Mosaic

RUNNING META-BLOCKCHAINS TO SCALE DAPPS

Benjamin Bollen, Martin Schenck
for OpenST Foundation
working draft

Abstract

Mosaic is a set of Byzantine fault tolerant (BFT) meta-blockchains defined in smart contracts on Ethereum. A meta-blockchain has an open set of validators staked on Ethereum, and has a smart-contract based consensus engine. Each block is committed by a super-majority vote. Mosaic introduces BFT consensus rules that allow validators to leverage more efficient auxiliary systems running in parallel for block production, while millions of transactions can be finalised before the header is asynchronously committed to Ethereum. Ethereum remains the sole source of truth by locking value tokens on Ethereum while minting a utility token representation on the auxiliary systems. Validators get rewarded collaboratively based on gas consumed thereby solving for the verifiers dilemma. As a result Mosaic can finalise thousands of transactions per second on Ethereum at a lower gas price. We have built an atomic gateway protocol that Ethereum DApps can deploy to leverage Mosaic to scale.

1. Introduction

2. A Mosaic of Cores

A meta-blockchain or core is defined in a smart contract by a genesis state, a staked validator set and consensus rules to progress the state.

2.1. OpenST-Mosaic

We assume without loss of generality that for every validator-address v^O in the validator set $\mathcal{V}^O$ on origin a corresponding, unique validator-address v^A on auxiliary A can be associated and we call the resulting validator set $\mathcal{V}^A$, the validator set on A. A single actor is assumed to control and be responsible for both v^O and the associated v^A. On origin v^O will always be held accountable for any signed message by v^A on A, because and only v^O has stake on origin.

Lemma 1 (Remote Accountability)**.** *A vote on A that violates a Casper Commandment can be punished on O.*

Proof. Given lemma **??**, a violation of either Casper Commandment can be proven on O without the respective transaction residing on O. If a validator ν^A published two distinct votes for the same target height, the relevant votes can be presented on O and the signature can be validated. An offending validator ν^O can be identified according to lemma **??** and its deposit slashed. The same is true if a validator ν^A votes within the span of its other votes. All relevant votes including their signatures can be presented on O. □

Theorem 2 (Partial View)**.** *Given Origin O with blocks b_i^O, checkpoints c_i^O can be reported on Auxiliary A, such that a set of validators $\mathcal{V}^O$ with stake on O can reach finality about the reported checkpoints c_i^O on A as $\mathcal{V}^A$ with accountable safety enforced on O.*

Proof. If you report checkpoints from Origin as $\langle h_c, h(c)\rangle$ where h_c is the hash of any checkpoint c on Origin and $h(c)$ is the height of checkpoint c in Origin's checkpoint tree, then you can vote on Origin's checkpoints

on Auxiliary according to the Casper FFG voting rules. A Casper FFG vote consists of two checkpoint hashes, two checkpoint heights, and the validator signature, all of which is now available on Auxiliary.

Accountable safety is enforced according to lemma 1. □

Theorem 3 (Leveraged Security)**.** *Given that the auxiliary chain A follows the Casper fork choice rule and given that the validators staked on Origin O, Casper can be applied to A as a whole; without validators staking on A. It is not necessary to report blocks or block headers from A to O.*

Proof. Validators staked on Origin

Lemma 1 proves that the validators can be slashed on Origin for their actions on Auxiliary. Even though Auxiliary is not necessarily lively, Casper still has plausible liveliness, as Casper assumes that the underlying chain keeps producing blocks. □

Reporting a finalised auxiliary checkpoint to Origin costs gas on Origin. Therefore, validators do not have an incentive to copy the state root from Auxiliary to Origin. We present an asynchronous mechanism that forces validators to copy the state root. We define the amount of consumed gas since the last copied checkpoint g. We define a limit of consumed gas l. The finalised auxiliary checkpoint with the smallest dynasty where $g \geq l$ must be copied to Origin.

$$\neg\exists f : h(f') < h(f) \wedge g(f') \geq l \tag{1}$$

If the validators fail to copy that finalised checkpoint, they are still incentivised to copy a later checkpoint to get rewarded (see below). When a validator copies a state root of a later finalised checkpoint from Auxiliary to Origin, then it can be proven that there exists a finalised checkpoint with a lesser height that is also above the gas limit. To do that, a fisherman could present the earlier finalised checkpoint to Origin after the later one was committed. The height, consumed gas, and validator signatures can be proven. When that happens, all validators in the validator set are punished and the fisherman is rewarded.

We must track the consumed gas on Auxiliary in order to for this proof to work.

Theorem 4 (Honesty Guard)**.** *An honest validator will not be punished by the gas limit rules.*

Proof. All validators, including the honest ones, would be punished if the lowest finalised auxiliary checkpoint above the gas limit were not reported to Origin. If there is at least one honest validator, however, that validator would report the auxiliary checkpoint to Origin. Thus, all validators get only punished if none of them is honest. □

2.2. Gateway

2.3. Reward Structure

2.4. Dynamic Set of Validators

3. Outlook

4. Conclusion

References