Static analysis of fault-tolerant distributed protocols

1 General information

- Advisor: Cezara Drăgoi
- Institution: Inria Paris, Antique Team
- Location: ENS, 45 rue d'Ulm, 75005 Paris
- Language: French or English
- Keywords: fault-tolerant distributed protocols, agreement, protocol modelization, automated verification, abstract interpretation

Summary

Fault-tolerant distributed data structures are at the core distributed systems, e.g., Amazon Dynamo, Apache Zookeeper, blockchain. Due to the multiple sources of non-determinism, their development is challenging and error prone. We aim to develop automated verification techniques that will increase the confidence we have in these systems. This requires identifying a protocol modelization framework that formalizes and simplifies the structure and specification of distributed systems, enabling the automation of their verification. Regarding the verification technique, we focus on static analysis based on SMT solvers and abstract interpretation. The targeted class of protocols includes solutions for consensus, e.g., Paxos [9], solutions for weaker qualitative forms of agreement, e.g., Lattice Agreement [5], k-set agreement, or qualitative solutions for agreement, e.g., blockchain algorithms. The internship is financed by the ANR project SAFTA - Static analysis of fault-tolerant algorithms.

2 Context

Highly available data storage systems and high processing power systems are available today due the massive development of distributed systems. A fault-tolerant distributed system is a set of independent processes that communicate via message passing, giving the illusion of a unique entity, despite possible communication problems or process failures. The standard way to make any applications fault-tolerant (available independently of network and hardware faults) is replication: the application is copied on different sites and all its clients are free to interact with any of the replicas deploying the application. The challenge posed by replication is to keep the replicas consistent, i.e., the application is in the same state at all replicas, despite any interaction with the client. Notorious examples are Amazon Dynamo (highly-available key-value

storage system), Apache Zookeeper (distributed hierarchical key-value store used to provide a configuration service and synchronization service), or the block-chain, the data structure bitcoin is built on top of.

The complexity of the design of fault-tolerant distributed algorithms is rooted in the FLP theorem [6] which states that in asynchronous networks (there are no bounds on the message delay), in the presence of faults (messages are lost or processes crash) it impossible for all processes to agree on a value, i.e., to solve consensus. To cope with impossibility results, existing algorithms make various assumptions on the network (e.g., bounding the number of messages that are dropped, or the number of processes that may crash) or on the provided consistency guarantees (e.g., processes might be allowed to agree on two different values instead of one). This leads to solutions with extremely complex flow of data, that simultaneously deal with the asynchronous nature of the network, the presence of byzantine (message corruption) or benign (message lost) faults, and updates of the processes local state.

Given the massive usage and the intricacy of distributed system, they are a prime candidate for automate verification. From a theoretical perspective, these are infinite state systems, communicating via unbounded channels, whose verification is in general undecidable. However, the state of the art shows that static analysis techniques like deductive verification and abstract interpretation, had a great impact on increasing correctness of sequential software, despite the fact that the verification of sequential programs is in general undecidable. These are incomplete verification tools, that either prove that the program satisfies the specification or they launch alarms, that is a collection of potential bugs in the program. We can cite the impact the Satisfiability Modulo Theory (SMT) solver Z3 [10] had for the verification of device drivers, or the static Analyzer Astrèe which was successfully applied to numeric embedded software. However, there are no automated verification tools to accompany the design of fault-tolerant protocols.

3 Objective

The goal of this project is to increase the confidence we have in replicated systems. We think that the difficulty does not only come from the algorithms but from the way we think about distributed systems. Therefore we propose to:

- **identify protocol modelization frameworks** that focus on algorithmic aspects simplifying the reasoning about fault-tolerance,
- define **automated verification techniques**, based on deductive verification and abstract interpretation for protocols
- define **refinement relations** between actual implementations and protocols that transport the guarantees achieved at the protocol level to the implementation.

The verification of fault-tolerant algorithms is in general undecidable but the usage of good abstractions could circumvent this undecidability result. There are two main obstacles to the verification problem: the complex control structure (interleavings, process crash, message delays, etc.) and the data which comes from potentially unbounded domains (unbounded arrays of integers, stacks, etc.). To overcome these obstacles we need to connect three research areas: *distributed algorithms*, which designs algorithmic solutions to fundamental problems, *systems*, which implements and optimizes these algorithms, and *formal verification*, which uses mathematical models to rigorously check that the algorithms and their implementations respect the intended specifications.

A good protocol language simplifies the representation of program behaviors, leading to simpler the proof arguments. We will investigate protocols formalizations that reduce the number of interleavings and have a modular structure, e.g., by separating algorithmic principles from network assumptions. Clearly a good protocol modelization framework must capture a variety of different systems. We are interested in formalizing systems like Paxos [3], Zab [8], View-Stamped [11], PBTF [2], blockchain [1, 7] etc.

We will develop automated verification techniques to prove safety and liveness properties of fault-tolerant distributed protocols. We will investigate the use of First-Order logic to formally define the specification and the distributed system's behaviors. We target fragments of the first-order logic that strike a balance between expressiveness and algorithmic properties. To automate deductive verification, we will define (semi-)decision procedures for satisfiability (implementable in SMT-solvers like the ones in [4]). Moreover, we will define algorithms that compute (over-approximations) of the fixed-point of the function associated with the programs transition relation, using the abstract interpretation framework. The project will increase the expressiveness power of the theories SMT-solvers and static analyzers can currently handle, when reasoning about set cardinalities and set comprehension, two key technical ingredients of any agreement proof.

Finally, to bridge the gap between verified protocols and implementations we will investigate refinement relations between the code of actual distributed implementations and verified protocols.

References

[1] Tendermint. `https://tendermint.com`.

[2] Miguel Castro and Barbara Liskov. A correctness proof for a practical byzantine-fault-tolerant replication algorithm. In *Tech. Rep. MIT/LCS/TM-590, MIT Laboratory for Computer Science*, 1999.

[3] Tushar D. Chandra, Robert Griesemer, and Joshua Redstone. Paxos made live: An engineering perspective. In *Proceedings of the Twenty-sixth Annual ACM Symposium on Principles of Distributed Computing*, PODC '07, pages 398–407, New York, NY, USA, 2007. ACM.

[4] Cezara Dragoi, Thomas A. Henzinger, Helmut Veith, Josef Widder, and Damien Zufferey. A logic-based framework for verifying consensus algorithms. In Kenneth L. McMillan and Xavier Rival, editors, *VMCAI*, pages 161–181. Springer, 2014.

[5] Jose Faleiro, Sriram Rajamani, Kaushik Rajan, G. Ramalingam, and Kapil Vaswani. Generalized lattice agreement. July 2012.

[6] Michael J. Fischer, Nancy A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, April 1985.

[7] Aggelos Kiayias Juan A. Garay and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. 2017.

[8] Flavio P. Junqueira, Benjamin C. Reed, and Marco Serafini. Zab: High-performance broadcast for primary-backup systems. In *Proceedings of the 2011 IEEE/IFIP 41st International Conference on Dependable Systems&Networks*, DSN '11, pages 245–256, Washington, DC, USA, 2011. IEEE Computer Society.

[9] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[10] Leonardo Moura and Nikolaj Bjorner. Z3: An efficient SMT solver, 2008.

[11] Brian M. Oki and Barbara Liskov. Viewstamped replication: A general primary copy. In *Proceedings of the Seventh Annual ACM Symposium on Principles of Distributed Computing, Toronto, Ontario, Canada, August 15-17, 1988*, pages 8–17, 1988.