Sistemas Distribuidos

Grado en Ingeniería Informática

Project (part 2 and 3)

Design and implementation of a peer-to-peer system

Félix García Carballeira
Francisco Javier García Blas

April 2, 2020

Content

1 Objective

The objective of this practice is that the student gets to know the main concepts related to remote procedure calls (RPC) and web services (WS).

2 Part 2 (RPC)

2.1 Description

In this part of the practice, the application developed in Part 1 is modified to add a service, based on RPC, which is responsible for storing all the information on users and content published by them. The scheme is as shown in Figure 1. In Part 1 all this information was managed directly by the developed server ("Server" in the figure). The objective now is that the storage of the information is done by a new RPC server (Storage Server). In this way, the server developed in Part 1 will take the role of server for the Java clients (User 1, etc.), and will be RPC client for the RPC server to be developed in this part.

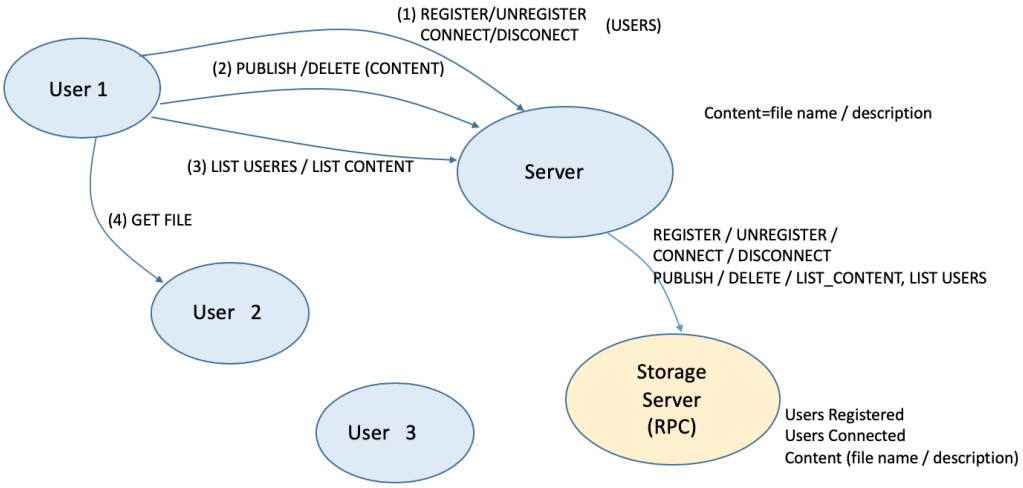

Figure 1: Peer-to-Peer system using RPCs.

All information about registered and connected users and published content must be stored in a persistent manner and managed by a server using the ONC-RPC model. This storage service must consist of at least the following services:

- Register a user.
- Unregister a user.
- Connect a user.
- Disconnect a user.
- Publish a file and its associated description.

- Get the list of users connected to the system.
- Get the list of files published by a user.

Each time the server made in the previous part receives a request using sockets, it routes it to the RPC server using the RPC calls that are defined. The protocol and the rest of the application do not change.

You are free to define the interface (`.x` file) you consider most appropriate. The storage of the information in the server must be persistent, either in simple files or databases, as it is being done in the first part. You are free to choose the solution you want. To obtain the list of users and files, dynamic lists can be used.

2.2 Programming language

For the development of this part, the C programming language and the ONC-RPC model will be used. The intermediary process ("Server" in the figure) is a client of the RPC service, while the developed ONC-RPC server offers the service.

3 Part 3 (Web Services)

3.1 Description

The objective of the third part is to develop and deploy a web service using JAX-WS. The service will be deployed, for simplicity, on the local machine where the client program developed in Java is executed, although it can be deployed on any other machine in the computer rooms. This web service will have a single operation consisting of converting a text sent by the Java client into another text in which all the words in the text will be in capital letters. This service will be used to transform the descriptions sent by the clients. More specifically, each client will use the web service to rewrite the description, changing everything to upper case before it is sent to the server (see Figure 2).

Once the server part has been developed, the client part that allows the web service to be invoked must be included in the Java client. Every time a user wants to publish a file and its description, he will invoke the web service to convert the text associated with the description to capital letters. Once converted to capital letters, the file and the description will be sent to the server as it has been done in Part 1 of the assignment.

3.2 Programming languages

For the development of this third part, the Java programming language and the JAX-WS API described in the course material will be used. Freedom is left to choose the interface and service definition that is considered most appropriate.

4 General rules

It is important to analyze the code of support provided with the project since it will be the starting point for the realization of the practice. The skeleton of the server and client program will be provided. These programs should be used as a starting point. All argument and command interpreter processing is already implemented.

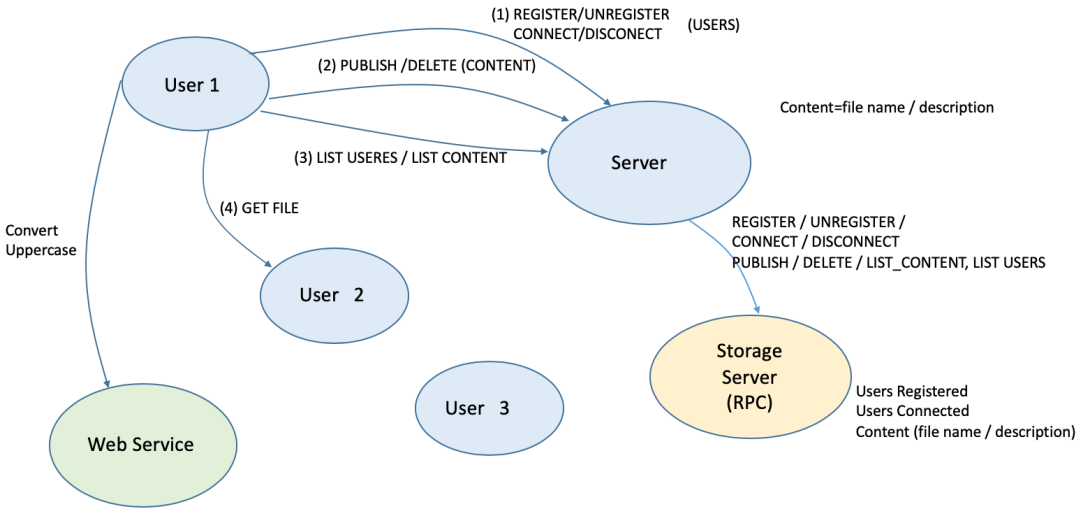

Figure 2: Peer-to-Peer system using JAX-WS.

1. Practices that do not compile or do not conform to the stated functionality and requirements will get a score of 0.
2. Practices that have warnings or are not commented upon will be penalized. A program that is not commented on will get a grade of 0.
3. The delivery of the practice will be done through the authorized deliverers. It is not allowed to deliver it through e-mail.
4. Special attention will be paid to detect copied functionalities between two practices. In case of detecting a copy, the University regulations will be applied and the students of the groups involved in the copy will have a 0 in the final grade of the course.
5. The practice must work in the computers of the computer classrooms (4.0.F16, 4.0.F18, 2.2.C05 and 2.2.C06) or in guernika.
6. The system must work with all system components running on different machines.
7. The protocol described must be followed carefully.
8. The memory must not exceed 10 pages in length.

5 Submission instructions

The practice will take place in groups of three students (maximum). The submission should only be given by a single member of the group in their teaching group. The same assignment should not be given repeatedly by all the members of the group.

The deadline for the entire assignment is **Sunday, May 17th.**

The submission will be made through Aula Global, through a deliverer who will be enabled for this purpose.

A compressed file must be delivered in zip format with the name **ssdd_p2_A_B_C.zip** where A, B and C is the NIA of the members who make the delivery.

The file have to contain:

- `authors.txt`, with the names and NIAs of the members of the group.
- `report.pdf`
- `client.java`
- `server.c`
- Text file that defines the RPC interface.
- `storageserver.c` file, with the implementation of the RPC storage service.
- Text file used to compile all the files.
- Java files necessary for the development of the text conversion web service.
- Text file named `README` with detailed instructions for the compilation and deployment of all the processes involved in the application.
- Any other file considered necessary for the compilation or evaluation of the practice.

The submitted `server.c` file must include the functionality of all completed parts. In particular, of the three parts if the development of the whole practice has been completed.

All the source files required for the compilation must be included and a text file named `README`, which will include detailed instructions for the compilation and deployment of all the processes involved in the application.

The assignment report should comment on the aspects of the practice development that it considers most relevant. Similarly, you can make any personal comments you consider appropriate. A document in PDF format must be submitted.

Do not neglect the quality of your practice report. Approval of the report is essential for approving the practice, as well as its correct operation. **If, when evaluating the report on your practice, it is considered that it does not reach the minimum admissible, your practice will be suspended.**

The report must contain at least the following sections:

- **Cover page** where the authors are listed (including full name, ISA and email address).
- **Table of contents**.
- **Code description** detailing the main functions implemented. Do not include source code of the practice in this section.
- **Description** of the way to compile and obtain the executable of all the processes involved. In addition, the way to execute them must be described.
- **Test battery** used and results obtained. More points will be given to advanced tests, extreme cases, and in general to those tests that guarantee the correct functioning of the practice in all cases. You have to take into account:

  - That the program compiles correctly and without texting if possible.
  - Avoid duplicate tests that evaluate the same program flows. The score in this section is not measured by the number of tests, but by the degree of coverage of the tests. It is better to have few tests that evaluate different cases, than many that always evaluate the same case.

- Conclusions, problems encountered, how they were solved, and personal opinions.

The following aspects relating to the presentation will also be scored:

- The report must have page numbers on all pages (except the cover).
- The text in the report must be justified.

5.1 Files preparation

The following steps must be followed to create the file to be submitted:

- You ahto to create the directory to prepare the materials to be delivered and check that it is in the delivery directory:

```
$ cd
$ mkdir ssdd_AAAAAAAAAA_BBBBBBBBB_CCCCCCCCC
$ cd ssdd_AAAAAAAAA_BBBBBBBBB_CCCCCCCCC
```

- Then you will proceed to copy all the files with the developed programs to the delivery directory and proceed to generate the zip file to be delivered:

```
$ cd ..
$ ls
... ssdd_AAAAAAAAAA_BBBBBBBBB_CCCCCCCCCC ...
$ zip -r ssdd_AAAAAAAAAA_BBBBBBBBB_CCCCCCCCCC.zip
    ssdd_AAAAAAAAAA_BBBBBBBBB_CCCCCCCCC/
```

6 Scoring

Only one delivery should be made which may contain the full functionality of all three parts, only two, or only one. The assignment will be qualified as follows:

- The first part of the practice will be scored on 7 points.
- The delivery of Part 2 will allow two points to be added to the first part.
- The delivery of Part 3 will allow one point to be added to the previous parts.

In this way, if only Part 1 is delivered, a maximum of 7 points will be obtained. Delivery of Parts 1 and 2 would earn up to 9 points, while delivery of Parts 1 and 3 would earn up to 8 points. The delivery of the three parts will allow to obtain up to 10 points. In any case, the delivery of the Part 1 will be mandatory.