0L - ETH Bridge

High level summary:

0L bridge is an "optimistic" bridge which leverages the principles of the Optics design: https://docs.celo.org/celo-codebase/protocol/optics.

Transaction can be submitted from a "home' chain to an "away" chain. Transactions are final after a certain amount of time (blocks) have passed on the away chain. Within that period a fraud proof can be submitted to stop the transaction and disqualify the bridge operator.

Merkle-trees on both home and away chains, provide gas-efficient transactions. Fraud proofs in the form of merkle inclusion proofs can be submitted to catch a malicious bridge operator.

Bridge operators are selected from the home (0L) chain's validator set. Validators are expected to run the services necessary for bridge operation alongside the usual node operations.

Definitions:

Orchestrator - A rust program for signing transactions on ETH, and querying ETH state,
Oracle.move - The 0L stdlib module for tallying votes.

- Only for 0L GAS deposits
- Escrow account

Escrow.move - To be implemented, move contract for depositing and refunding GAS coins.
ETH Contracts - A collection of contracts owned by 0L validator set on ETH.

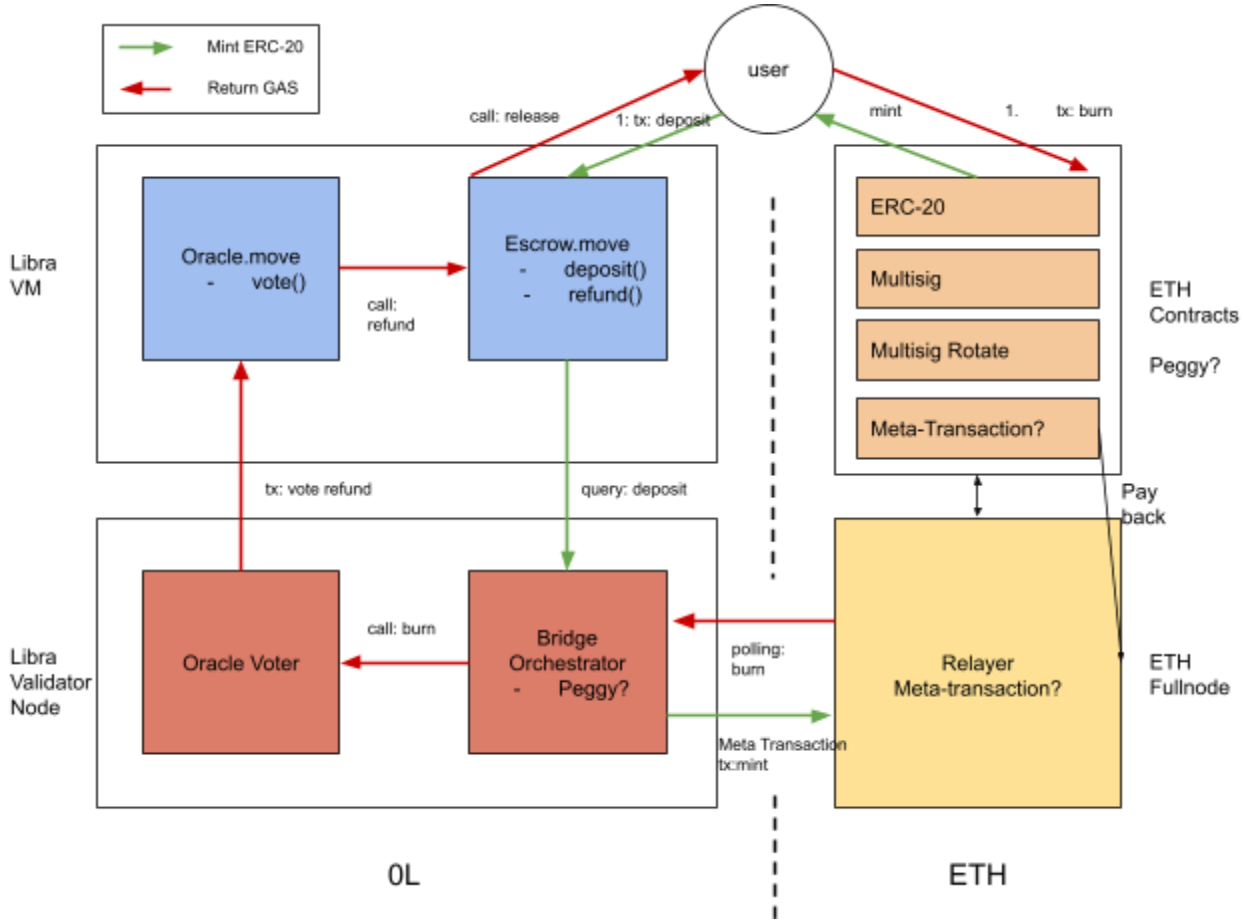
- ERC 20, mint and burn
- The Optics "socket" contract

ETH fullnode - A third-party http service which provides transaction forwarding to, and queries from ETH chain.
GAS -- Gas tokens on 0L.
eGAS -- “Wrapped” OL gas tokens on Ethereum.

Eth Contracts include:

ERC-20: a contract for wrapped GAS (eGAS), with mint and burn functionality.
Multisig: a contract which allows dynamic changes to the set of signing keys.
Upgrade: capability for the ERC-20 contract (and other contracts) to be upgraded to a new contract based on a vote by the multisig members.

Multisig Rotation: TBD how the multisigs which can submit



Requirements

Orchestrator

1. The Orchestrator must be able to reliably query state and execute transactions on both the Ethereum and OL blockchains.
2. When a user transaction on OL locks new GAS tokens into Escrow.move, the Orchestrator must recognize this event and faithfully vote on Ethereum for a corresponding quantity of eGAS tokens to be minted.
3. When a user transaction on Ethereum burns a quantity of eGAS tokens through the ERC-20 contract, the Orchestrator must recognize this event and faithfully vote on OL for a corresponding quantity of GAS tokens to be released to that user on OL.

4. When the validator set changes on OL, the Orchestrator must execute an Ethereum transaction to cast a vote to remove the old validator and promote the new validator in the ERC multisig.

Prior art for Orchestrator:

The Optics Orchestrator for listening and writing transactions to Eth chain:
https://github.com/celo-org/optics-monorepo/tree/main/rust

The Peggy/Gravity orchestrator.
https://github.com/PeggyJV/gravity-bridge/tree/main/orchestrator

ETH Contracts

1. The ETH Contacts must provide a function allowing multisig members to mint new eGAS tokens and assign them to a given user (pending majority vote)
2. The ERC-20 contract for eGAS must provide a function that allows an Ethereum user account owning eGAS tokens to burn them.
3. The Multisig Rotate contract must provide a function allowing multisig members to cast a vote to remove a given key and another function allowing multisig members to cast a vote to add a given key.
4. If a majority of multisig members votes to remove a given key, then that key must be removed from the multisig. Similarly, if a majority votes to add a new key, that key must be added.
5. Every ETH contract must be upgradable based on a majority vote by the multisig.

Move Contracts

1. The Escrow.move module must provide a deposit() function allowing OL user accounts to call the function to lock GAS tokens in escrow.
2. The Escrow.move module must provide a refund() function allowing OL validator accounts to cast a vote to release escrowed tokens back to a given user account.
3. The Escrow.move module must enforce that only validator accounts may call the refund() function, and that tokens are only released if a majority of validators cast the same vote to release them.

Research topics:

Does Escrow.move listen for events?
Can the oracle voter exist outside of the STDLib?

Validators will be paying Ethereum transaction fees. How are they incentivised to do so?

- Obviously when tokens are being moved back from Ethereum → OL then the user requesting the transaction can pay the fees. But what about going the other way?