Distributed Systems Project, Spring 2014, First exercise

Select this assignment if (your student ID modulo 5 + 1) equals 3.

Assignment 3: Bully election algorithm

Write a program that implements the bully election algorithm.

Specifications

Command line interface:

```
program configuration_file line
```

where `configuration_file` is the name of the configuration file and `line` is the line of this client (see below).

The configuration file has an undetermined number of lines, each in the following format:

```
id host port
```

where `id` is an integer used in the manner described below, `host` is either the hostname or the IP address of the client and `port` is the port on which it is listening at that address. Your client program takes as argument one integer, which indicates its own ID, i.e., the line in the configuration that indicates what port this client should use. The client can ignore the hostname for its own line, but needs to use the other lines to know who are the other clients in the system. There is no upper limit to the number of lines in the configuration file. If the argument given to the program does not exist in the configuration file, your program is allowed to crash immediately.
NOTE: It's easiest to have all the clients run on the same machine during early development.
NOTE: You can develop your programs in any environment, but it must also be runnable on the Ukko cluster.

Running of the Program

Make sure all programs are running before having them start executing the algorithm. In the following, we use terms node and program interchangeably, since individual programs are intended to simulate different nodes.

The node with the largest ID is the initial coordinator. Other nodes ping the coordinator at regular intervals, roughly once per 5 seconds. If the coordinator does not answer, then the node must initiate an election as per the algorithm. Once a new coordinator is selected, nodes must start pinging that node.

The behavior of a coordinator is as follows. Once a node has been elected to be the coordinator, it will terminate its execution roughly 10 seconds after being elected. During this time it must be able to reply to pings from other nodes. The initial coordinator should terminate roughly 10 seconds after having been started. Eventually all programs should terminate.

Output Format

Each individual program should produce an output of its run. Use the following syntax for output:

- Coordinator terminating: `t i`, where `i` is the ID of the coordinator.
- Start of election: `e [v]`, where `v` is a vector of IDs to which the election message is sent. This is logged by any node starting an election.
- End of election: `c i`, where `i` is the ID of the new coordinator. This is logged by all nodes.

NOTE: Because we use partly automated tools for checking the output, your output must match exactly the format above. Do not add any other text or lines. Failure to comply will lead to a reduced grade.

Guidelines

You are free to choose any programming language, but we recommend using a higher level language, e.g., Ruby or Python, even if you have to learn the language from scratch during the assignment.

The actual contents of the messages are irrelevant, as long as you are able to implement the required functionality. No interoperability between groups is required so you are free to choose the format.

Deliverables

Program source code with documentation.

Timeline

The assignment is due on January 28th at 10:00. No extensions will be given.

Return

Return your code by email to Liang.Wang@cs.helsinki.fi as one tar-archive. Please indicate clearly your name and student ID in every source code file.