

# 433-652: Distributed Systems – Principles and Paradigms

## Assignment 1 – Multithreaded Dictionary Server

---

### Problem Description

Using a client-server architecture, design and implement a multi-threaded server that returns the meaning of a word as stored in a dictionary.

This assignment has been designed to demonstrate the use of two of the fundamental technologies that have been discussed during the lectures:

- *Sockets*
- *Threads*

Hence, the assignment must make an **EXPLICIT** use of the two above. By explicit, I mean that in your application Sockets and Threads must be the lowest level of abstraction for network communication and concurrency.

### Implementation Guidelines

Design two components:

- a server, which maintains the dictionary. The server must maintain a dictionary file and a separate index containing the list of word and their meaning, and a support structure for speeding up the dictionary search respectively. The server is a process that keeps listening on a port and must serve multiple clients concurrently. Threads must be used to achieve this functionality. Suggested options for implementing the server are: thread per connection or thread per request. Please refer to "*Lecture 3 - Concurrency: Multithreaded Programming*" for the details about these two models.
- a client, which is used to query the server for the meaning of a given word. The client should implement a function that is used to query the dictionary with the following input/output:
  1. Input: string (word to search)
  2. Output: status code (found, not found, error) and String[] (meaning(s) of the word)

A possible example of the function is the following:

```
RemoteDictionary dictionary = new RemoteDictionary(address, port);  
  
Result result = dictionary.search("Assignment");
```

where:

RemoteDictionary is the class implementing the access to the remote dictionary and the search method.

Result is the class containing two properties (int errorCode, String[] meanings).

A call to the search method results into the client application establishing a connection with the server by opening a socket on <address, port>. The specific protocol used to query and exchange data with the server is left to the student to decide, it is important to notice that the communication has to be **RELIABLE**. This means that even though both TCP and UDP are allowed for implementing the assignment, in case of the use of UDP it is necessary to provide an infrastructure allowing reliable communication.

## Error Handling

It is expected that, on both the server and the client side, errors (by means of exception handling) are properly managed. The errors include the following:

- input from the console for what concerns the parameters passed as command line
- network communication (broken connection, address not reachable, bad data...)
- I/O to and from disk (cannot find the dictionary file, error reading the file, etc...)
- other errors you might come up with

The application will be tested and validated against all these errors.

## Implementation Language

The assignment can be implemented in one of the following languages:

- Java
- C#
- C++

It is **NOT ALLOWED** to use any of the following support libraries or methods: RMI (Java), Remoting (C#), RPC(C++), but an explicit use of sockets should be made. For what concerns concurrency it is **EXPECTED** to use threads.

There is no privileged choice of language, please use the one you are more proficient and comfortable with. Mind that in the lectures I have shown examples with Java and C# (more will be posted on the LMS for C# and Java). This does not make Java or C# preferred, but the fundamentals, at least for the Java programming language, have been given so that the assignment can be accomplished.

## Final packaging and delivery

The assignment will be composed **AT LEAST** by two executables (.class in case of Java) one for the server and the other one for the client (DictServer and DictClient). In case the implementation is done in Java the following is expected:

```
server: java DictServer <port> <dictionary-file> <dictionary-index>
```

```
client: java DictClient <address> <port> <word-to-look-for>
```

For what concerns the server, the last two parameters may vary according to the implementation of the dictionary chosen. For example, you might want to use a more complex structure than two files. In this case a configuration file for the dictionary might be more suitable. The configuration file only relates to the dictionary part of the parameters.

A report, describing:

- the problem context in which the assignment has been given
- a brief description the components of the system
- an overall class design and an interaction diagram
- a critical analysis of the work done followed by the conclusions

is expected. Please mind that a report is a **WRITTEN** document, do not put only graphs. A report without any descriptive text addressing the problem and the analysis of the work done will not be considered valid.

The length of the report is not fixed. A good report is auto-consistent and contains all the required information for understanding and evaluating the work done. Given the level of complexity of the assignment a report in the range of 4 to 8 pages is a reasonable choice for describing with the sufficient level of details what is needed in the report. Please mind that the length of the report is simply a guideline to help you in avoiding extremely long and meaningless report and written works with insufficient information.

It is important to put your details (name, surname, student id) in:

- the head page of the report
- as a header in each of the files of the software project

This will help to avoid any mistakes in locating the assignment and its components on both sides.

## Deadline

The assignment must be handed to the lecturer at the following email address: (csve@unimelb.edu.au). A zipped file containing the following directory structure is expected:

- <directory> *src* (containing all the source code and eventually instruction on how to compile it)
- <directory> *report* (containing a pdf file)
- <file> *readme.txt* (if you need to give additional instruction information)

The zip file must be named with your Student ID. (<student-id>.zip).

The deadline for submitting the assignment is: **Tuesday 31 August (5 pm)**.

## Marking

The marking process will be structured by first evaluating whether the assignment (application + report) is compliant with the specification given. This implies the following:

- use of sockets and threads for the application;
- proper use of concurrency in the server;
- proper handling of the errors;
- a report with the requested content and format;
- timeliness in submitting the assignment in the proper format (names, directory structure, etc..).

Should all of these elements be addressed, the assignment will be evaluated 60% of the whole mark, that is 6. The rest of the mark is devoted to two elements:

- *excellence (3 marks - 2 for code, 1 for report)*. What does excellence mean? The assignment has not only be implemented by using the minimum requirements but a smart design has been done, a proper interaction with the user has been provided (notification of errors, which really help to understand what went wrong) and most importantly the code documentation! For what concerns the report excellence means that the report is flawless and well written with the proper graphs and a complete discussion and analysis of the code implemented (i.e.: scalability with tests across different dimensions such as number of concurrent connections, size of the dictionary).
- *creativity and plus (1 mark)*. Any additional implementation not required such as a GUI for managing the client or the server or other enhancement of the code introducing advanced features for the research will be considered a plus.

**NOTE (EXTREMELY IMPORTANT):** the excellence and the creativity and plus marks (4 marks) cannot compensate any lack in the first part (the one that scores up to 6 marks). Please concentrate your efforts in getting the first 6 marks done and then proceed with the rest. You can easily replicate the test that will be performed by the lecturer during the demonstration.

## Demonstration Schedule and Venue

The student is required to provide a demonstration of the working application and has a chance to discuss with the lecturer the design and implementation choices made during the demo. By that time the lecturer has already read the report and the student will have 15-20 minutes to provide a complete demonstration of the working system.

You are not allowed to make any change to the report and the code in the period of time lasting between the submission of the assignment and the demonstration day.

Two days will be scheduled for demonstrations in order to accommodate everyone:

- Thursday 2 September 2010
- Friday 3 September 2010

The venue is Lab 2.17 (level 2, at the right corner of the building entering from the main entrance). You are free to develop your system where you are more comfortable with (at home, on one pc, on your laptop, in the labs...) but the assignment is meant to be a distributed system that works on at least two different machines in order to separate the client from the server. **PLEASE**, once you have completed the assignment (at least the first 60%) come to the university labs and check it there. Lab 2.11 is accessible for this purpose.

After the submission of the assignment, a schedule for the demonstration will be published on the LMS website. And a booking process for a slot will start in order to have a complete schedule. (I will basically start filling Friday and then proceed to Thursday if someone is not able to come on Friday, if someone of you cannot come in the assigned slot we can arrange a different one).