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

!!!Wrong due to lack of knowledge!!!

An OpenST Gateway $\mathcal{G}$ transfers state objects between two chains. The transfer is bi-directional. Transfer of one state object SO takes place from a source blockchain to a target blockchain. To enable bi-directional transfer, two Gateway smart contracts g and $\overline{g}$ must be deployed, one per chain. A transformation function t_m defines a mapping of a source state object to a target state object. Thus a Gateway can be written as $\mathcal{G} = (g, \overline{g}, t_m)$ where g is the smart contract on the source chain and $\overline{g}$ is the smart contract on the target chain. A simple case for t_m is the transfer of the storage root hash of a contract.

Transfer happens in two phases:

1. Confirm certain event has taken place on source

2. Process certain action on target

5.2. OpenST Mosaic

Mosaic transfers blockchain states between two blockchains. The origin blockchain $\mathcal{O}$ is assumed to have liveliness and BFT security properties. The auxiliary blockchain $\mathcal{A}$ is not assumed to have any security features. Mosaic finalises the state of $\mathcal{A}$ on $\mathcal{O}$ and confirms the transfer securely on $\mathcal{A}$.

An independent, trustless validator set $\mathcal{V}$ observes both chains and votes on blocks on both chains. We layer the Casper FFG Proof-of-Stake [1] algorithm on top of the chains to be block proposal mechanism agnostic.

5.2.1 Observing the origin chain

We want to be able to observe $\mathcal{O}$ from $\mathcal{A}$. Since $\mathcal{A}$ is not assumed to have any security features, we need to know whether the state of $\mathcal{A}$ has been successfully transferred to $\mathcal{O}$. Only then can we assume that the given state of $\mathcal{A}$ is final.

On auxiliary, validators from the validator set vote to justify and finalise blocks from origin according to the Casper set of 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 proposal. The block gets justified with a $\frac{2}{3}$ majority as per Casper.

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 a Proof-of-Work 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 fork. 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 and is not discussed in more detail in this paper.

5.2.2 Leveraging origins security

Once a certain state of $\mathcal{A}$ is finalised on $\mathcal{O}$, it is considered final in the sense that blocks of $\mathcal{O}$ can be considered final. By finalising $\mathcal{A}$ on $\mathcal{O}$, we apply the security properties of $\mathcal{O}$ on the finalised checkpoints. Therefore $\mathcal{A}$ can be considered secure in certain intervals.

5.2.3 Asynchronously finalising on origin

To compress auxiliary, consider the following. A transaction is a transformation from one state to another. A block is an ordered set of transactions, moving from one state to another. You could write a block as a transaction that transforms the state from the input of the first transaction of the block to the output of the last transaction of the block. If you consider blocks on auxiliary as transactions, you can form blocks with an ordered set of these transactions. These blocks we call *OSTblocks.*

In this section we will demonstrate how auxiliary is finalised by copying its state root to origin. Validators vote on auxiliary's checkpoints on auxiliary 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

The validator set should be variable and open. Validators can join and leave the validator set by sending a *deposit message* or a *withdraw message*, respectively. The rules from Casper apply and the validators will

join and leave the validator set at dynasty $d + 2$ after their message has been processed in dynasty d. When a validator sends a *deposit message* at OSTblock height B_α

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`.