L4 - TCP

1. Introduction

1.1 Objectives

This lab aims at consolidating the basic concepts required for programming distributed applications that use TCP communication.

We also expect you to gain some familiarity with the sockets API in Java for TCP communication.

2. Application

The application you shall develop is a variant of client-server application described in the first lab. The difference is that you should use TCP, rather than UDP.

Server

The server program shall be invoked as follows:

> **java Server <srvc_port>**
> where:
>
> > <srvc_port> is the port number where the server provides the service

The server program should have an infinite loop in which the server waits for a request from a client, processes the request and sends the respective response to the client.

To observe the operation of your solution, the server shall reveal what it is doing by printing messages with, e.g., the following format:

> **<oper> <opnd> * :: <out>**
> where:
>
> > <oper> should be "register" or "lookup", according to the operation invoked;
> > <opnd> * is the list of operands received in the request for the operation;
> > <out> is the value returned by the operation, if any.

Client

The client program shall be invoked as follows:

> **java Client <host_name> <port_number> <oper> <opnd> ***
> where:
>
> > <host_name> is the name of the host where the server runs;
> > <port_number> is the port number where the server provides the service ;
> > <oper> is "register" or "lookup", depending on the operation to invoke;
> > <opnd> * is the list of operands of the specified operation:
> >
> > > `<DNS name> <IP address>`, for `register`;
> > > `<DNS name>`, for `lookup`.

To observe the operation of your solution, the client shall reveal what it is doing by printing messages with the format used by the server, i.e.:

> **<oper> <opnd> *:: <out>**
> where:
>
> > <oper> <opnd*> and have the same meaning and format of homonyms arguments from the command line and
> > <out> is the value returned by the operation invoked or "ERROR" if any error occurs.

For fun: If your server is not able to serve **concurrent** client requests, modify it. To test this new version modify the client so that it sleeps for a time interval after receiving the server reply and before closing the socket. Allow for the sleep duration to be configured via the command line.

3 Documentation

- Java API for TCP communication.
- Oracle's (TCP) Socket's Tutorial