Offscale's consensus algorithm

Vladimir Komendantskiy

25th September 2019

Abstract

Using prior work on Lachesis consensus [3] and following informal implementation notes by Offscale [8], I am aiming at defining a gossip-inspired consensus algorithm in a way amenable to incremental implementation and formal verification.

1 Introduction

Replicated state machines underpin blockshain technology. Blockchains whose state contains a unique linearly ordered chain of blocks follow replicated state machine design to achieve replication of the same chain across the network.

It should be noted that Offscale's design is one of a *blockchain* since, as Maxim Zakharov put it in [8] exactly, "tasks/problems the consensus layer solves: ... creating a *linear order of all events* of a frame based on lamport timestamps of the events and synchronisation patterns". To put the quote in context, it should be added that frames are linearly ordered too, hence all events that are members of complete frames are linearly ordered. Therefore an Offscale blockchain pursues the same fundamental goal as most existing blockchains such as Bitcoin or Ethereum. However the design of the replicated state machine used by such an Offscale blockchain is different due to choices made to address inherent problems in today's decentralised mainstream blockchains: low transaction rate, low transaction confirmation speed and strong synchrony assumptions. By increasing the rate at wich transactions can be submitted to the network, and submitted transactions confirmed, an Offscale blockchain aims to solve a part of the scalability problem which limits the use of blockchain technology to the "store of value" use cases, and to apply thus improved blockchain technology in domains that strongly rely on high thransaction throughput such as large-scale heterogenous computational networks commonly referred to as the Internet of Things. By lifting synchrony assumptions an Offscale blockchain addresses security concerns of synchronous protocols, Bitcoin included, which require setting an interval between blocks to at least the time greater than the maximum message delay on the network [5]. The minimum interval requirement poses a straightforward problem for such protocols: what if the expected maximum message delay does get exceeded for a majority of nodes making them unable to produce or receive blocks, thereby making it possibly for a minority of nodes construct the longest (and therefore valid) chain in the meantime? This is a possible attack vector on a synchronous blockchain which can be exploited by a maliciously conspiring adversary. In an Offscale blockchain and in an asynchronous blockchain in general, this kind of a minority adversary will not be able to mutate the global state because, even if the majority experiences a network split, a majority vote is still required in order to mutate the global state, which is why the minority will not be able to make progress without completely leaving the original network.

2 Related work

Gossip-based consensus algorithms gained wide recognition thanks to cryptocurrency projects such as IOTA or Hashgraph [1]. MaidSafe's Parsec [2] consensus also uses a gossip protocol as a base layer on top of which a binary agreement protocol is used to construct a linearly ordered list of blocks. Parsec makes weak synchrony assumptions to guarantee eventual message delivery.

Out of all the protocols listed above, Hashgraph has received the best coverage with regards to formal verification [4] and scrutiny of the implementation [7]. This has led to the underlying directed acyclic graph (DAG) data structure spreading over to other algorithms [2, 3]. However it should be noted that the choice of the same DAG data structure is not a requirement for a gossip consensus protocol. More specifically, even if a gossip protocol uses a DAG to store events and child-parent relations between those, the number of parents for non-genesis events does not have to be exactly two as in Hashgraph, and there can be various reasons to prefer a greater number of parents.

3 Model assumptions

Any protocol performs up to its theoretic assumptions. Protocols with stronger assumptions – such as assumptions of linear message ordering and eventual delivery – may be easier to define and prove statements about. When the assumptions don't hold, the statements that rely on those assumptions don't hold either and the protocol may exhibit arbitrary behaviour. Therefore having strong assumptions may help in proving statements but those statements will be weak and would cover a small number of cases.

Offscale's consensus model assumptions are as weak as it is possible for stating protocol properties. In the model, I assume total asynchrony between participants: messages from a single sender can be reordered and an arbitrary number of messages can be dropped. Therefore a failed message sender is indistinguishable from a still operating message sender whose communication links are down. The only assumption is that a message cannot be received before being sent.

A required protocol property is Byzantine fault tolerance. For a consensus network of size N, I allow the maximum theoretically possible number of Byzantine participants t where $3t < N$.

4 Basic notions

Honest participant is a consensus network participant adhering to the protocol.
Malicious participant is a consensus network participant exhibiting arbitrary, Byzantine behaviour.

References

[1] Leemon Baird. *The Swirlds Hashgraph Consensus Algorithm: Fair, Fast, Byzantine Fault Tolerance.* Swirlds Tech Report SWIRLDS-TR-2016-01. 2016.

[2] Pierre Chevalier, Bartolomiej Kamiński, Fraser Hutchison, Qi Ma, Spandan Sharma. *Protocol for Asynchronous, Reliable, Secure and Efficient Consensus (PARSEC).* 2018. `https://docs.maidsafe.net/Whitepapers/pdf/PARSEC.pdf`

[3] Sang-Min Choi, Jiho Park, Quan Nguyen, Andre Cronje. *Fantom: A scalable framework for asynchronous distributed systems.* 8th February, 2019. `https://github.com/SamuelMarks/consensus-rough-notes/blob/`

cd603414cf16526ea8e94ed341d9021eb8e0041f/papers/Fantom__A_scalable_framework_for_asynchronous_distributed_systems.pdf

[4] Karl Crary. *Hashgraph consensus aBFT proof in Coq*. 23rd October, 2018. https://swirlds.com/downloads/hashgraph-coq.zip

[5] Rafael Pass, Elaine Shi. *Rethinking Large-Scale Consensus*. 30th Computer Security Foundations Symposium (CSF), 2017. https://ieeexplore.ieee.org/document/8049715

[6] Team Rocket. *Snowflake to Avalanche: A Novel Metastable Consensus Protocol Family for Cryptocurrencies.* 16th May 2018. https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV

[7] Eric Wall. *Hedera Hashgraph – Time for some FUD*. 2nd September 2019. https://medium.com/@ercwl/hedera-hashgraph-time-for-some-fud-9e6653c11525

[8] Maxim Zakharov. *Consensus schema.* 20th September, 2019. https://github.com/SamuelMarks/consensus-rough-notes/blob/356d9d2fdf8872c23e57f65154522979f522b068/Consensus-schema.md