COMP90015: Distributed Systems – Assignment 1 – Multithreaded Dictionary Server – 10 marks

Problem Description

Using 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* - ***TCP/UDP Sockets*** ***(If you were aware of TCP before joining this subject and had experience in TCP sockets programming, then you are recommended to use UDP)***
- *Threads* – ***Multithreading based on worker-pool architecture***

Note: In worker-pool architectures, the server creates a fixed pool of worker threads and uses them to process clients' requests.

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 (but you are most welcome to come out with your own design/ interfaces)

Design two components:

- a server, which maintains the dictionary. The server must maintain a dictionary file containing the list of words and their meaning, and a support structure for speeding up the dictionary search respectively. The server must serve multiple clients concurrently. Threads on a worker-pool must be used to achieve this functionality.
- 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 to 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. However, it is important to notice that the communication has to be **RELIABLE**. This means that an infrastructure allowing reliable communication over UDP has to be implemented.

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 (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 must be implemented in **Java**. Utilization of technologies such as RMI and JMS are **not allowed**.

Final packaging and delivery

1. At least two class files, one for the server and one for the client (DictServer and DictClient), are expected. They must be executed by invocation of the following commands:

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

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

In case of 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.

2. A report, **in PDF format,** 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

Please mind that the report is a **WRITTEN** document, do not put only graphs. A report without any descriptive text addressing the problem, architecture, protocols, 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. Please mind that the length of the report is simply a guideline to help you in avoiding an extremely long or insufficient report.

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 at the following email address of **Rodrigo**: (**rnc@unimelb.edu.au**). A compressed file containing the following directory structure is expected:

- <directory> *src* (containing all the source codes)
- <directory> *report* (containing the pdf file)
- <file> *readme.txt* (if you need to give additional information)

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

The deadline for submission of the assignment is: **Tuesday 27 August (5 pm)**.

Note: You need to submit printed report during demonstration.

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

For all of the above elements, the assignment will be evaluated 60% (6 marks) of the whole mark. The rest of the marks are assigned to two elements:

- *excellence (3 marks - 2 for code, 1 for report)*. 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 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 and size of the dictionary).
- *creativity and plus (1 mark)*. Any additional implementation such as a GUI for managing the client or the server or other enhancements to the code introducing advanced features (such as processing of user requests within a server in scalable, reliable, and efficient manner) 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.

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.

Note: You need to submit printed report during demonstration.

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

- **Group 1: Thursday 29 Aug 2013**
- **Group 2: Friday 30 Aug 2013.**

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.

After the submission of the assignment, a schedule for the demonstration will be published on the website, likely **to be from 1 to 4pm.** **Demo Venue: To be announced later.**

Further Info: If you need any clarification on the Assignment, kindly ask questions during the lectures, so that all students benefit from it. Any serious issue, please contact Rodrigo: (rnc@unimelb.edu.au).