LEVIATHAN
Turning Liveness Into Finality

ZYGOMEB
Optim Labs
zygomeb@gmail.com

January 3, 2024

1 ABSTRACT

We present the Leviathan scaling protocol, which under assumption of liveness and deterministic transactions with exclusive state access on the settlement chain (equivalent to Cardano's eutxo) allows for the instant finality transaction layer with increased throughput.

Leviathan, to achieve its full potential, however, requires an account abstraction primitive for users to leverage the full potential of its latency improvements. This is where an Account Abstraction is leveraged, coming together into a framework within which we recoup composability of Cardano protocols.

What has been seen to be a task facing significant challenges between batchers breaking atomic composability of DEX transactions, and latency concerns for apps requiring fast confirmations on liquidations are all resolved by way of our unified framework. Within this paper only the scaling solution of Leviathan is outlined, leaving the description of the account system for another document.

While a solution like Hydra has, when the assumption of a single honest participant is violated, its isomorphism with the base layer broken, and is able to take control of all funds within it –

Leviathan maintains the equivalence with the base layer at all times, as a fully embedded execution layer conforming to the rules of the L1's spending rules, and under compromise of the sequencer network at worst it shall halt, thereby granting no custody of these funds to the network.

Even when constrained by the data throughput of the settlement layer, the Leviathan protocol allows unconstrained state growth and instant finality induced by, and only limited in settlement time by the sequencer network.

2 INTRODUCTION

A **Deterministic Transaction** is defined to be one that, were the input state to it change for the data it interacts with, it would become invalid. In presence of a virtual machine that runs custom validation rules beyond that of the ledger, that virtual machine must also have its result uniquely defined by the input state.

This general definition is broad enough to be easily made use of for more general application, but for the sake of mental modeling it is important to reiterate that we're working on Cardano's eutxo model. Of particular importance is the fact that this deterministic validation receives no additional input between when the transaction is made and signed and when it is included in a block, meaning that it acquires no dependency on, for example, the exact block hash, timestamp, or otherwise. It is important to state, as there is a class of protocols which change the execution context of the transaction the moment it is included in a block vs when executed beforehand.

Bringing us to the **Guaranteed Transaction** is one which is deterministic, and, has no input to it which can change or become invalid. This is a transaction which is completely independent of any time-based validity range, and, has a guaranteed exclusivity to the spending rights to the inputs to it.

In this work, we operate on this class of transactions and build a framework within which such transactions can be operated with, allowing us to take advantage of the logical consequence of operating only on them:

Guaranteed Transactions under assumption of the liveness of the ledger, have instant finality.

And furthermore:

Guaranteed Transactions can be used to construct a chain of state changes without the bottleneck of the settlement ledger's throughput capacity.

Let's take a moment to unpack that second statement on our practical example. Using the terminology of our Transaction Chaining article, we take as inputs to the N+1-th transaction the outputs of the N-th, without any required ledger confirmation of the N-th transaction. The first transaction in this sequence we call "Transaction Chain", is based on the inputs final in the ledger. Therefore by induction, if each transaction in the sequence is a Guaranteed one, and under assumption of liveness, we prove that the whole sequence is final.

Of note is the fact that as long as this sequence is finite, no matter how long, the induction process allows us to assert this finality, so long as the network does not invalidate these transactions through an update or other dynamic processes. We may also call this sequence the **Leviathan Chain Extension**.

3 IMPLEMENTATION

It is worth noting that, in order to fulfill the requirements for a transaction to be Guaranteed, we must separate the user from the action, i.e. necessarily decouple the user from the deterministic nature of this transaction, as otherwise both the latency and throughput benefits yielded from Leviathan are diminished. A common sequencer network must be formed that, through whatever smart contract encoded consensus, must form the Transaction Chain.

For participant tokens to be consumed in transactions made in a guaranteed transaction chain we must construct an account abstraction layer that allows transactions to be made on behalf of the user, given witness to their desire (a signed message, or an intent).

In such a setup, while the network, depending on its implementation, may halt, unlike a traditional rollup it needs no fraud proofs or multisignature guards to make sure that its continued honesty is a given.

The main problem of using Leviathan as opposed to a traditional rollup, is that for the increased security of leveraging L1 transactions directly, it makes no gain in their cost and yields no data compression. Both of which mean that economically Leviathan may be quite costly (on par with the L1 chain) relative to a rollup.

Furthermore, the extraordinarily high demand for a transaction to be considered guaranteed makes implementation of any such system extraordinarily hard. All of the transaction inputs would ideally be subject to a separate chain transaction ordering consensus, passing proof to the settlement L1's Leviathan chain extension via cryptographically signed proof of the correct ordering. As far as current implementation is possible, fully decentralized guaranteed transactions are subject to further research and development, given certain ledger constraints that make it more difficult.

3.1 Account Abstraction

Why do we even need it? In the process of trying to create a transaction type which would have the guaranteed property and marrying it with user input we arrive at a point where we must both take a message from a user and let a party (the sequencer) have complete control over the L1 transaction creation. In other words, we need an account holding the user's funds that lets them signal to the network what kinds of transactions they want to make. In other words, signal their intent.

A key factor to consider when designing the correct abstraction for the account is that, calling it an account abstraction is a slight misnomer. While this may be done and designed merely as an account abstraction, the opportunity granted by having such a framework open, and a requirement to be entirely subject to the sequencer's ordering, mean that all applications would have to be entirely contained and free of context outside of the account world. A reminder, for Guaranteed Transactions we need full static context of the transaction. These applications may not do anything unless the sequencer wills it. More on the topic of the sequencer in the next subsection.

Henceforth we assume the sequencer to be honest. Zooming into the world of Cardano's eutxo specifically, the extension of the abstraction into a framework, an execution layer unto itself, yields a few core benefits. Notably, it establishes the system for delivery of arbitrary oracle or time inputs in real time, and depending on its design, new types of tokens (for example, full erc20 spectrum). As far as application development goes on the Leviathan Chain Extension, within this framework and for practical purposes, we may disregard any and all concurrency or throughput blockers that were previously ones on the L1.

The only concern within the large concept of Leviathan is that it yields no space benefits nor does it yield any cost benefit alone. This can be amended by making the Account layer built using Zero Knowledge techniques, and unto an image of a ZK Rollup.

3.2 Sequencer Network

The crux of the practical implementation of the abstract concept of a guaranteed transaction is the Sequencer, a party (possibly a sidechain) with executive authority to elect blocks in the chain extension.

Decentralization of a sequencer network is one of the biggest challenges facing protocols of similar characteristics to Leviathan, many opting to start off centralized and gradually transition to a more decentralized implementation.

3.3 Time Travel

For the Leviathan Chain Extension, L1s Present is the Past, and its Present is the Future. We must navigate that fundamental time dilation tactfully. Of particular note is the entry into the Future and exit into the Present. This entry is in and out of the contract's purview and may be regarded as semantically similar to bridging.

Tying into this problem is the travel into and out of the Future, as it necessarily must either wait for the Past to reach the Present, or wait until Finality in order to teleport into the Future. Similarly, an application may exist both in the Past and the Future, and one such application would be one that manages the stake of the L1 ledger while allowing the usual in-and-out of the Future system behavior.