OpenST

A FRAMEWORK FOR BUILDING TOKEN ECONOMIES

Benjamin Bollen, Martin Schenck
for OpenST Foundation

Abstract

We present improvements to the OpenST protocol. OpenST is a framework powered by Ethereum to build token economies. We lay out in detail two contributions, OpenST Mosaic and OpenST Gateway, which work together to scale Ethereum OpenST Mosaic is a layer-2.

1. Introduction

2. OpenST

2.1. Use Cases

2.1.1 BT

2.1.2 DApp

3. Related Work

Verifiers' Dilemma

Interblockchain Communication

Casper FFG [1]

4. Our Contribution

5. Openly Scaling Blockchains

5.1. OpenST Gateway

5.2. OpenST Mosaic

Mosaic transfers blockchain state roots from a source to a target. An independent, trustless validator set observes both chains and votes on blocks.

Validator $\mathcal{v}$. Validator set $\mathcal{V}$. Origin $\mathcal{O}$. Auxiliary $\mathcal{A}$.

Gateway contract g. Gateway $\mathcal{G} = (g, \overline{g}, t_m)$.

State root SR. State object's storage root SO.

Block on origin $b_{\mathcal{O}}$. Block on auxiliary $b_{\mathcal{A}}$. OSTblock $\mathcal{B}$. Block at height i $b_{\mathcal{O}i}$. OSTblock at height α $\mathcal{B}_{\alpha}$.

OSTBlock gas limit OL.

5.2.1 Observing the origin chain

Validators transfer state roots bi-directional. We will first demonstrate how they transfer the state root from origin as the source to auxiliary as the target. In order to apply Mosaic universally, we abstract away from the consensus algorithm of origin and layer Casper [1] on top of the chain. On auxiliary, validators from the validator set vote to justify and finalise blocks from origin according to the Casper rules. In order to do so, a validator proposes the block header of a checkpoint from origin to auxiliary. The other validators in the current validator set vote on the block. The block gets justified with a $\frac{2}{3}$ majority as per Casper. Blocks that will be proposed in the future must include all previously finalised blocks in their history. This is not checked computationally on auxiliary, but rather by the validators by means of voting.

As a result, validators transfer state roots from origin to auxiliary in regular intervals. When the Gateway transfers state objects to auxiliary, their existence on origin can be proven against the state root that is now stored on auxiliary.

Origin cannot follow Casper's fork choice rule when the justification and finalisation is only recorded on auxiliary. Furthermore, we cannot expect origin to adapt to Mosaic's requirements. To avoid conflicting forks of origin, we propose to trail behind origin's head by a specific number of blocks to make a conflict unlikely. However, if at any time a conflict does arise, auxiliary must agree to switch to origin's branch. There is an economic incentive to do so as auxiliary could not be finalised anymore and any earnings on the auxiliary chain were practically lost. Coordination of auxiliary's miners or stake holders should be done off-chain.

5.2.2 Leveraging origins security

5.2.3 Asynchronously finalising on origin

In this section we will demonstrate how auxiliary is finalised by copying its state root to origin. In order to be able to universally apply Mosaic, we abstract away from the consensus algorithm of the utility chain and layer Casper [1] on top of it. Validators vote on auxiliary's checkpoints as per the rules defined in the Casper paper. When a checkpoint is finalised, its state root can be transferred to origin. To transfer the state to origin, a validator of the current validator set $\mathcal{V}_\alpha$ proposes the closing of the current OSTblock.

5.2.4 OSTgas

OSTgas is auxiliary's alternative to gas on Ethereum. Transactions pay gas to the miners or stake holders. And miners or stake holders pay gas to the validators in order to finalise their state on origin.

Validators need to agree on when to transfer a finalised checkpoint of auxiliary to origin. Generally, validators can transfer any finalised checkpoint. However, that leads to a transaction on origin and therefore incurs costs.

5.2.5 Rewarding validators

5.2.6 Variable validator set

6. Securing Simple User Experience

6.1. Token Holder Contracts

APIs; no need to follow the chain.

7. Platform

7.1. Token Rules

7.2. OpenST Platform

8. Outlook

8.1. Neo and Cardano

9. Conclusion

References

[1] Vitalik Buterin and Virgil Griffith: Casper the Friendly Finality Gadget (2017) URL `https://github.com/ethereum/research/tree/master/papers/casper-basics`.