Project Proposal
for
Consensus Protocol Simulator

Haobin Ni

October, 2017

1 Introduction

Distributed consensus has been an active field of study for nearly half a century. From the classic work of Byzantine agreements [1] [2], FLP [3] and DLS [4] lower bounds, to practical protocols such as PBFT [5], Paxos [6] and Raft [7], including the recent success of Bitcoin [8] and Ethereum [9], the corresponding analysis [10], many protocols have been proposed to solve the problem of establishing consensus among honest players despite the influence of corrupted players in a distributed system.

Outside academia, consensus protocols are the key component of data centers/cloud services to provide reliable scalability. This includes well-known examples such as Google Chubby [11] and Apache Zookeeper [12].

However, a considerable gap has been observed between the two worlds: On one hand, cryptographers describes their protocols in a very abstract way using frameworks such as universal composition [13] because they focus on proving the security properties. On the other hand, system engineers want an exact version of the protocol since they need to make it run on the machines. As consensus protocols can be very subtle, this gap causes many implementations fail to achieve their theoretical correctness [14] [15] [16]. How can one make sure what the engineers have implemented is what the cryptographers have proven?

This project explores one possibility for bridging such a gap - a simulation tool to build/test intermediate representations for consensus protocols. While the representation must be precise to simulate the execution, it also needs to be concise - not adding to much the abstract description of the proposed protocol.

Building such an artifact will not only serve as a starting point towards the synthesis of consensus protocol implementations from its specification but also provides a prototype platform for creating and testing such protocols. Furthermore, it will be a handy pedagogical tool for people to learn and try out consensus protocols.

2 Goal

The goal of this project is to build a simulator which can **generate the execution path of a protocol with given parameters**. The simulator will have some built-in options for all of the parameters but users can also plugin their custom components without changing the underlying platform/framework. Considering the scope of this course, a list of possible protocol/parameters is followed for reference.

- **Environment**: The environment specify what the protocol needs to do and what the protocol can do.
  - Network model: Synchronized or Partially synchronized or Asynchronous
  - Communication model: Authenticated Channels or PKI
  - Consensus type: Byzantine agreement, Binary agreement, Linear-ordered log, etc
  - Other environment parameters: total number of nodes, network delay, random oracle etc
- **Protocol**: The protocol to be simulated. The protocol and the environment should match. e.g. It is not well-defined to run a synchronous protocol in asynchronous network.
  - Dolev-Strong Protocol
  - Herding Protocol
  - Nakamoto Blockchain
  - Sleepy Blockchain
- **Adversary**: The user can adopt an adversary program to play the corrupted players to test the guarantees of the protocol.
  - Adversary mode: static or adaptive
  - Adversary type: byzantine, crash, halt
- **Measurement**: The measurement is a function that takes the execution path as input and do various measures, such as consistency, validity and chain quality etc.

Though the above list looks very unwieldy, in the final artifact, there will be some basic UI to make it easier to use. It is also possible to support batch experiments where the user can specify a group of parameter to experiment on.

3 Methodology

This project will be completed in a few fast iterations. A new protocol will be supported in each iteration thus new components must be implemented and necessary changes to the framework will also be made. We will get a running prototype after each iteration and the new insights gained can be tested fairly quickly. Compared to the traditional cascading program paradigm which decides the total design first then carry out the implementation, I think this way of agile development is more suitable for this project since it is more flexible to changes and new ideas.

4 Time Table

Expected Deadline: End of Nov.

- Iteration 1: Bootstrap & Dolev-Strong (- Oct 31)
- Iteration 2: More on framework & Herding Protocol (- Nov 7)
- Iteration 3: Nakamoto (- Nov 18)
- Iteration 4: Sleepy (- Nov 25)
- Wrap-up and final report: (- Nov 29)

5 Future Works

This project can be extended further in multiple ways. Here're three of possiblie directions to explore.

1. **Support more protocols**: Extend the framework and implement more protocols. This will make the simulator more powerful in describing protocols.

2. **A domain specific language for protocols**: This simulator is in fact somewhere between the ideal world and the real world since it can simulate "an honest third party" or ideal cryptography. So from the interfaces of this simulator we can probably generalize a language to describe the protocols. This description can be very useful since it has a decent amount of abstract. I can see applications in both synthesizing the protocol from specifications and building formal proofs for the protocols.

3. **Composition of the protocols**: In the current design of the protocol, composability of the protocols are limited. Idealized cryptography components are treated as blackbox functions but they are built-in and one cannot call other protocols to compose larger protocols. This is a meaningful direction to explore since we want similar composability as we have seen in theory. I believe it is possible to add another level of abstract and somehow "compile" the protocol before it actually runs.

References

[1] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (July 1982), pp. 382–401.

[2] Danny Dolev and H. Raymond Strong. "Authenticated Algorithms for Byzantine Agreement". In: *SIAM Journal on Computing* 12.4 (1983), pp. 656–666.

[3] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. "Impossibility of Distributed Consensus with One Faulty Process". In: *J. ACM* 32.2 (Apr. 1985), pp. 374–382. ISSN: 0004-5411.

[4] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. "Consensus in the Presence of Partial Synchrony". In: *J. ACM* 35.2 (Apr. 1988), pp. 288–323. ISSN: 0004-5411.

[5] Miguel Castro and Barbara Liskov. "Practical Byzantine Fault Tolerance". In: *Proceedings of the Third Symposium on Operating Systems Design and Implementation*. OSDI '99. 1999, pp. 173–186.

[6] Leslie Lamport. "Fast Paxos". In: *Distributed Computing* 19 (2006), pp. 79–103.

[7] Diego Ongaro and John Ousterhout. "In Search of an Understandable Consensus Algorithm". In: *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. 2014, pp. 305–319.

[8] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System". In: (May 2009). URL: `http://www.bitcoin.org/bitcoin.pdf`.

[9] Wood Gavin. "Ethereum: A Secure Decentralised Generalized Transaction Ledger". In: (Apr. 2014). URL: `http://gavwood.com/paper.pdf`.

[10] Rafael Pass, Lior Seeman, and abhi shelat. *Analysis of the Blockchain Protocol in Asynchronous Networks*. Cryptology ePrint Archive, Report 2016/454. `http://eprint.iacr.org/2016/454`. 2016.

[11] Mike Burrows. "The Chubby Lock Service for Loosely-coupled Distributed Systems". In: *Proceedings of the 7th Symposium on Operating Systems Design and Implementation*. OSDI '06. 2006, pp. 335–350.

[12] Patrick Hunt et al. "ZooKeeper: Wait-free Coordination for Internet-scale Systems". In: *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*. USENIXATC'10. Boston, MA, 2010, pp. 11–11.

[13] R. Canetti. "Universally composable security: a new paradigm for cryptographic protocols". In: *Proceedings 2001 IEEE International Conference on Cluster Computing*. 2001, pp. 136–145.

[14] Ding Yuan et al. "Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-intensive Systems". In: *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*. OSDI'14. 2014, pp. 249–265.

[15] Colin Scott et al. "Minimizing Faulty Executions of Distributed Systems". In: *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. 2016, pp. 291–309.

[16] Pedro Fonseca et al. "An Empirical Study on the Correctness of Formally Verified Distributed Systems". In: *Proceedings of the Twelfth European Conference on Computer Systems*. EuroSys '17. 2017, pp. 328–343.