Formal Fault-Tolerance Proofs for Distributed Algorithms

Mandy Zammit Adrian Francalanza
Department of Computer Science
Faculty of ICT
University of Malta
mzam0041@um.edu.mt adrian.francalanza@um.edu.mt

Distributed Algorithms express problems as concurrent failing processes which co-operate and interact towards a common goal [1, 2]. Such algorithms arise in a wide range of applications, including distributed information processing, banking systems and airline reservation systems amongst others. It is desirable that distributed algorithms are well behaved both in a failure free environment and even in the presence of failure (i.e. fault tolerant). To ensure well behavedness for all executions of distributed algorithms formal correctness proofs are needed. This is due to the concurrent nature of such algorithms, where executions of the algorithms result in different interleavings amongst parallel processes (i.e. there is a large number of possible execution paths).

However distributed algorithms are often described in semi-formal pseudo code and their correctness criteria[1] in natural language. The current status quo has a number of shortcomings. First is, the discrepancy between the semi-formal description of the algorithm and the actual implementation which is expressed in a formal language. This discrepancy is often a principal source of errors in the deployment of these algorithms. Second, due to the exponential interleaving amongst parallel processes and the subtle forms of process interference at these interleavings, becomes rather subtle and delicate.

Unfortunately formalisation tends to increase complexity because formal analysis forces you to consider every possible interleaving. At the same time the use of proof mechanizations still does not yield a clearly expressed proof that is easy to construct or understand; rather, it yields monolithic mechanical proofs that are very hard to digest.

In our work we address the complexity problem introduced by formalisation. We formalise and prove the correctness of three Broadcast Protocols, namely; *Best Effort Broadcast*, *Regular Reliable Broadcast*, and *Uniform Reliable Broadcast*. We use an asynchronous PI-calculus together with fail-stop processes and perfect failure detectors to encode the protocols and their correctness criteria, and bisimulation ($\approx$, a notion of coinductive equivalence) for the correctness proofs. In order to alleviate complexity, we extend the work conducted by Francalanza et. al. [3] where we make use of the concept of Harnesses, and a novel approach to correctness proofs whereby proofs for basic correctness (correctness in a failure free setting) and for fault tolerance (correctness in a setting were failures can be induced) are disjoint. We extend their methodology to produce inductive proofs containing coinductive techniques at each inductive step rather than having one large coinductive proof. This would allow us to move away from the large monolithic proofs.

References

[1] Lynch, Nancy A. , Distributed Algorithms, Morgan Kaufmann (1st edition), 2007.

[1] correctness criteria specify the well behavedness of an algorithm

[2] Tel, Gerard, Introduction to Distributed Algorithms, Cambridge University Press (2nd edition), 2001.

[3] Francalanza, Adrian and Hennessy, Matthew, A fault tolerance bisimulation proof for consensus, Proceedings of the 16th European conference on Programming, Springer-Verlag, 2007.

[4] Kühnrich, Morten and Nestmann, Uwe, On Process-Algebraic Proof Methods for Fault Tolerant Distributed Systems, Proceedings of the Joint 11th IFIP WG 6.1 International Conference FMOODS '09 and 29th IFIP WG 6.1 International Conference FORTE '09 on Formal Techniques for Distributed Systems, Springer-Verlag, 2009.