Distributed Systems Project, Spring 2015, First exercise

Select this assignment if (your student ID modulo 4 + 1) equals 4.

Assignment 4: Gossiping

Write a program that implements gossiping.

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.

Node with ID 1 starts a gossip. It selects 3 other nodes at random and sends them a message containing a unique gossip ID. Do not send the message back to the node where you got it from. Each of the receiving nodes selects 3 nodes at random, forwards the gossip to them, who in turn select 3 nodes at random, etc. If a node has already forwarded a given gossip message, it will not send it again.

Other nodes are free to start their own gossips after they have received at least one gossip message from another node. Nodes should send one gossip message when they receive their first message and thereafter send one gossip for every 5 messages they receive. Each node should start 10 gossips after which it should finish its execution. Given the probabilistic nature of gossiping, it is possible that some nodes are not able to finish properly. If this is the case, those nodes are

allowed to terminate as soon as they realize that proper execution is no longer possible.

Feel free to play with the numbers “3” and “5” in the above description to see how gossiping works in different conditions.

Output Format

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

- Receiving a message: `r i [v]`, where `i` is the gossip ID and `v` is a vector containing the IDs of nodes through which this gossip has passed.
- Sending a message: `s i [v]`, where `v` is a vector containing the IDs of nodes through which this gossip has passed, i.e., the ID of the sender for a new gossip and for “old” gossips the received vector with the sender appended.

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 20th 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.