Distributed Systems Project, Spring 2015

Lamport Clocks

Jonne Airaksinen, 013932592
19.01.2015

Program overview

The main program is writter with Node.js and is supplemented by a couple of Bash scripts for startup of the nodes and for killing unresponsive processes It uses a simple UDP server and client for communication between nodes.

In addition to this readme.pdf-file, the project contains the following files:

- lamport.js – The main Node.js file that is run on every Ukko-node specified in the config.
- startup.sh – A bash script used to startup the Node.js processes over ssh on Ukko-nodes specified in the file.
- config.txt – Configuration file containing the information of all nodes, and the hosts and ports they run on.
- README.md – Command line / GitHub friendly readme, containing the same information as this .pdf
- kill_processes.sh & ukkonodes – The bash script is used for killing unresponsive processes over ssh on all Ukko-nodes specified in the file ukkonodes. Helpful mostly for development.

Running the program

To start the Node.js processes on all specified Ukko nodes, just run the startup.sh-script. **Note that** if the contents of the config file changes, those changes must also be made in the startup.sh file. The Node.js processes are started over ssh and run concurrently on the Ukko nodes.

Handling departed nodes

As the exercise sheet specified, communication with a departed node should be handled in a ”graceful manner”. Sending a message to a departed node does not cause any problems for this implementation, since the sender doesn't wait for confirmation of arrival or a reply from the target node.

This of course means that some messages are just lost, but I took the freedom to leave this flaw into the program code. If done properly, a master node that keeps track of the active nodes could exists for this purpose, but I deemed this out of the scope for this exercise.

Program output

While running, each Node.js process prints its output to the command line, with each local event, sent message and received message on its own line. From the exercise sheet, I understood this was the wanted functionality and this has been implemented in the code.

However, if instead of this the wanted functionality was to print out the whole history of each process at the end of its run, this is also possible. In the lamport.js file there's a commented out console.log(history) on line 143. Uncommenting this causes a string containing every local event, sent message and received message to be printed at the end of each the run of each process.

If wanted single files per node were actually the wanted output, I suggest commenting out every line with a console.log call on it, except for line 143. Then the single string output is easily available for further analysis.

Known problems

During the development of this exercise, there were some issues with the reading of the config file. For some reason, a mysterious force sometimes added a newline character to the end of the config file, causing the Node.js code to read in an unwanted empty line, which naturally lead to problems.

Making sure the config file has just the required configuration files and nothing else should be enough to counter this problem, but in case of mysterious newline characters, commenting out the line 68 in lamport.js and uncommenting the line 73 in the same file may help solve the problem.

As mentioned in the lamport.js comments, if the program is only run with the provided config file, there should be no problems.