Distributed Systems 2013/2014
Programming exam   6th of January of 2014
Duration: 150 min
Consultation: any documentation installed in the computer and a "cheat sheet"
Public code: **ODG501** (http://sigex.fe.up.pt/)

1. Introduction

The exam consists in solving in Java or C the following problem.

2. Problem

Write a program that 1) collects values from temperature sensors (using the producer/ /consumer paradigm) and that 2) provides the average temperature to clients.
To facilitate the program writing and its assessment you should implement each of these functionalities in separate parts as follows.

3. Part 1: Client-server

Implement a server and respective client using **TCP** and the language you prefer.
The server should support a single operation:
average, which returns the temperatures average value (received from the sensors) and execute indefinitely, waiting for requests from clients and answering them.
The client should send a single request, wait for the server response and print it.
Both the server and the client should print in the standard output (`System.out/stdout`) messages indicating: 1) the sending of a message; 2) the reception of a message. The printed message should include the received / sent messages or respective summaries.
It is up to you to define the format of the messages exchanged between client and server.
Invoking the programs in Java The server should be invoked as follows:

```
java Server <port> <average>
```

where

`<port>` is the socket port where the server receives the clients' requests and which it will use to send the respective responses;

`<average>` is the value (float) that shall be returned in the responses (considering that this version does not include code to collect the temperatures).

The client shall be invoked in the following way:

```
java client <addr> <port>
```

where

`<addr>` is the server IPv4 address;

`<port>` is the port of the socket used by the server.

Invoking the programs in C The server and client programs should be named server and client, respectively. The arguments are similar to those of the corresponding programs in Java.
You should submit an ASCII file named `client-server.txt` containing the commands needs to compile your programs

4. Part 2: Producer-Consumer

In this part you should develop a consumer and a producer using multicast communication. (If you do not know how to use multicast, you can use UDP unicast communication with a 30% penalty in the corresponding assessment).

The producer must transmit in multicast the temperatures measured by a sensor. These measurements are passed as arguments in the program and should be sent with a period of approximately 1 second. (You can use `Thread.sleep()`/`sleep()`). After having sent all messages, the program should terminate.

The consumer should execute an endless cycle within which it receives the messages sent by the producer. Whenever the producer (consumer) sends (receives) messages it should print to the standard output the same messages or their summaries. It is up to you to define the format of the messages sent by the producer.

Invoking the programs in Java The consumer program should be invoked as follows:

```
java Consumer <addr> <port>
```

where

`<addr>` is the IPv4 multicast address to be subscribed by the consumer;

`<port>` is the port of the socket of the multicast group to be used by the consumer.

The producer program should be invoked as follows:

```
java Producer <addr> <port> <val1> ... <valn>
```

where

`<addr>` is the IPv4 multicast address subscribed by the consumer of the service;

`<port>` is the port of the socket of the multicast group used by the consumer.

`<val[1..n]>` is the value (type `float`) that should be sent in each of the messages).

Invoking the programs in C The consumer and producer programs should be named consumer and producer, respectively. The arguments are similar to those of the corresponding programs in Java. You should submit an ASCII file named `producer-consumer.txt` containing the commands needs to compile your programs

5. Documentation and Submission

The documentation of the JDK 1.6 API is available in the following URL:
`file:///opt/docs/`
To submit your solution, you should compress in a single `.zip` file all files that compose your solution, i.e., the files with the source code (if you used C then include the file with the compilation commands, too). **IMP**. The `.zip` file should contain files, only, not directories.