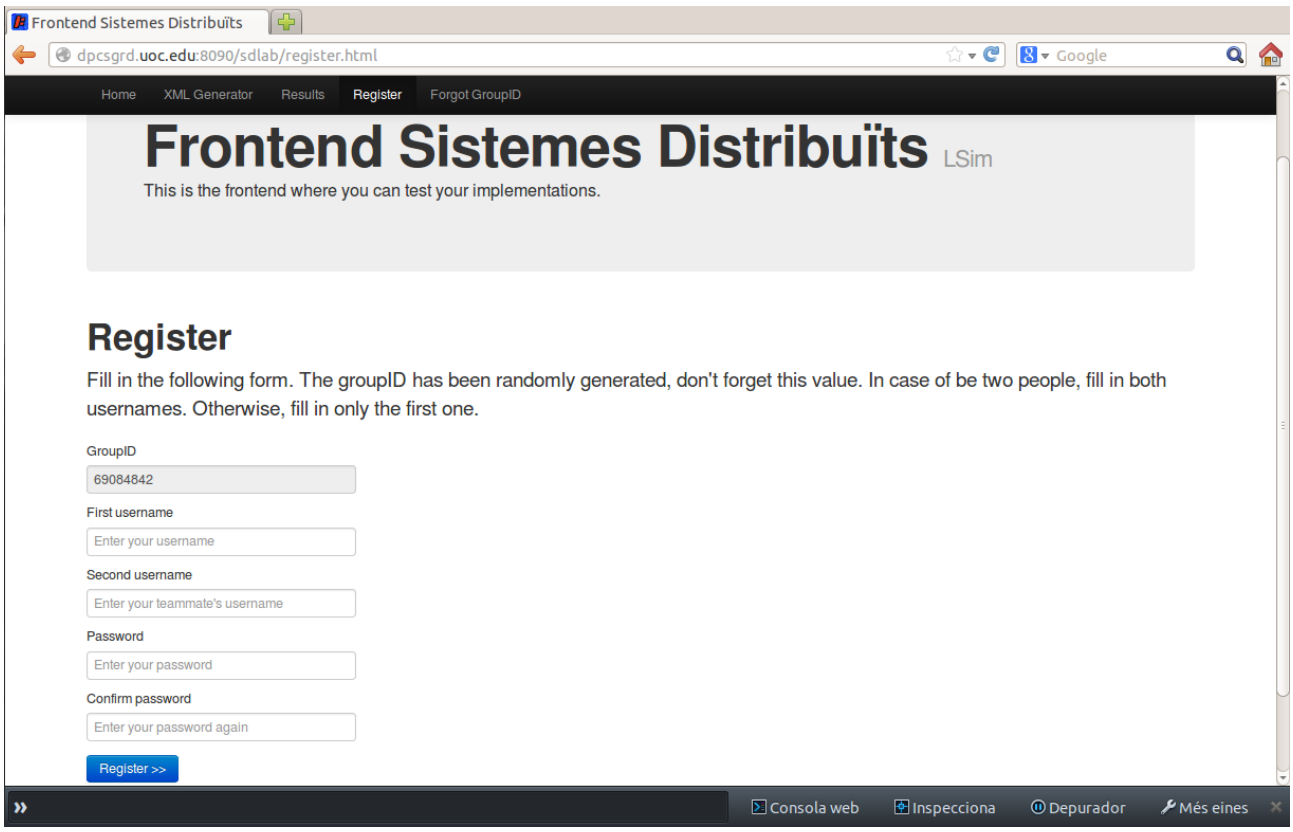
Formal **evaluation** will be done in **sdlab** (http://dpcsgrd.uoc.edu:8090/sdlab/).

The evaluation will be done in a realistic deployment (a cluster): three nodes will run instances of your implementation and two nodes instances of the teacher's implementation of the practical assignment.

Step 1. Register to sdlab

- Go to sdlab: http://dpcsgrd.uoc.edu:8090/sdlab/
- You are **strongly advised to do the phases 2 to 4 in groups of 2 students**, even though it is also possible to do it individually.
- Register both members of the group (or only you if you do the practical assignment individually).



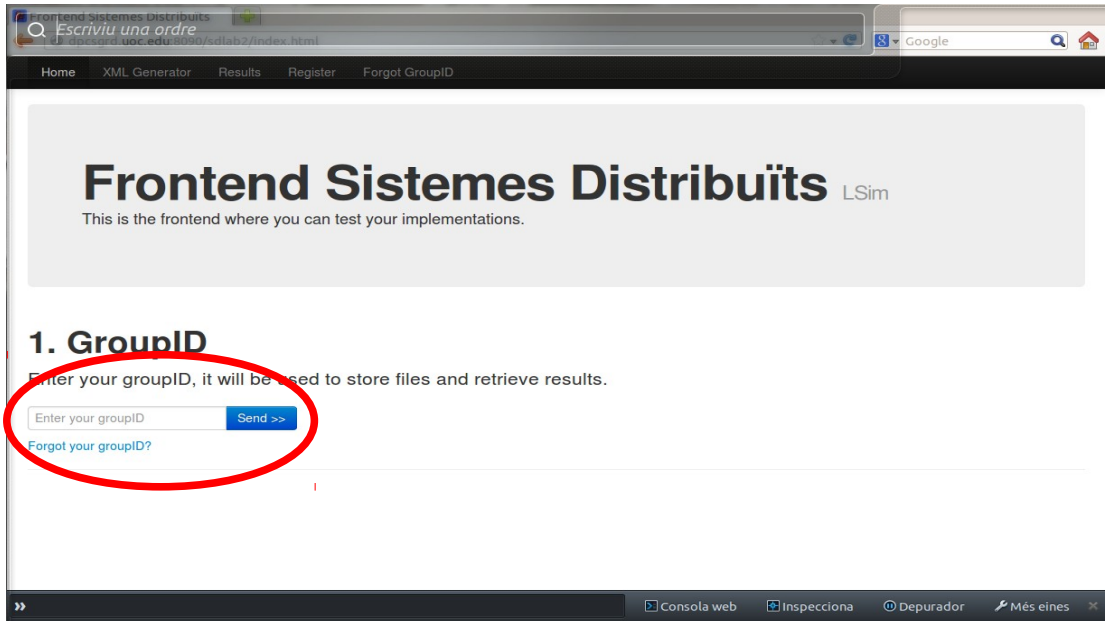


In case you forget your groupId your will always be able to get it introducing your username and password in "**Forgot GroupID**" option from the sdlab option's bar.

Step 2. Prepare the submission of your practical assignment

2.1 Introduce your groupId

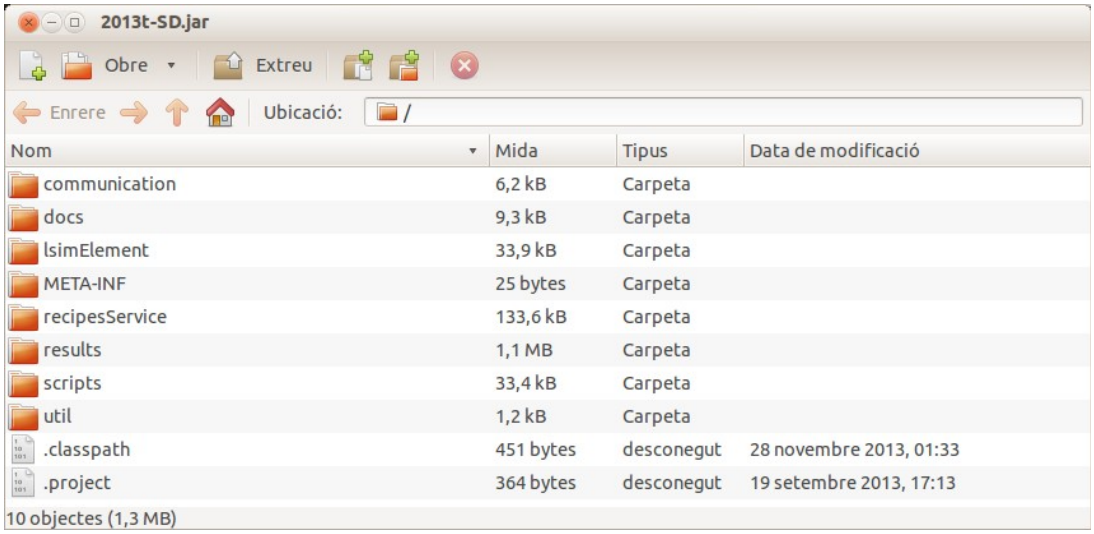
- Introduce your groupId and press "**Send**".
- Only registered users will be able to send experiments to sdlab.



Step 2.2. Submit your solution of the practical assignment

Step 1. Create a .jar file named **`2013t-SD.jar`** (**OTHERWISE WILL NOT WORK**) of the practical assignment code that contains all your created code.

- To generate a .jar file using eclipse: *File -- Export … -- Java -- Jar File -- Next --* (follow the following steps until the .jar file is created)
- The generated .jar file of the whole project should be something similar to:



Step 2. Create a zip file named **`server.zip`**.
IMPORTANT: `server.zip` MUST HAVE ONLY `2013t-SD.jar` file (without any folder)

Step 3. Submit for evaluation

- Select the phase to be evaluated.
  - Tests are incremental, i.e. the correct evaluation of a phase also evaluates that all previous phases are correct.
- Press "**Run**".
  - When preparing the execution you will see: "*Starting experiment...*"
  - When execution starts running it will show the execution id.

    (e.g. *ExperimentId: 2013t-SD[20131208022132])*
  - Evaluation will last for around 8 to 10 minutes.

Step 4. Check the result of your current submission

Step 4.1. Get the result of current submission

- Press "**Get results**" button.
  - Note that the evaluation of the practical assignment will last from 8 to 10 minutes.
  - After this time, if your execution hasn't finished retry the submission. Servers fail randomly and it may happen that at the end of an execution there aren't enough Servers connected.
  - If a second execution doesn't finishes, check your solution or notify it to your lab tutor in case the self-evaluation environment is not working properly.

Step 4.2. Get the result of all your submissions

- Select “**Results**” option from top bar menu of sdlab.
- Introduce your groupId