Distributed Systems 2014/2015
Programming exam 13th of January of 2015
Duration: 150 min
Consultation: any documentation installed in the computer
Public code: **YQA435** (http://sigex.fe.up.pt/)

---

1. Introduction

The test consists of the resolution in Java or in C of the functionalities necessary for the implementation of the Garcia-Molina's "bully" algorithm for electing a leader:

1- Failure detection, with a maximum rating of 16 (over 20) points.
2- Election of the leader, with a maximum rating of 20 (over 20) points.
Maximum rating of 18 points if using unicast communication instead of multicast.

2. Preliminary notes

Tip 1 - To measure time intervals you can use `Thread.sleep()/sleep()` (see `man 3 sleep`).
Tip 2 - To measure the timeout, you can use `SO_RCVTIMEOUT` (see `setSoTimeout() / setsockopt()`, and, if you choose C, `man 7 socket`).
Tip 3 - Use `System.currentTimeMillis() / gettimeofday()` to resume a timing interrupted by some event.

Implementation in C The names of the programs presented below follow the conventions used in Java. If you choose an implementation in C, you must follow the conventions of this language. So, in this case, the names are identical to those specified, except the first letter must be lower case and not uppercase. Command line arguments are identical to Java ones. If they opt for the implementation in C, they must also submit a **Makefile** that allows compiling your programs by invoking `make`. Alternatively, you can submit an ASCII file name `compilation.txt` with the commands for compiling your programs.

3. Failure detection

One of the features required by the "bully algorithm" is failure detection. In this paragraph you should implement a protocol based on the periodic sending of "**ping**" messages, i.e., a process, **monitor**, should periodically send a "ping" message to another process, **target**. This one should send a response for each received "ping" message.
Implement this protocol using **TCP** and your preferred language (Java or C).
The absence of a response or the interruption of the connection shall be considered failures of the `target`.
Both the `monitor` and the `target` must print to the standard output (`System.out/stdout`) indicating: 1) the sending of a message; 2) the receipt of a message. The printed information shall include incoming/outgoing messages or their summaries. It is up to you to define the format of the messages exchanged between client and server.

Invocation of programs in Java. The `Target` must be invoked as follows:

```
java Target <port>
```

where
`<port>` is the socket port where the `target` will receive ping requests, and which it will use to submit responses.

The `Monitor` should be invoked as follows:

`java Monitor <addr> <port> <period> <timeout>`

where

`<addr>` is the IPv4 address of the `target`;

`<port>` is the socket port used by the `target`;

`<period>` is the period of ping messages;

`<timeout>` is the timeout for target failure detection.

4. Election

In this point, you must implement a variant of the Bully election protocol with multicast communication (if using unicast UDP communication, the maximum rating will be 90%). The figure illustrates the algorithm that differs from the original in that the multicast of the ELECTION message allows to eliminate BACKOFF and HALT messages, i.e., a process that receives a message ELECTION of another process with a lower identifier shall cancel the execution of any protocol election that has started. On the other hand, if you are not in election mode and receive an ELECTION message you should switch to election mode.

Essentially, your program should perform an execution of the algorithm. This execution must be started after: the expiration of an initial timeout, or the receipt of an ELECTION message. To avoid an "avalanche" of simultaneous executions, the duration of the initial timeout must be uniformly distributed in the range `[0; <id> x <delay>]`, where `<id>` and `<delay>` are command line arguments (see below). To generate random numbers you can use `java.util.Random/random()` (see `man 3 random()`). To allow you to follow the execution of the protocol, your program must print it in the standard output all messages sent and/or received.

Tip- Try to implement the protocol as a state machine.

Program invocation in Java The program must be invoked:

`java Bully <addr> <port> <id> <T> <delay>`

where

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

`<port>` is the socket port of the multicast group to be used by the process;

`<id>` is the process identifier (to determine the leader)

`<T>` is an upper limit of the communication delay between 2 processes in **ms**.

`<delay>` is the time (in **ms**) that should be used, together with <id>, to determine random timings (see above).

5. Documentation and Submission

The Java API documentation is available in the following URL: `file:///opt/java-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, or makefile, too). **IMP**. The `.zip` file should contain files, only, not directories.