Formal Verification Of Pastry Using TLA$^+$

Tianxiang Lu[1,2], Stephan Merz[1], Christoph Weidenbach[2]

[1] Inria & LORIA, Villers-lès-Nancy, France

[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany

1 Motivation

Pastry [1, 2] is an algorithm that provides a scalable distributed hash table over an underlying P2P network. Several implementations of Pastry are available, but to the best of our knowledge the correctness of the algorithm has not been verified formally. Since Pastry combines rather complex data structures, asynchronous communication, concurrency, resilience to churn and fault tolerance, we believe that it makes an interesting target for verification using TLA$^+$. More precisely, our goal is to model the join and lookup protocols of Pastry using the TLA$^+$ language, and to use the associated tools to verify significant correctness properties.

2 Contributions

We have modeled a significant part of the Pastry algorithm, including message routing and joining and departure of nodes. The first challenge in modeling the algorithm was to determine an appropriate level of abstraction. For example, we abstracted from real-time aspects and replaced timeouts by non-determinism, in order to make model-checking more feasible and to simplify the overall model. The second challenge was to fill in details for the formal model that are lacking in the pseudo-code description of Pastry [1], and we referred to open-source implementations. Finally, we had to determine what correctness properties the algorithm should satisfy, which was not explicitly claimed in original paper. No correctness properties are asserted in [1]; our main objective is to verify that there can never be two different Pastry nodes that consider themselves responsible for any single key.

The TLC model checker was an invaluable help for validating our TLA$^+$ models for small finite instances, for understanding the detailed behavior of Pastry, and for discovering properties and, more frequently, non-properties. Initial results on model checking our Pastry specification appear in [3]. In particular, we found a problem in the protocol for joining new nodes as it is described in the original Pastry paper [1]: if two nodes join concurrently between two neighbor nodes then the leaf sets of the new nodes may be incomplete, eventually leading to a situation where both nodes claim responsibility for the same key. The designers of Pastry pointed us to a technical report [2] that provides a solution for this problem, and we updated our model accordingly.

After TLC could find no more errors, we embarked on a formal proof of the property, for arbitrary instances. In a first step [4], we postulated hypothetical invariants of the underlying data structures, and used TLAPS to prove that these imply our global correctness property. A detailed TLA$^+$ model together with proof[1] could be found in [5]. Again, TLC helped us fine-tune the formulation of these invariants, and verify them over small instances. We have meanwhile refined these invariant properties to lower-level invariants and proved that these are indeed inductive for a restricted model where no nodes are allowed to leave the network and where nodes are assumed not to join concurrently in the same region of the Pastry ring. Our current models consist of about 1kloc for the specification and 15kloc for the proof. We plan to progressively relax the constraints so that nodes can freely leave and join the network.

3 Related Work

Zave [6] has used Alloy to model Chord at a much higher level of abstraction where operations such as *join* or *stabilize* are considered atomic and non-interfering. Focusing on eventual consistency, she has found a flaw in the original description of the algorithm and suggests a repair that may be correct—however, Alloy is not supported by a theorem prover like TLAPS to formally prove the invariants.

4 Our Experience with the TLA$^+$ Tools

The trace exploration features provided by TLC through the TLA$^+$ toolbox were invaluable for understanding counter-examples produced for corner cases and for improving our models. It might be possible to improve the display of the (part of an) action that is causing a state transition, perhaps by "zooming into" the action predicate that is evaluated.

Once TLC did no longer quickly produce error traces, we turned to the command-line version and let the model checker run on a 8-processor server. Our models generate more than 30 billion states even for instances with just four nodes; hash collisions are therefore highly probable. The use of several worker threads during state exploration led to a significant speed-up and worked without a glitch. However, after letting TLC run for weeks, the process sometimes ended up with a huge memory footprint but without any more visible progress or even CPU usage. We suspect this to be a problem of the Java run-time and are eager to use the new distributed version of TLC.

[1] The code could be downloaded at `http://www.mpi-inf.mpg.de/~tianlu/software/PastryTheoremProving.zip`.

TLAPS improved significantly since we started this work around the end of 2009. Use of the new SMT-based backend may help us shorten our proofs, which contain significant arithmetic reasoning. The toolbox was very helpful for zooming into parts of the proof, for non-linear proof editing, and for jumping back and forth between definitions and proofs. Checking the status of a proof works well for small and medium-sized proofs, thanks to fingerprinting. However, the memory footprint of the Eclipse editor and the proof manager can become critical for large proofs. For example, our main invariant proof has about 12500 lines, and even allocating 4GB of main memory to Eclipse is barely enough to generate all proof obligations for status checking. We believe that it should be possible to reduce the memory overhead, and perhaps generate proof obligations incrementally rather than upfront.

References

[1] M. Castro, M. Costa, and A. I. T. Rowstron. Performance and dependability of structured peer-to-peer overlays. In *Intl. Conf. Dependable Systems and Networks (DSN 2004)*, pages 9–18, Florence, Italy, 2004. IEEE Computer Society.

[2] A. Haeberlen, J. Hoye, A. Mislove, and P. Druschel. Consistent key mapping in structured overlays. Technical Report TR05-456, Rice University, Department of Computer Science, August 2005.

[3] T. Lu, S. Merz, and C. Weidenbach. Model checking the pastry routing protocol. In *10th Intl. Workshop Automatic Verification of Critical Systems*, pages 19–21, Düseldorf, Germany, September 2010. Universität Düseldorf.

[4] T. Lu, S. Merz, and C. Weidenbach. Towards verification of the Pastry routing protocol using TLA$^+$. In *FMOODS/FORTE 2011*, volume 6722 of *Lecture Notes in Computer Science*, pages 244–258, Reykjavik, Iceland, 2011. Springer.

[5] T. Lu, S. Merz, and C. Weidenbach. Towards verification of the Pastry routing protocol using TLA$^+$. Research Report MPI-I-2011-RG1-002, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, October 2011.

[6] P. Zave. Using lightweight modeling to understand chord. *Computer Communication Review*, 42(2):49–57, 2012.