Level 2 blockchain networks and extended consensus

Alexey Kiselev, Alexander Ivanov, Sergey Nazarov, Oleg Taraskin

Abstract: This white paper presents a novel strategy to enhance blockchain scalability through extended consensus mechanisms and re-using of economic incentives. By harnessing the consensus achieved in the primary blockchain layer, this approach simplifies and boosts the efficiency of subsequent layers. The proposed method enables cross-network interaction, ensuring secure communication between different layers. The paper provides an outline of consensus implementation, miner registration, and rollback calculations. Additionally, it addresses compatibility with Ethereum Virtual Machine networks and proposes future directions involving zero-knowledge proofs integration and optimization for streamlined interactions. Ultimately, this framework fosters an interconnected ecosystem of scalable networks while leveraging foundational consensus mechanisms.

1. Introduction. Re-staking and extended consensus.

At its core, blockchain serves as a global clock synchronization system, ensuring transaction order and providing a unified view of the global state. To achieve this, economic incentives or mechanisms like Proof of Work or Proof of Stake are employed. These mechanisms help in selecting a round leader to propose the global state view and make it difficult to reverse once accepted.

The consensus process in a global blockchain can be complex, with multiple state views competing for agreement, leading to forks and branching. It also brings about the possibility of rollbacks and lacks a clear finality of the state view in certain cases. However, once consensus is achieved and transactions are ordered, a global clock synchronization is established, which can be relied upon as long as the consensus remains intact. Consequently, this time ordering, along with the economic incentives, can be extended to establish a time ordering of transactions in related networks, such as Layer 2 (L2) built on top of the original Layer 1 (L1).

Addressing blockchain scalability is currently focused on creating multiple Layer 2 networks on top of a multi-purpose Layer 1 [2] [3] The primary function of Layer 1 is to provide global consensus and facilitate interactions between multiple Layer 2 networks. The consensus mechanisms in Layer 2 networks can be simplified or even eliminated if there exists an L2 entity responsible for transaction ordering and providing zero-knowledge proofs of state changes, verifiable on Layer 1.

However, this approach may not be optimal, as it could lead to reduced decentralization in Layer 2. For instance, if a sequencer-like entity is present in Layer 2, it may choose to censor specific transactions. As a result, a certain level of simplified consensus typically exists in Layer 2, although not as intricate as in Layer 1. Layer 2 consensus usually requires different economic incentives and mechanisms than those used in Layer 1, but it can leverage the established L1 consensus to some extent.

On the other hand, once the consensus is achieved on Layer 1, it can be utilized as a tool to facilitate consensus in Layer 2 as well. For example, a round leader in a related Layer 2 network can be selected through a smart contract on Layer 1. There is no need to run the full consensus process on Layer 2, such as selecting the longest branch, since it has already been performed on Layer 1. An L1 smart contract can also select the round leader based on the same economic incentives that exist on Layer 1.

This way, the same economic mechanisms on Layer 1 are reused in Layer 2, effectively exporting the Layer 1 consensus to Layer 2. This process of consensus export resembles "re-staking," where the same stake is used to secure multiple networks. By leveraging the established consensus on Layer 1, the redundancy is reduced, making the overall system more efficient and scalable.

2. General extended consensus implementation

For each new L2 network, a special contract ("chain contract") is deployed in L1. Miners commit to participating in a specific L2 network by invoking a dedicated function on the L1 contract. The commitment is valid for a specific time period. A certain minimum generating balance is required to commit to participation. Explicit commitment helps in miner selection during an epoch on L2.

Each block in L1 signifies an epoch on L2. Several L2 blocks may be produced during an epoch, mimicking the NG approach in Waves L1 and providing the similar scalability benefits. During an epoch, one of the miners with a valid commitment is selected to produce L2 blocks. The chain contract uses a modified Fair PoS formula [1] to calculate the next L2 block timestamp for each L2 miner: instead of a VRF (Verifiable Random Function), a hash of the VRF from the current block in L1 is used as a hit source. The contract will only accept blocks from the miner with the earliest timestamp. If the selected miner is offline, then the entire epoch would be empty.

An alternative approach would be to pre-select several miners for an epoch, and assign each of them a slot. In this case, if one of the miners is unable to produce a block in a timely manner, then only their slot in an epoch would be empty.

The miner packs L2 transactions in a block, signs it and invokes a chain contract method. The invocation basically saves L2 block metadata to the L1, so that L1 consensus can resolve possible forks. Block metadata consists of the reference to the previous block, timestamp, transactions Merkle root and L2 state hash. Chain contract ensures that the block metadata is valid.

The chain contract maintains the longest chain (by epoch count), and also the number of blocks which have been signed by half the stake of the miners committed to maintaining this L2 network. We'll call this number the maximum rollback depth.

In the first block of an epoch, the miner is free to choose the block to reference, the only limitation is that it can't reference the block beyond maximum rollback depth. Each subsequent block in this epoch must reference a previously submitted block. In other words, all blocks in an epoch can only form a chain, not a tree.

When the miner decides to reference one of the earlier blocks (not the most recent one), the chain contract starts tracking the new chain. The epoch which has the most recent common block of the two chains is called the fork site. The chain contract checks if the longest chain has changed at the beginning of each epoch. When the fork site goes beyond maximum rollback depth, the chain contract stops accepting new blocks referencing the shorter chain. We are able to enforce the absence of long-range forks by prohibiting rollbacks beyond the fork site block because the chain contracts enables an effective branch selection, dependent only on the L1 consensus.

3. L2NG - L2 consensus based on Waves L1

- Minimum generating balance is 20000 Waves (adjustable to match the L2 chain contract complexity limits)
- Only one L2 miner can post blocks during an epoch.
- Miner can either continue the main chain, fork a new chain from the main chain, or continue one of the existing chains. Forking from one of the alternative chains is not permitted.

3.1. Miner Registration and Selection

Miners commit to participation in L2 by calling the join() function of the chain contract. The contract ensures that the miner's generating balance is not lower than required and adds the miner to the participant list. At the start of each epoch, the contract iterates over the list of the participating miners. For each participant, the contract checks their generating balance. If the balance is lower than required, the miner is evicted from the list, and will no longer participate in L2 (unless they explicitly re-join).

The contract also calculates the "block delay" for each of the participants. The miner with the lowest delay will be allowed to post L2 blocks to L1 during this epoch. Block delay is calculated using a variant of the Fair PoS delay calculation formula, with take(blake2b256(VRF), 8).toInt() used as the hit source. The chain contract also calculates the total generating balance of the participating miners. Miners can leave at any time by calling the leave() function of the chain contract.

3.2. Posting Blocks

The chain contract accepts blocks only from the designated miner. The miner posts the first block of the epoch by calling either extendMainChain(), extendAltChain() or startAltChain() functions of the chain contract. The subsequent blocks are posted by calling the appendBlock() function. The subsequent blocks in this epoch can only reference the previous block posted by this miner. The miner can't create or extend several chains.

In addition to the process described in Miner Registration and Selection, the chain contract performs the following actions:

3.3. Extending Main Chain

The miner appends the first block of the epoch to the main chain by calling the extendMainChain() function of the chain contract. The chain contract:

- Ensures the miner is allowed to generate blocks in this epoch.
- Ensures the miner's active chain is either the main chain, or one of the inactive alternative chains.
- Calculates the new maximum rollback depth.
- Deactivates all alternative chains which have started below maximum rollback depth.
- Sets the miner's current chain to the main chain.
- Saves miner generating balance in this epoch for further finality calculations.

3.4. Creating Alternative Chain

The miner starts the new alternative chain by calling the startAltChain() function of the chain contract. The chain contract:

- Ensures the miner is allowed to generate blocks in this epoch.
- Ensures the block references one of the blocks above the maximum rollback depth.
- Ensures the miner's active chain is either the main chain, or one of the inactive alternative chains.
- Allocates the new identifier for this chain.
- Sets the miner's current chain to this chain.

Upon successful completion, the contract returns the new chain identifier.

3.5. Extending Alternartive Chain

The miner extends one of the alternative chains by calling the extendAltChain() function of the chain contract. The chain contract:

- Ensures the miner is allowed to generate blocks in this epoch.
- Ensures the miner's active chain is either this alternative chain, or their last block is below the starting block of this alternative chain.
- Sets the miner's current chain to this chain.
- Calculates the supporting balance for this chain (from the start of this chain onwards).
- If this chain is supported by more than a half of the total balance, this chain becomes the new main chain. All other alternative chains, as well as the former main chain, are deactivated.

3.6. Rollback depth calculation

- Start with the head of the main chain, an empty list of known miners, zero balance and height equal to current height.
- While balance not greater than half of the total generating balance of committed miners:
- If current block miner is not in the list of known miners:
  - Add current block miner to the list of known miners.
  - Add current block miner's generating balance to the total balance
- Set rollback height to current height.
- Set current block to the parent of current block.
- Return height.

4. Cross-network interaction

It is possible to maintain cross network bridgers based on the proposed set-up, ensuring liquidity and information transfer between level 1 and multiple level 2 networks connected to it. To ensure trustless interaction between networks cryptographic digests of various L2 networks state have to be posted onto L1 and also L1 state digest has to be posted on L2.

L2 state digest is posted to the main L2 contract on L1; in order to enable L1 to L2 interaction a special contract has to be set up on L2, where L1 state digest is updated. Using it the smart contract can verify bridge transactions on L2, which include the proof of transaction validity, which, in turn, includes the Merke Tree path of the account that makes the L1 transaction to L1 Merkle Tree state root hash.

The current L2 miner is tasked with submitting the corresponding L1 state to the bridge smart contract on L2, she is able to do it since she tracks L1 state, where the miner selection on L2 is taking place. It can also make the bridge smart contract invocation in the target chain with the corresponding transaction in the chain where the bridge transaction is initiated.

Based on the same mechanism cross L2 communication may be taking place. Based on L2 state digests posted to the mining smart contract on L1 a miner of a given L2 can post state digests of other L2 to it, thus enabling bridge transaction verification across different L2's.

Bridge transactions have to include a corresponding token ID on the target chain. The bridge SC on the target chain has to possess enough tokens to execute the transactions. Token ID correspondence between different chains has to be established once a new token is added to the smart contract and cannot be changed later. It requires a specific bridge SC method that registers the token ID correspondence.

5. New L2 Bootstrapping

Multiple L2 networks can be supported based on Waves L1. These networks utilize L1 as their consensus engine and can also interconnect through it. While they don't need to be fully synchronized within each epoch, they synchronize on the epoch level as the epoch succession is determined by the consensus on L1.

To bootstrap a new network, enough miners must support it. The network originator deploys a new network contract on L1, offering incentives to miners to support it. These incentives may come in the form of a fungible token with specific utility, such as paying transaction fees on L2. Each new network is assigned a unique address prefix.

Activating the new network requires a minimum stake set by the network originator. This stake should generally be at least comparable to the total L1 mining stake to ensure proper non-byzantine guarantees on L2. However, there might be situations where a lower stake is acceptable, but this may impose certain restrictions on liquidity transfers to and from such a network.

Node software will support both Waves and EVM networks, and there is potential to implement other smart contract engines as well.

6. Conclusions and future work

We have devised a strategy that enables us to efficiently reuse economic incentives and the comprehensive consensus mechanism of the primary L1 blockchain layer on associated L2 chains. This approach will initiate the launch of the inaugural EVM L2 on the Waves platform, followed by the introduction of several other L2 chains. Collectively, these chains will constitute an interconnected ecosystem, all rooted in the foundational Waves layer.

In pursuit of enhancing the proposed framework, the integration of zero-knowledge proofs stands as the logical next stride. These proofs, along with cryptographic digests of the present L2 state, can be cryptographically verified on the L1. This process supplants the necessity for the conventional longest branch selection, a prerequisite for confirming the accurate L2 state.

Furthermore, our plan entails an upgrade to the Waves L1 infrastructure to better accommodate interactions with L2. This enhancement will include optimization for swift verification of ZK proofs, thereby fostering a more seamless and efficient system.

References

1. https://www.researchgate.net/publication/335147247_Fair_Proof_of_Stake
2. https://github.com/OffchainLabs/nitro/blob/master/docs/Nitro-whitepaper.pdf
3. https://polygon.technology/papers/pol-whitepaper