Distributed System: Assignment 0
Standalone Database

Petru Eles, Sergiu Rafiliu, and Ivan Ukhov
{firstname.surname}@liu.se

January 14, 2013

1 Introduction to the Programming Project

Welcome to the course! Over six programming assignments, from zero to five, you will learn the structure of the major components of a distributed system [1] and the interactions between these components. To this end, you will implement your own distributed system, namely, a distributed database of short messages known as fortunes [2]. The final configuration of the system, which you will obtain by the end of the fifth assignment, is depicted in Figure 1. The system is composed of a number of servers/peers (SP), maintaining copies of the database (DB) and ensuring that these copies are consistent across all the peers using a distributed locking mechanism (L), and a number of clients (C), interacting with this peer-to-peer network in order to perform various operations on the data. The participants of this network discover each other through a so-called name service (NS); for clarity, these communications are not shown in Figure 1. In each assignment, you will make a step towards the outlined goal by focusing on a particular component of the final system:

0. Standalone Database — you will build a non-distributed database;
1. Client-Server Database — you will make your first attempt of distributing the database by considering the client-server model;
2. Middleware: Object Request Brokers — you will learn the importance of name services and implement a mechanism for interaction with them;
3. Middleware: Peer-to-Peer Communications — you will develop means of interaction between peers arbitrary joining and leaving the system;
4. Middleware: Distributed Locks — you will implement mechanisms of distributed mutual exclusion to protect the data from concurrent operations;
5. Client-Server Database with Replicas — you will combine all the above components to produce a distributed database with data replication to ensure the efficiency and robustness of the overall system.

For each assignment, you will be given a skeleton of the code that corresponds to a particular part of the system, and your task will be to understand what the code does and to complete the implementation. The programming language

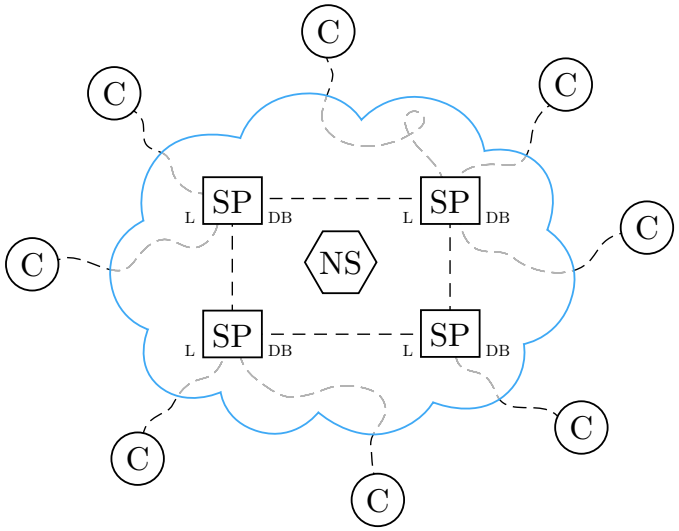

Figure 1: A distributed database with a number of servers/peers (SP) and a number of clients (C). Each server maintains a copy of the database (DB) protected by a distributed lock (L). The discovery process is facilitated by a name service (NS).

that we shall be using throughout the course is `Python` [3], more precisely, the second version of `Python` [4]. Even if you are not familiar with `Python`, the language is straightforward to learn, and the source code, provided to you, with a little bit of reading [5, 6] will be enough to readily get started and to successfully finish this programming project. The above-mentioned skeletons for the assignments and the corresponding instructions (just like the one below for the zeroth assignment) are provided to you on the web page [7] of the course.

2 Introduction to the Assignment

The goal of the zeroth assignment is to introduce you to `Python`. Here, we consider a non-distributed system that consists of a number of standalone machines, each of which acts as a server and as a single possible client of this server at the same time. The scenario is depicted in Figure 2. The machines do not interact with each other and perform all the operations on their own copies of the data without any synchronization. In the following, we shall refer to such a standalone machine with the server/client functionality as simply `Client`.

3 Data Format

In all the assignments, the list of fortunes, *i.e.*, the actual data in our database, will be stored in a text file conventionally called `fortune.db`. Such a file will be provided to you along with the source code; however, you are free to come up with your own lists of fortunes. The format of `fortune.db` is rather straightfor-

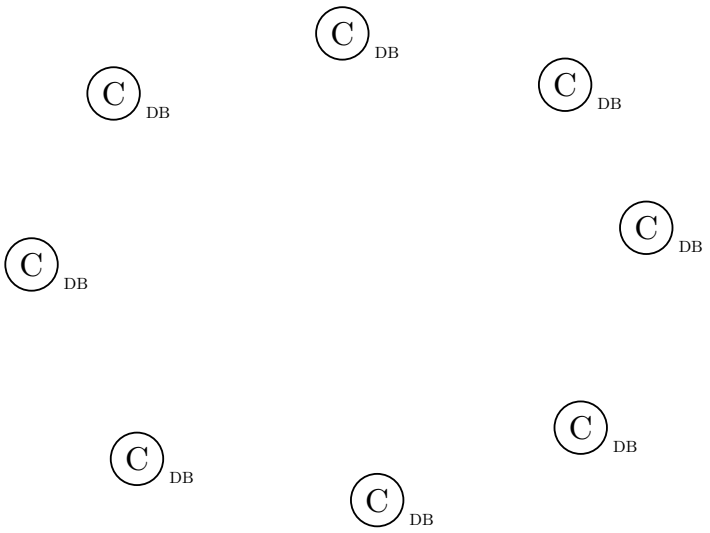

Figure 2: A number of standalone clients (C) with their own databases (DB).

ward: each message ends with a new line containing a single percent sign "%". For example, the following text contains two fortunes:

```
Everything should be made as simple as possible, but not simpler.
    -- Albert Einstein
%
According to the latest official figures, 43% of all statistics
are totally worthless.
%
```

Note that a message can have several lines.

4 Your Task

4.1 Preparation

Download [8] and extract the source code for the assignment. In the archive, you will find the following files, which you should read and understand:

- `lab0/client.py` — the main file of the application that each machine of our non-distributed system is running (no changes are needed);
- `lab0/dbs/fortune.db` — a list of fortunes (a thorough read and a deep understanding are not required here);
- `modules/Server/database.py` — a module with the class `Database` needed for `client.py` (should be modified).

In order to run the application, open a terminal window, change the current directory to `lab0`, and execute the following command:

```
$ python ./client.py -i
```

Following the menu of the program, you can try to read a random fortune from the database or to compose a new one:

```
Choose one of the following commands:
    r             ::  read a random fortune from the database,
    w <FORTUNE>   ::  write a new fortune into the database,
    h             ::  print this menu,
    q             ::  exit the application.

Command> w Take it easy.
Command> r
None
```

As you can see, the issued commands do not work since the code is incomplete.

4.2 Implementation

Your task is to fix the problem outlined in Section 4.1. Specifically, you should implement two functions in `database.py`: one is for reading random fortunes (the `r` command in the application menu) from `fortune.db` and the other one is for writing new fortunes (the `w` command in the application menu) into the same file `fortune.db`.

5 Conclusion

Having the assignment completed, you can now easily perform the reading and writing operations on the database. However, if you run several instances of the application on different machines (see Figure 2) or in different terminal windows on the same machine and try to add a new fortune in one of them, the other clients will not see the change. The reason is that the clients do not collaborate with each other and, therefore, do not try to keep their copies of the data consistent. In the next assignment, we will mitigate this problem by having a single server machine with an arbitrary number of pure client machines.

References

[1] http://www.ida.liu.se/~TDDD25/lecture-notes/lect1.frm.pdf.

[2] http://en.wikipedia.org/wiki/Fortune_(Unix).

[3] http://en.wikipedia.org/wiki/Python_(programming_language).

[4] http://docs.python.org/2.7/.

[5] http://docs.python.org/2.7/tutorial/index.html.

[6] http://docs.python.org/2.7/library/index.html.

[7] http://www.ida.liu.se/~TDDD25/labs/index.en.shtml.

[8] http://www.ida.liu.se/~TDDD25/labs/assignment0.zip.