Mosaic
Running Meta-Blockchains to Scale Decentralised Applications

B. Bollen, P. Gevorgyan, S. Khedar,
D. Kumar Nath, M. Schenck
for OpenST Foundation
working draft

OpenST Mosaic v0.10 - September 2018

Abstract

This is a working draft summary of notes and whiteboard sessions as we have been iterating towards the implementation for Mosaic v0.10. This draft is shared for review and comments by the community. This is not the version intended for publication, and you are invited to review critically all content. The code is work in progress at `github.com/openstfoundation/mosaic-contracts`.

1 Introduction

Mosaic is a parallelisation schema for decentralised applications. Mosaic introduces a set of Byzantine fault-tolerant (BFT) consensus rules to compose heterogeneous blockchain systems into one another. Decentralised application can use Mosaic to compute over a composed network of multiple blockchain systems in parallel.

A decentralised application is an application for which the computation is requested, performed, and paid for by independent actors. To ensure correct execution of decentralised computations first an order on the inputs of execution must be determinable. To date Ethereum is the leading networkof nodes that enables developers to write and deploy code which can be called by independent users to collectively construct a shared state of the application.

The Ethereum network achieves agreement on the collective state by constructing a blockchain. A blockchain derives correctness from full replication of the computation by nodes and for Ethereum today the chain is appended through Proof-of-Work consensus.Proof-of-Work (PoW) introduces a block reward incentive for nodes to keep producing blocks and thereby securing the chain of historical blocks. However, PoW produces a serial execution model and nodes cannot be collectively rewarded for the computation they all have to perform, which renders the system computationally inefficient[1].

Active research and implementation work is ongoing to scale Ethereum's transaction throughput by dividing the verification work into multiple sections, also referred to as *shards*. At the same time Ethereum is committed to moving from the probabilistic Proof-of-Work consensus engine to a BFT based Proof-of-Stake (PoS) consensus engine[2]. The outset of a BFT consensus engine is to collectively produce a blockchain which provably - under certain honesty assumptions - cannot finalise conflicting blocks.

Bitcoin's arrival in 2009 is widely accepted as a turning-point, proving by demonstration, that a network can be constructed that balances economic interests of independent actors to collectively secure digital assets against double-spending. Ethereum innovates on Bitcoin by introducing a generalised scripting language, enabling a broad developer community for decentralised applications to grow.

Thus far a vision for a decentralised web has been a major driver of innovation. However, to power global networks of billions of users and decentralised computation on vast decentralised data stores, it is reasonable to assume no single *backbone-blockchain* network will suffice. Much like the internet is a composed network of networks, we can conceive decentralised applications to transcend any single blockchain and execute across a network of blockchain networks.

In this work we detail two protocols. A first layer-2 protocol constructs *meta-blockchains* on top of existing blockchain networks to extend their state space and transaction throughput capacity. A second protocol, called a *gateway*, acts as a message bus to send typed messages atomically between the underlying blockchain and a meta-blockchain running on top of it.

Together these two building blocks can be used by decentralised applications to construct a parallelisation schema to increase computational capacity at lower gas cost. In the simplest form a decentralised application can offload transaction processing to a single meta-blockchain. More advanced applications can deploy logic across multiple meta-blockchains and process transactions in parallel while asynchronous messages can be sent between meta-blockchains.

[1]explain verifiers dilemma

[2]refer to Ethereum v2

2 Composing auxiliary systems into an origin

A first objective of Mosaic is to compose auxiliary blockchain systems into an origin blockchain, such that the capacity of the origin blockchain can be extended and the security properties of the origin blockchain can be inherited by the auxiliary blockchain.

To this end consider two blockchain systems, an origin blockchain O with state transition rules t_O which progress the state Σ_O

$$t_O : \Sigma_O \rightarrow \Sigma'_O$$

and similarly consider an auxiliary system A with state transition rules t_A such that $t_A : \Sigma_A \rightarrow \Sigma'_A$.

Transition rules for the origin and auxiliary system can be similar but do not have to be. In our discussion, when we need an example, we will reference to the origin blockchain as Ethereum (Byzantium), running Proof-of-Work, and for the auxiliary system we consider Go-Ethereum, running "clique" Proof-of-Authority (PoA). We deliberately use PoA for our considerations for the auxiliary system to emphasise that the security for the composed auxiliary system is not derived from the consensus engine of the auxiliary system. Rather the security properties of the composed system must rely only on the security properties of the consensus rules of the origin blockchain and on the Mosaic BFT consensus rules composing the auxiliary system into the origin system[3].

On the origin blockchain we define a meta-blockchain $\mathcal{B}$ with a staked validator set $\mathcal{V}_\mathcal{B}$. The blocks B_i of $\mathcal{B}$ are committed to a *core* contract on origin O. For a given history of O[4] the meta-blockchain cannot fork if we enforce that block B_{n+1} can only be (proposed and) committed after block B_n has been committed.

We define a meta-block $B_i(K, T, S) : \Sigma_A \rightarrow \Sigma'_A$ to progress the state of the auxiliary system A, where we call K the *kernel*, T the *transition object* and S the *seal*. A block B_i at height i is committed with a seal S when a $+\frac{2}{3}$ supermajority of the weighted votes has been verified[5]. We will detail

[3]The block proposers of the auxiliary system do have the ability to censor transactions from the auxiliary system if they have an ability to collude.

[4]The origin blockchain O might be probabilistically finalised, in which case it is easy to observe that a transaction to a contract can always assert that it is valid only on the intended branch of history by referencing a historical blockhash.

[5]While a vote counted on a BFT system itself is already BFT for a normal majority, we still require a supermajority, because the votes to commit a meta-block will be collected on the auxiliary system for which there is no assumption it is BFT, and we want Mosaic to be able to finalise transactions on the auxiliary system before they are committed to origin.

later how the meta-block is constructed, but first we will describe the vote messages that together can combine to seal a meta-block.

2.1 Checkpoint finalisation - Casper FFG

The reader is assumed at this point to be familiar with the work presented in *Casper the Friendly Finality Gadget*[6] (FFG), as we will build on the logic and proofs presented there. We will intend to align with definitions and notations where possible and highlight deviations where appropriate. In this section we briefly (and incompletely) summarise core concepts from the paper, so that the shared concepts are established for the reader.

Casper FFG presents an algorithm to build up an overlay network of vote messages presented to a smart contract about the blocks already produced by the underlying network of nodes running the blockchain. The overlay network aims to repeatedly economically finalise a single branch of the underlying network. It does so by allowing staked validators to cast votes which together can construct a *justified chain*.

A justified chain is formed by a sequence of *supermajority links*, where a link identifies a *source* blockhash s and height h_s, and a *target* blockhash t and height h_t. A vote message in Casper FFG is a signed link $\langle s, t, h_s, h_t \rangle_v$ by a validator v. A $+\frac{2}{3}$ supermajority of the same link signed is a supermajority link which justifies the target block if the source block is itself already justified[7].

In the work it is shown that checkpoints along the justified chain can additionally be considered *finalised*, if and only if they are justified themselves and their direct child is justified. It is then shown that should validators finalise a checkpoint on a contradicting branch of the history of the underlying blockchain, then minimally $\frac{1}{3}$ of the validator weight must have signed vote messages that violate either one of two rules, *the Casper slashing conditions.*

These slashing conditions can be intrinsically validated given two signed vote messages from a validator v. As such a validator must never sign two vote messages $\langle s_1, t_1, h_{s_1}, h_{t_1} \rangle_v$ and $\langle s_2, t_2, h_{s_2}, h_{t_2} \rangle_v$ for which either $h_{t_1} = h_{t_2}$ or $h_{s_1} < h_{s_2} < h_{t_2} < h_{t_1}$ holds.

It is additionally shown that validators can always finalise a new checkpoint, without being required to violate the slashing conditions given that the block proposers propose blocks to append to the finalised fork, ie. follow the fork selection rule of the overlay network.

[6] ref: Casper the Friendly Finality Gadget by V. Buterin and V. Griffith, November 2017

[7] The initial "genesis" block of the overlay network is considered justified by definition.

2.2 Finalising auxiliary systems

Given a validator set $\mathcal{V}_{\mathcal{B}}$ staked for the meta-blockchain on origin, validators v can submit vote messages on the auxiliary system about the auxiliary system of the form

$$\langle \tilde{T}, s, t, h_s, h_t \rangle_v \tag{1}$$

where $\tilde{T}$ is introduced as the `Keccak256` hash of the transition object T. We require that any transition object is calculable by a smart contract on the auxiliary chain for any finalised checkpoint along the justified chain. For a given link, we define the transition object to refer to the state of the blockchain at source block height h_s. It then follows that for a given source blockhash s $\tilde{T}$ is uniquely determined by s and the same properties of accountable safety and plausible liveliness hold for a finalised checkpoint on a justified chain constructed by such *externalised vote messages* (1), if we trivially extend the slashing conditions to accommodate for the transition object hash.

A validator $v \in \mathcal{V}_{\mathcal{B}}$ must never sign two externalised vote messages $\langle \tilde{T}_1, s_1, t_1, h_{s_1}, h_{t_1} \rangle_v$ and $\langle \tilde{T}_2, s_2, t_2, h_{s_2}, h_{t_2} \rangle_v$ such that any of the following three conditions holds:

1. $h_{t_1} = h_{t_2}$,
2. $h_{s_1} < h_{s_2} < h_{t_2} < h_{t_1}$,
3. $s_1 = s_2 \wedge T_1 \neq T_2$.

As the slashing conditions can be intrinsically asserted to have been violated given two externalised vote messages by the same validator, there is no communication overhead to assert a possible violation of these conditions on the origin blockchain; even if the justified chain is being constructed on the auxiliary system. As a result the validators in $\mathcal{V}_{\mathcal{B}}$ can be held accountable on the origin blockchain for their voting actions on the auxiliary system. The slashing condition can be asserted on both systems and there is a clear incentive for the honest validators to assert any violation without delay on both the origin and auxiliary system, naturally synchronising the validator weights when such an event occurs.

We will later in this work address a dynamic validator set $\mathcal{V}_{\mathcal{B}}$, where validators can join and log out on the origin system. However, already with a static validator set, we can observe that the finality gadget first introduced for (re)defining economic finality in layer-2 of Ethereum's PoW - turning miners into mere block proposers and introducing a PoS (partial) consensus engine in the smart contract layer - can also be applied to finalise an independent (auxiliary) system with a validator set whose stake is held on an external (origin) blockchain.

Our final goal here is clear to see: by defining assets first on the origin blockchain before moving them onto the auxiliary blockchain economic finality as agreed upon by the validators staked on the origin blockchain is decisive. The nodes participating in the layer-1 auxiliary blockchain are forcefully turned into block proposers if their rewards are also awarded in assets held by the origin blockchain. By externalising all stake and all value of all assets, the auxiliary system is economically bound to follow the fork selection rule of the Mosaic validators.

Lastly, we introduced a transition object hash $\tilde{T}$ in the vote message to externalise otherwise intrinsic properties of the auxiliary system. This transition object allows the Mosaic validators to coarse-grain and abstract the auxiliary system into a meta-blockchain when proposing meta-blocks on origin.

3 A mosaic of cores

4 Outlook

5 Conclusion

6 Appendix

References