Consensus protocol Working Group

1. Motivation

There are some aspects we consider required about how validator nodes operate to enable access to open main-nets and be regulatory compliant at the same time. We foresee the need of updating the consensus protocol in order to incorporate these functionalities.

2. Main Goals

- Set an optimal number of active validator nodes.
- Writer node signature required for transactions.
- Prevent empty block generation.
- Achieve automatic rotation of validator nodes according to performance and decentralization thresholds.

3. Participants

- Tech Lead: Diego López León
- Coordinator: Antonio Leal Batista
- Supervisor: Marcos Allende López
- Partners
    - <partner 1>

4. Detailed description

4.1. Optimize number of validators

There is no hard limit to the number of validator nodes participating in the IBFT 2.0 rounds. We need to define a number that performs well in terms of decentralization but also from a practical point of view.

Blocks minted within the IBFT 2.0 protocol contains all the necessary information for validation contained into the extraData field. This information is an RLP encoded list with the following elements:

- 32 bytes of data from the --miner-extra-data CLI param.
- Round validators addresses (unbounded, 20 bytes per address).
- Round number.
- Signatures of validators (potentially as much as validators but no less than required quorum, 65 bytes per signature).

As may be observed, a long list of validators will increase the block size with impact on the bandwidth usage, potentially affecting our expected transaction throughput.

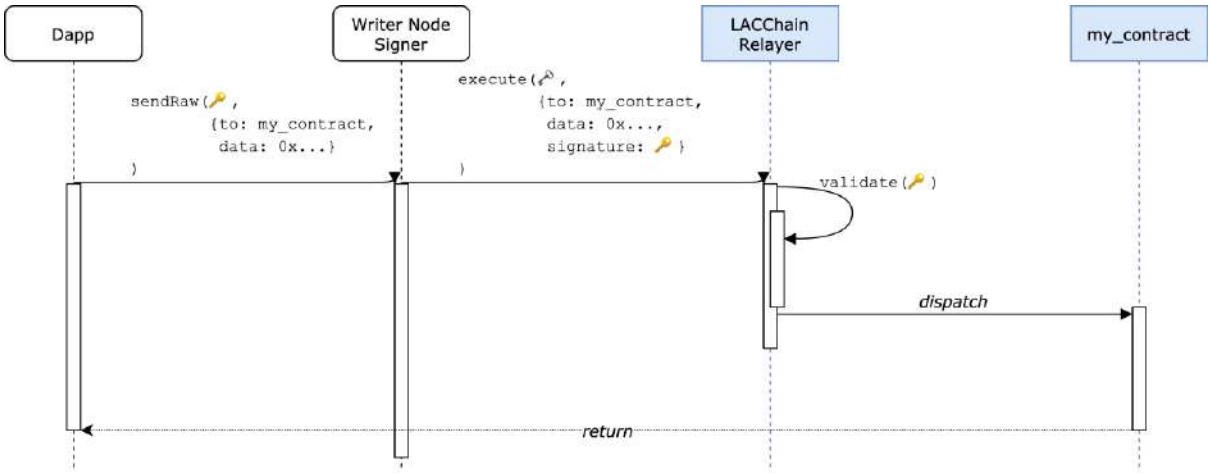
Technical milestones:

- Analyze current network topology latency:
  - The goal of this task is to evaluate current state of decentralization and define expectations of the network distribution.
- Analyze current transaction throughput:
  - By stressing the network, we should observe if we can reach the theoretical goal set by the combination of `blockperiodseconds` and `gasLimit`, and how an increasing list of validators may affect it.
- Define expected transaction throughput per second:
  - With the throughput obtained we must define a minimum number of transactions per second in terms of scalability for its usage.
- Define a maximum block size to reach the expected throughput with a theoretical latency:
  - Using the throughput obtained before, we should analyze how an increment in the block size by adding new validators could affect network performance without compromising transaction scalability.

4.2. Writer nodes permissioning

The LACChain Ethereum network uses a smart contract permissioning mechanism provided by the Hyperledger Besu implementation.

In the mentioned smart contract, we can whitelist writer nodes to be the only ones enabled to send transactions to the network. Thus, if any third party wants to interact with the LACChain network they should go through the whitelist writer nodes before invoking the destination contracts. This can be achieved by a meta-transaction mechanism, where writers act as a proxy before invoking the target contract.



Thinking in LACChain as an infrastructure provider, being able to provide support for an Infura-like layer seems within its scope. This layer should be transparent to developers in their

experience/perspective, simultaneously will allow the LACChain network to gain control over who is responsible for the network usage.

Technical milestones:

- Develop proxy smart contract:
  - This smart contract is responsible for dissecting the meta-transaction to validate its structure and extract the original sender, smart contract destination and existing parameters.
- Define a protocol for interacting with LACChain enabled smart contracts:
  - Every target smart contract running in the LACChain Ethereum network should implement an interface to receive the original sender and the destination call. This working group has the responsibility to define and document that interface.
- Develop a server application mimicking JSON-RPC Web3 methods to wrap and sign third party transactions before sending them to the proxy:
  - Aiming to be as developer transparent as possible, the writer nodes will act as transparent proxies between users and the blockchain. This implies implementing a minimum set of JSON-RPC Web3 methods, so the most popular libraries have the proper support.
- Define infrastructure requirements to support scaling:
  - The writer nodes will become the entry point for users who wants to modify the blockchain state. Thus, the writer nodes are expected to be under heavy load; a list of optimal infrastructure requirements must be defined to ensure network good health.
- Perform security audits of the smart contracts, protocols and infrastructure.
  - Being this the entry point to the network it's our understanding it will become a first target for attacks. All these pieces must be heavily audited and support upgradeability under critical flaws.

4.3. Prevent empty blocks generation

The IBFT 2.0 consensus algorithm defines that a new block must be minted on every round. Under low network usage this generates unnecessary space consumed by the blockchain because most of the blocks will be empty.

Technical milestones:

- Analyze consensus algorithm to recognize valid empty blocks but prevent them to be added to the blockchain.

4.4. Automatic validator rotation

The IBFT 2.0 consensus implementation in Hyperledger Besu stores the list of validators within the block structure. This list can be modified by the permissioned validators through a voting mechanism, adding or removing a single member on each round.

We came up with two ideas listed below, even though we'll implement the later one.

The first one involves moving the list of validators into a smart contract. Doing so, we can gain the flexibility and security of on-chain operations. We could cast multiple votes per round or even let the LACChain permissioning committee to keep absolute control of this list.

Technical milestones:

- Create a *validators* smart contract and move the voting mechanism there.
- Evaluate moving the different validator sealings to became regular transactions.
- Modify *validators* contract to select an appropriate subset of validators enabled on every round.

The second idea is to use the existing voting mechanism but enforce validators to act under the command of the LACChain permissioning committee. We decided to go with this idea because it doesn't involve modifications on Hyperledger Besu that could take time to go into upstream.

Technical milestones:

- Create a *validators* smart contract for vote emitting:
    - The LACChain permissioning committee will control the list of validators.
- Create a *validators* software to listen to blockchain events and send votes to the network through JSON-RPC[1].
    - Every validator node should run this daemon and perform the necessary votes to enable the list of validators defined by the LACChain permissioning committee.

[1] https://besu.hyperledger.org/en/stable/HowTo/Limit-Access/Local-Permissioning/