Distributed Systems
Project Erlang - Hierarchical System

Etienne Mauffret

March 7, 2022

Introduction

In this project, we aim at build a distributed application where the nodes are hierarchically distributed. We will use Erlang feature to this end. The system should be able to run some classic task of a distributed systems: counting, broadcasting, gather and scatter some data, reaching a consensus, etc.

A hierarchical system is a system where some nodes are responsible for other nodes. The nodes are separate in two (or more) categories : top and bottom level. The nodes at the top level knows some nodes of a bottom level that forms a group it's responsible from. A top level node also know about others top level nodes. A bottom level nodes only knows from node of its own group.

If you want to add a feature of your choice, feel free to ask me about it as it could take place of another feature.

Beyond the code itself, you are expected to handle a full project repertory :

- the .erl file;
- a README file that explain how to run your code and available feature. This document should be short and easy to read; This document should also explain everything you change according to the questions;
- a script that launch your project with desirable argument;
- *(optional)* any needed additional files for your code;

Finally, you are expected to provide a small report which should details your choices of implementation and discussion about your solutions and features (strength, weaknesses and sketch of proofs for desired properties). This document can be redacted either in English or French and should not be longer than a few pages (once again, keep it simple and readable!). You can use the application of your choice to write this report (LaTeX, Word, ...) as long as the final file is a PDF.

Every files should be place in a archive named ***name*.tar.gz**. If you want to use another archive service, please contact me before the submission.

Part 1

The grand scheme of Everything

Let's start by creating the system. We will use erland node as separation for the group. Each erlang node will have one top level node that should be able to communicate with bottom level node of the same erlang node and with other top level node.

Question 1

a) Create an erlang program that allow you to run a given amount of nodes. Each node should be able to start a random number (from a given range) of processes. A process that spawn should print it's ID, it's group ID and go into an infinite loop. A top level node should print a message before the creation of its bottom level nodes and once they are all set up.

b) test your program with a small system of 3 top level nodes and 3 to 5 bottom level nodes.

Part 2

Basic hierarchical operations

Now that your system is functional, let's start with the basics. We want the top level to gather some information about the global system and performs some simple tasks.

Question 2

a) Implement a protocol so that a top level nodes can know the number of nodes in the system. A top level nodes should not know how many bottom level nodes qre represented each top level but only the sum of nodes.

b) Implement a protocol broadcast a message through the whole system. The source of the message could be any node (top or bottom level).

c) Implement a protocol to compute a sum through the whole system. Each top level node should eventually get the result.

d) Implement a protocol to merge lists produced by bottom level nodes. Every top level nodes should eventually have the whole list.

e) Implement a protocol so that any node can create or kill any bottom level node. Make sure that every top level nodes keeps tracks of the number of participants of the system. Those protocols can be dangerous, so lets just assume no one will abuse it. :-)

Part 3

Representing but not leading

Once the basics functions are done, we need to get implement some more advanced features such that the consensus or the mutual exclusion so dear to your tdman.

Question 3

a) Implement a protocol to solve the consensus problem within a group and make sure that the top level node of the group knows about the decision.

b) Implement a protocol to solve the consensus between the top level nodes. The value proposed by a top level node must be a value that it's group agree upon.

c) Implement another protocol to solve the consensus between the top level nodes. However this time, the decision must depend on the number of bottom level node represented by each top level node (i.e a node that represent 5 bottom level node should have more weight that a top level node that represent only 1 bottom level node).

d) Does this way to solve the consensus seems more efficient than performing a classic consensus? What are the pros and the cons?

Now that you are able to solve the consensus, you know which problem is coming.. Yup that's right, the mutual exclusion problem.

Question 4

a) Is it relevant to use the consensus protocol to determine which process should go in the critical section? Explain your reasonning.

b) Implement a protocol to solve the Mutual exclusion protocol within a group. You may use a token based or permission based algorithm. Each node has a random number of time it wants to enter the critical section before to just give the permission or handle the token directly.

c) Implement a protocol to solve the mutual exclusion through the whole system. You may use a token based or permission based algorithm.

d) Now make sure that your protocol treat all nodes equally. That is that it do not wait for a whole group to finish its won mutual exclusion before to give the permission to another group.

Part 4

(Bonus) When everything goes wrong

Until now, everything was fine in a world without failure. But you know how the real world goes.. We consider here to king of failure: message failure and node failure.

A message fails if it does not reach its recipient. In this project, node failure are crash.

Question 5

a) Implement the failure protocol in the system so that each message has a given probability of failure. At any time, a node could crash as well.

b) Implement protocols to handle those failure. What hypothesis are needed?