BlockfinBFT Compared to other Consensus Algorithms

Introduction

This document compares //STORE's BlockfinBFT consensus algorithm to some of the popular Byzantine Fault Tolerant (BFT) consensus algorithms to answer the following question.

Why did //STORE choose to design its own consensus algorithm instead of using one of the well established algorithms?

Creating a new algorithm has its downsides like poorly studied security characteristics, lack of general expertise in the community, provability, etc., so the decision to create a new algorithm should have strong backing. This document discusses our research, our findings, and why we decided a new algorithm is necessary to achieve //STORE's goals.

This document compares BlockfinBFT against pBFT, HoneyBadger BFT, and Tendermint. These 3 protocols cover a wide variation of network setup, safety and liveness guarantees, and performance characteristics.

//STORE goals

Most blockchains offer limited decentralization either in their design or in their governance. If the network has a tendency towards centralization, there is no need to use blockchains and decentralized ledgers in the first place because they will always be less efficient than a centralized system. So, decentralization was a core requirement when we conceived //STORE. All leader or committee-based consensus algorithms have a long-term tendency towards centralization, so such algorithms are undesirable. The following goals were set for //STORE.

1. High decentralization — where the BFT tolerance is required on the *entire network* instead of a small committee elected to create the next block. This requires participation from all the nodes in the network.
2. High throughput — highly decentralized networks tend to have poor throughput. We need a design that offers high throughput, while at the same time maintaining high decentralization.
3. A consensus algorithm that works in a real world scenario where network connectivity and latency are unpredictable.

A consensus algorithm that satisfies the above requirements was needed. The rest of this document captures our analysis of various consensus algorithms against the above requirements.

pBFT

pBFT runs a three-phase agreement protocol among replicas before it executes a request. At each level (think block height), a *proposer* tries to get a unique value 'locked' in two phases — *prepare* and *commit*. First, a proposer tries to obtain 2f+1 prepares. A set of 2f+1 signed prepares is called a *commit-certificate*. In the second phase, replicas commit to the commit-certificate. If 2f+1 replicas commit to a certificate, it becomes a *committed decision*.

Advantages

- pBFT is a well studied classical consensus protocol in the literature.
- Ability to provide *transaction finality* without the need for confirmations like in Proof-of-Work models such as the one Bitcoin employs.

Drawbacks

- The model only works well in its classical form with small consensus group sizes due to the cumbersome amount of communication that is required between the nodes.
- The model is also susceptible to *sybil attacks* where a single party can create or manipulate a large number of identities (nodes in the network), thus compromising the network. This is mitigated against with larger network sizes, but scalability and the high-throughput ability of the pBFT model is reduced with larger sizes and thus needs to be optimized or used in combination with another consensus mechanism.
- The model critically relies on a weakly synchronous network for liveness. https://eprint.iacr.org/2016/199.pdf describes a setup with an adversarial scheduler that violates this assumption, and indeed prevents PBFT from making any progress at all.
- Requires out-of-protocol measures such as slashing, to combat malicious leaders.

So, while pBFT is the most studied classical BFT protocol, its mathematical proof alone is not sufficient to guarantee liveness in practical setups. Since a *proven* property (liveness) can so easily be defeated requiring following protocol assumptions in a strict manner (which is not practical in real world scenarios) we concluded that pBFT will not serve our goals discussed previously.

Projects using pBFT

The following projects use pBFT, but employ secondary measures to overcome its limitations.

Zilliqa — Zilliqa employs a highly optimized version of classical pBFT in combination with a PoW consensus round every ~100 blocks. They use multi-signatures to reduce the communication overhead of classical pBFT. This is also a direct result of their implementation of pBFT within their sharding architecture so that pBFT consensus groups remain smaller within specific shards, thus retaining the high-throughput nature of the mechanism while limiting consensus group size.

Sharding is undesirable because of the //STORE's primary use case as a global payments infrastructure. We cannot guarantee that merchants, developers, and their users will all be in the same shard. Cross-shard communications are very expensive and they affect the finality of transactions that depend on such communications. So, Zilliqa's approach with sharding is unsuitable for //STORE.

Hyperledger — Hyperledger Fabric is an open-source collaborative environment for blockchain projects and technologies that is hosted by the Linux Foundation and uses a permissioned version of the pBFT algorithm for its platform. Since permissioned chains use small consensus groups and do not need to achieve the decentralization of open and public blockchains such as Ethereum, pBFT is an effective consensus protocol for providing high-throughput transactions without needing to worry about optimizing the platform to scale to large consensus groups.

//STORE is not a permissioned network. While it requires staking for its nodes to be eligible to participate in the consensus and uses a fixed number of nodes in a given phase, it is an open network and doesn't assume *trust* among the nodes.

Variations of pBFT

There are tens of offshoots of pBFT such as Zyzzyva, Q/U, etc. which address some of the shortcomings of pBFT with either relaxing certain definitions (of finality, for example) or assuming best case scenario in the primary consensus rounds and falling back to classical pBFT in a worst case scenario. They also optimize one or more rounds in the consensus process based on specific use cases and assumptions. But these protocols are less studied and much less implemented, so their security properties are largely unknown too.

HoneyBadgerBFT

HBB is an asynchronous consensus protocol, which makes it very desirable to use in the //STORE setup.

Advantages

- Asynchronous protocol. Doesn't make any timing assumptions like pBFT.
- High throughput. A 104 node network achieved 1,500 TPS.

Drawbacks

- Requires a permissioned setup for its threshold encryption.
- It claims to be leaderless, but the algorithms presented do not discuss how block proposals from different leaders converge.
- Not as well studied as pBFT and there is only academic interest in this protocol.

Tendermint

Tendermint is functionally similar to pBFT from protocol perspective. It assumes partial synchrony like pBFT, but uses gossip instead of a broadcast primitive for communication with peers. Architecturally, it separates the consensus engine from the application layer using ABCI interface. So, the protocol can be used with any application. The same set of advantages and drawbacks discussed with pBFT apply here as well. In addition, Tendermint uses a round-robin approach to selecting the block proposer, which makes block proposers susceptible to DDoS attacks because the block proposer selection is predictable. More importantly, limiting consensus group size remains as the primary concern. For example Cosmos, which is based on Tendermint, has a limit of 100 validators in its mainnet, which will be expanded to 300 validators in next 10 years. This shows the practical limit on the consensus group size.

//STORE has spent quite a bit of time researching Tendermint extensively. We have published our performance test results in 8 and 21-node setup, with nodes distributed geographically on different AWS regions. See our GitHub for our research on Tendermint's performance. While we are able to achieve high throughput, our concerns with the leader-based protocols and the practical limit on the consensus group size made Tendermint unsuitable for our needs.

Why leaderlessness is important?

We can observe that all leader-based protocols waste a lot of resources. In Bitcoin and Ethereum for example, it is the electricity used by miners who eventually lost to the winning miner (although Ethereum uses *uncle blocks* as additional security to the chain, the *work* done by these miners is actually wasted). Similarly in PoS blockchains, the work (block proposal and consensus rounds) done by a proposer is wasted, if the consensus falls through because of lack of agreement. In

Tendermint this can happen if pre-commit votes didn't happen within the protocol defined timeout. In such a case, a new proposer is elected for the next round, throwing away all the work done by the current proposer. While this waste is not as significant as in PoW, it is a waste that *could* happen in every round. With a leaderless algorithm, this waste can be prevented by a pipelined approach to building blocks. While the block finality is probabilistic — more than ⅔ of nodes need to make the same decision — the protocol makes progress without throwing away the work done by the nodes.

With this background, we now present BlockfinBFT.

1. The following presentation sets out the groundwork for BlockfinBFT and describes the consensus steps.
   https://docs.google.com/presentation/d/e/2PACX-1vRJorF_Iu7qToFWyTORNy4Xha1FfFgSzd5HbpNx7iTRPKLShAfKQ4pRoHz1R6tm_VdtQWuJXxXcHfoN/embed?start=false&loop=true&delayms=60000&slide=id.p3
2. The following video walks through the simulation of the consensus steps. It proves the correctness of the algorithm in the presence of adversaries.
   https://www.youtube.com/watch?v=7qD6KrUJZ-4
3. The following presentation compares trustnessless of //STORE with Bitcoin.
   https://docs.google.com/presentation/d/1ak8JTLbuha6kngAW4pNeFVKSVAqDFozauApiSkHYc6g/edit#slide=id.g4959126e67_0_0
4. The following spec provides the database schema for //STORE blockchain.
   https://storecoin.com/media/BlockfinBFT-Database-Schema-//STORE-Header.pdf