Simulation of Distributed Algorithms

Erwan Jahier Karine Altisen Stéphane Devismes

Verimag Lab
Contacts: {`Erwan.Jahier, Karine.Altisen, Stephane.Devismes`}`@univ-grenoble-alpes.fr`

The design of distributed algorithms usually involves high-level models to describe the system and its semantics. The model abstracts away most of the very precise behavior of the system to rather focus on the main nature of distributed computation, that is: processes have local programs and memory and can only share information using message exchanges.

Distributed algorithms are thus developed and analyzed in such a (theoretical) model: usually their specification as well as their complexity are proven. Now, the distributed nature of the computation involves many situations and particular cases; furthermore, the design of distributed algorithms becomes harsher when in one hand the requirements are reinforced and in the other hand the environment becomes more intricate (how to obtain a fault-tolerant solution in a very highly dynamic environment?). This is why tools helping the design are mandatory.

In particular, we propose here a simulator, called SASA, that can be used as a debugging tool to interactively test the solution under development. Furthermore, it allows to compute benchmarks, *i.e.*, simulations can be launched as many times as necessary on various inputs (for example, a family of randomly generated networks) in order to obtain performance evaluations and in particular average time complexity.

The SASA simulator focuses on a family of fault-tolerant solutions called self-stabilizing algorithms. As a matter of fact, SASA sticks to the high-level model for which those algorithms are developed. SASA aims at accepting various input languages for defining the distributed algorithms. Currently, they can only be written in OCAML. The tool is also written in OCAML, and is still in development; enhancements of the SASA simulator encompass:

- the support for other input algorithm languages;
- the development of a compiler targeting the Lustre language, to perform efficient simulations, and to take advantage of Lustre verification tools (model-checkers);
- the analysis of existing distributed algorithms;
- the design of new distributed algorithms;
- the design of search heuristics to find worst-case scenarios (based on potential functions);
- a connection with formal proofs.

The goal of the internship is to cooperate with our team in at least one of those directions.
https://verimag.gricad-pages.univ-grenoble-alpes.fr/synchrone/sasa

Working context. The internship is part of ANR project ESTATE[1]. The student will be integrated in the lab Verimag[2].

Possible extension into a PhD thesis.

[1]https://wp-systeme.lip6.fr/estate/
[2]http://www-verimag.imag.fr/