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 [1]

Interblockchain Communication [2]

Casper FFG [3]

4. Our Contribution

5. Openly Scaling Blockchains

5.1. OpenST Gateway

OpenST Gateway enables chain-to-chain transfer of state objects.

5.2. OpenST Mosaic

OpenST Mosaic enables chain-to-chain state transfer. That allows us to secure an insecure chain's state on a secure chain. We call the secure chain *origin* $\mathcal{O}$ and the insecure one *auxiliary* $\mathcal{A}$. In order to be blockchain proposal mechanism agnostic, we layer Casper [3] on top of any auxiliary chain. We apply Casper to auxiliary the same way its application to Ethereum is described in the paper. We assume origin to have at least two properties: practical byzantine fault tolerance (PBFT) [4] and plausible liveness [3]. Furthermore, we assume origin to have a concept of finality or at least economic finality.

Mosaic consists of Mosaic nodes that observe both $\mathcal{O}$ and $\mathcal{A}$ as well as a specific Gateway like the ones described in section 5.1. That means every Mosaic node runs a full node per chain, an independent client software that observes both full nodes, and there is one smart contract per chain that makes up the Gateway.

It is required to move state from auxiliary to origin in order for the state to be finalised. State is only assumed to be final on origin, as we do not assume any security properties of auxiliary. On the other hand it is required to move state from origin to auxiliary, because auxiliary must be made aware of the fact that its state has become final. When we say that we transfer state, we actually refer to the transfer of the root of the state tree. Any state can be proven against a known root.

Validators are actors that vote on state that is transferred between the two chains. In addition, on auxiliary, they take the role of the Casper [3] validators. Their deposit, however, is stored on origin. Similarly to Casper, any validator's deposit rises and falls with rewards and penalties, respectively. When we say "$\frac{2}{3}$ of validators", we equally refer to the deposit-weighted fraction. However, there are different kinds of voting message types, as described below.

The set of validators needs to be able to change. The logic is similar to that of Casper with the notable difference that, instead of dynasties, we use OSTblocks (described below). When a potential validator sends a *deposit message* on origin at OSTblock height α, then the validator ν will join the validator set at OSTblock height $\alpha + 2$. Analogous to Casper, we call $\alpha + 2$ the validator's *start OSTblock*, $\mathrm{BS}(\nu)$. If a validator ν's withdraw message is included on origin's blockchain during an open OSTblock $\mathcal{B}_\alpha$, it similarly leaves the set of validators at OSTblock height $\alpha + 2$. We call $\alpha + 2$ the validator ν's *end OSTblock* $\mathrm{BE}(\nu)$.

Validators can vote on a number of events: (a) origin state transfers to auxiliary, (b) checkpoint justifications on auxiliary, and (c) auxiliary state transfers to origin. On auxiliary, checkpoints are justified and finalised by supermajority links as described in Casper. Origin's (economical) finality is assumed as a given. Thus, when state is transferred between the two chains, a simple $\frac{2}{3}$ majority vote on that state commits it to the receiving chain.

OSTblocks are a compressed representation of auxiliary's transactions. Transactions in an OSTblock each represent one block of the auxiliary chain. An OSTblock transaction represents the state change from the beginning of its associated block to the end of that block. Therefore, OSTblocks are generated at a slower pace. We call α the height of OSTblock $\mathcal{B}_\alpha$. The next OSTblock, that points to $\mathcal{B}_\alpha$ in its header, is OSTblock $\mathcal{B}_{\alpha+1}$ at height $\alpha + 1$. Due to the aforementioned compression of blocks on auxiliary, $\mathcal{B}_\alpha$ may contain auxiliary blocks $b_{\mathcal{A}i}$ to $b_{\mathcal{A}i+m}$.

The OSTblock header of block $\mathcal{B}_\alpha$ at height α contains:

1. A link to the previous OSTblock
2. The signatures of the validators in $\mathcal{V}_\alpha$
3. The state root of the last block of auxiliary that is contained within the OSTblock
4. The set of validators for the next OSTblock, $\mathcal{V}_{\alpha+1}$

The body of an OSTblock consists of all the transactions that make up the blocks on auxiliary that are contained within this OSTblock. We do not store the body of an OSTblock on origin, only the header. The body can be re-created at any time from the auxiliary chain and proven against auxiliary's state root that is securely stored on origin.

We consider an OSTblock *closed,* when the full header with the state root from auxiliary is committed to origin. We consider an OSTblock *opened,* when the new OSTblock header has been transferred to auxiliary.

In order to be block proposal mechanism agnostic, we layer Casper [3] on top of auxiliary. Validators vote on supermajority links. Auxiliary must follow Casper's fork choice rule. Checkpoints get finalised. The Mosaic validators can transfer the state of a finalised auxiliary checkpoint to origin. A validator uses the Mosaic Gateway to transfer the state to origin. On origin, a $\frac{2}{3}$ majority vote is required by the validators in order to commit the state to origin. The state is regarded economically finalised when the block that contains the commit is considered economically final on origin.

On the auxiliary chain, *OSTgas* is the equivalent to gas on Ethereum. For transactions to be included in blocks on auxiliary, a fee is paid for the consumed OSTgas.

Validators are allowed to commit every finalised state from auxiliary to origin. However, that may be infeasible from an economic viewpoint of the validators. Therefore, we propose points on the auxiliary chain where the

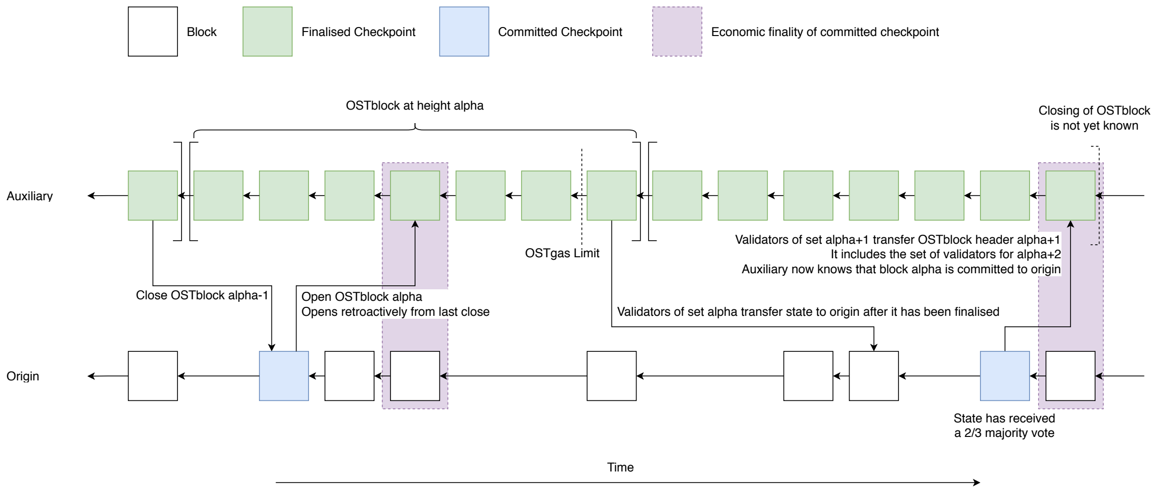
Block
Finalised Checkpoint
Committed Checkpoint
Economic finality of committed checkpoint
OSTblock at height alpha
Closing of OSTblock is not yet known
Auxiliary
OSTgas Limit
Validators of set alpha+1 transfer OSTblock header alpha+1
It includes the set of validators for alpha+2
Auxiliary now knows that block alpha is committed to origin
Close OSTblock alpha-1
Open OSTblock alpha
Opens retroactively from last close
Validators of set alpha transfer state to origin after it has been finalised
Origin
State has received a 2/3 majority vote
Time

Figure 1: **The Mosaic state transfer.** The closing of $\mathcal{B}_{\alpha-1}$ means that the finalised checkpoint $b_{\mathcal{A}i}$ is the last block of $\mathcal{B}_{\alpha-1}$. Even though the details of $\mathcal{B}_{\alpha}$ will not be known until the opening is transferred to $\mathcal{A}$, the first block in $\mathcal{B}_{\alpha}$ will be $b_{\mathcal{A}i+1}$. Once a checkpoint is finalised after the OSTgas limit has been reached, $\mathcal{V}$ will signal the closing of $\mathcal{B}_{\alpha}$ to $\mathcal{O}$.

validators must close the OSTblock. Whenever a certain fixed limit of OSTgas has been spent on auxiliary since the opening of the OSTblock, the validators *must* close the OSTblock at the next finalised auxiliary checkpoint. If a validator does not vote to commit that checkpoint, its deposit can be slashed. The OSTgas limit is marked in figure 1 by the dashed line.

Actors on the auxiliary chain have an economic incentive to finalise their state on origin. The block creators of auxiliary will pay a fraction of their earned OSTgas to the validators as a fee when the validators move the state to origin. Due to the OSTblocks, validators can only copy the state from auxiliary to origin after the state of origin has been copied to auxiliary. Thus, validators are incentivised to close an OSTblock and open a new one in order to earn the OSTgas fee for the transfer of auxiliary's state.

It must be possible to hold validators accountable if they misbehave. In order to do so, anyone must be able to present proof of malicious actions on origin, as the validators' deposits are stored there. When a validator violates a slashing condition on auxiliary by making an illegal vote, that fact is stored in the state of auxiliary. Therefore, it can be proven against the state root of auxiliary. Since the state root of auxiliary is available on origin, the proof and the subsequent slashing can be carried out on origin as well.

To prevent long range revisions, we apply the same rules that Casper applies: after a validator leaves the set of validators, its deposit is kept for another four months. Casper votes on auxiliary can only be carried out up to three months into the past. That leaves one month headroom to blame an adversary on origin and get its deposit slashed.

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] Luu, L., Teutsch, J., Kulkarni, R. & Saxena, P. Demystifying incentives in the consensus computer. *IACR Cryptology ePrint Archive* **2015**, 702 (2015).

[2] Kwon, J. & Buchman, E. Cosmos: A network of distributed ledgers (2017). URL `https://cosmos.network/docs/resources/whitepaper.html`.

[3] Buterin, V. & Griffith, V. Casper the friendly finality gadget (2017). URL `https://github.com/ethereum/research/blob/master/papers/casper-basics/casper_basics.pdf`.

[4] Castro, M., Liskov, B. & et. al. Practical byzantine fault tolerance. In Leach, P. J. & Seltzer, M. (eds.) *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, vol. 99, 173–186 (1999).