Summary

This proposal is highly-opinionated – onchain token voting does not work. Low turnout, group-think, and bribery are unsolved issues with token voting, and the incentives are misaligned—even with a 'senate and delegates'-style representative democracy.

This quote sets the scene on chain voting fallacies.

What if I told you that the republic is now under the control of the dark lord of the Sith?

https://www.youtube.com/watch?v=OZtYhpzrcoM

Instead of on chain token voting, this proposal uses staking and social consensus to correctly align incentives and help facilitate network upgrades. In all other aspects it is identical to the Republic.

Shoutouts

Thanks to @LasseAztec and @brunny-eth for reading drafts of this. And Lasse again for his proposal, from which this borrows lots of ideas.

Background

The act of staking creates a simple consensus, in a loose sense. A group of people all have to come to consensus to stake on the same thing.

On Ethereum, this consensus is extended. Stakers also validate blocks and transactions. As a ZK L2, Aztec does not need to extend it's consensus to cover transaction validation, but it does need to be extended to facilitate upgrades and prevent censorship.

In a well designed staking system, the following truths apply:

1. Stakers are incentivised to maximise the Token's value, as their stake, and rewards are denominated in the native token.
2. Honest stakers will always elect to stake on the canonical version of the chain, where in their view, they will be able to maximise rewards, in return for work.
3. The version of the network that has the social consensus on an upgrade is the version the community will follow. This version will have the highest fee revenue and/or MEV and will enable stakers to maximise their rewards.
4. Assets bridged to an L2 from an L1 or from the real world must decide which consensus they will follow for an upgrade; Ethereum's staking consensus, the L2's consensus (e.g., Aztec), or their own governance system (offchain or otherwise).
5. If an asset follows Ethereum's social consensus, it will not move during a hard fork. A hard fork can't steal these L2 assets, as they are backed by Ethereum and the L2's state transitions are verified on Ethereum's consensus layer
6. If an asset follows Aztec's social consensus, it will move in a hard fork. A hard fork could elect to steal bridged assets.
7. If an asset decides to follow its own governance system it can be stolen by that governance system (Hello trusted bridge attacks…)
8. If an asset follows Ethereum's social consensus, it will not move during a hard fork. A hard fork can't steal these L2 assets, as they are backed by Ethereum and the L2's state transitions are verified on Ethereum's consensus layer
9. If an asset follows Aztec's social consensus, it will move in a hard fork. A hard fork could elect to steal bridged assets.
10. If an asset decides to follow its own governance system it can be stolen by that governance system (Hello trusted bridge attacks…)
11. If the total value staked inside Aztec (the value securing Aztec's consensus) is far smaller than the total value locked inside Aztec (the value of all assets in the Aztec ecosystem), staking attacks are possible.

Details

Setup

This proposal defines two smart contracts on L1; a token contract and an incentives contract.

The token contract is a standard ERC20 contract.

The incentives contract has the sole ability to mint / burn tokens from the token contract and provides a fucntion for stakers to add stake or exit their stake.

Both the incentives contract and the token contract are immutable and can't be upgraded.

Block Rewards

The incentives contract will pay out block rewards, by minting new tokens, to registered instance(s) of the Aztec rollup. The incentives contract defines an emission schedule of emissions per slot, defined as a rate.

This can occur once per slot.

A slot is defined in the incentives contract as a period of time, for which a block is expected. i.e 2 mins.

If there are multiple implementations of the rollup contract, i.e an upgrade has occured, the block reward will be payed pro-rata to the stake on any given instance for a given slot. For example:

% Block Reward = Stake On Rollup / Total Stake * Block Reward for Slot

% Uncle Block Reward = Stake On Rollup / Total Stake * Uncle Block Reward for Slot

The first canonical rollup implementation address is set in the incentive contracts constructor when it is deployed. This ensures that only the initial version of Aztec decided by the community via social consensus can perform a future upgrade and receives block rewards on day 1.

Post the one time setup, an upgrade is split into 4 phases:

1. Proposing an upgrade
2. Staker voting
3. Time Lock period
4. Deciding the canonical chain

Phase 1 - Upgrade Proposal

The canonical rollup (as defined by the incentives contract registry) is able to signal that the network wants to upgrade to a new instance of the rollup.

This new instance of the rollup must be deployed to L1 and social consensus used to convince stakers and builders to upgrade their implementations to support the new functionality.

How is an upgrade proposed?

In B52, the header of a proposed block is modified to contain an upgrade signal. The upgrade signal is the address of a new roll-up contract.

This maintainers of the open source software will update the code run by sequencer so that the software sets an upgrade signal for a new rollup address, above a given block height.

If Sequencers decide to update to the latest sequencer client, they agree with the upgrade and will start broadcasting the signal during block production.

In this model, the current rollup contract will interact with the incentives contract to distribute block rewards every time a block is marked as canonical. If an upgrade signal is set, this is broadcast to the incentives contract by calling the processRollup

method on the incentives contract.

The incentives contract will store the upgradeSignal and start tracking the number of blocks it receives with this signal.

Phase 2 - Staker voting

In B52, the block builder is dependent on votes from the proving network to get their block on chain.

The open source maintainers of the staking/proving client, will via social consensus update the implementation to vote for vote for a given upgrade, past a certain block height. An indiviudal prover can agree with this decison by updating to the latest client.

If the prover network votes on a block network with an upgrade signal, it is implicitly voting on an upgrade, without having to do an L1 transaction, and the maintainers of the software (core devs) are the ones deciding if the upgrade is good for the network.

The incentives contract will track the number of blocks the upgrade signal is present for. For an upgrade to be listed on the incentives contract, it must be present for 750 out of any 1000 blocks. (Needs modelling). i.e blocks can be skipped but in any 1000 blocks 750 must contain the upgrade signal.

Phase 3 - Time-lock

If a proposal passes the previous 750/1000 block window, it will be listed on the Incentives contract registry and marked as inactive.

The registry contract (like in the republic) will enforce a time-lock, enabling users who disagree with the upgrade and have portal assets that follow the registry, to exit.

Assets bridged via a portal contract rely on the portal contract to decide which instance of a rollup the assets are bound to in an upgrade.

As outlined by @LHerskind , portals can be:

1. Ungoverned, they are bound to one version of the rollup.
2. Governed by portal governance e.g makerdao
3. Follow a registry

Once the time-lock has expired, the new rollup instance can be activated by anybody.

Phase 4 - Canonical chain activation

After the time-lock, if activate is triggered on a rollup implementation, the incentives contract will mark it as the canonical chain, providing it has the highest stake compared to other rollup instances, and activate was called within a time period.

Activating a rollup implementation, will copy across previous rollup state and allow the new rollup to be eligible for block rewards.

IMPORTANT

. The registry must be append only, and not allow re-inserting of previous implementations as canonical, or double spends of portal assets are possible.

How is stake moved between rollups?

Outside of an upgrade, stakers can manually move their stake on L1 by interacting with the incentives contract. This is only possible for activated rollup implementations.

When a staker votes on a block that contains an upgradeSignal

, they commit their stake to move across to that rollup, when it is activated.

The rollup circuits are modified to achieve this as follows:

1. The merge rollup circuits in B52 already have to prove that the entity constructing the proof is a valid staker for a given rollup implemetation. This is achieved by passing in a staking tree root for the rollup, as defined by the incentives contract.
2. If the block header contains an upgrade signal, the prover must also prove they have correctly migrated their stake by marking it as pending for the new rollup. Stake is stored in an indexed merkle tree on the incentives contract. The leaf of the tree is:

Current rollup address: CURRENT_ROLLUP_ADDRESS Pending rollup address: UPGRADE_SIGNAL Public Key: STAKER_PUBLIC_KEY

1. Calling activate requires a zk-proof, which will prove a new stake tree root. The proof will re-write the stake tree move the pending rollup address to the current rollup address for each leaf that has the rollup being activated marked as pending.

Preventing Rapid Stake changes

Ethereum uses activation delays for new stakers to prevent an actor acquiring > 50% of the stake quickly. The incentives contract would have similar functionality to ensure that:

1. The incentives contract implements an entrance queue and an exit queue, to ensure that stake can't be dialed up and down quickly to exploit portal assets in a hardfork.
2. Only those already staking can vote on a new rollup.
3. Stake can't be added to a registered, but not yet active, rollup via the incentives contract.

Flow Diagram

See rough flow below:

[

image

735×655 49.8 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/626c0f1fc535f8cf063ecc8c41b6a60eeeef4f89.png)

Comparisons and Issues

Benefits

This proposal has some key benefits:

No L1 voting costs, voting is purely via changing software implementations.

Voters vote with their with their economic interest. This makes vote buying harder, as a bribe must overcome the networks staking rewards the staker would be foregoing.

Software implementors play a key role in the upgrade process, via social consensus.

Stakers are harmed by not participating in social consensus, as they are not maximising their staking rewards.

Comparisons

This proposal is very similar to the Republic, especially in it's stance on portals, flexible upgrades and a registry contract.

This proposal disagrees with the use of a senate / token based voting in the republic to facilitate such upgrades. The proposal replaces token based voting with stake based vote.

This is for a few reasons:

1. This proposal tries to follow the Ethereum model as governance as closely as possible, with differences only arising for bridged assets. This model is the most battle tested governance that exists.
2. The overhead of stakers "voting" is lower than that of on chain token votes, even with a senate. This is because voting occurs on L2, there is no additional gas cost to participate. Stakers or block builders just need to decided which implementation they will run.
3. Honest stakers are incentive aligned as they are trying to earn on their stake, from the network, so care about how future functionality of the network may effect this.
4. Stakers are more sophisticated than end users as the act of staking is interactive. This should mean they higher participation in governance. Liquid staking can be thought of as senate style system in this model. i.e if a staker uses a staking service, they are defering their vote to that service.

Issues

This proposal suffers from a bootstraping issue, where in the early days of the network, the stake will have low value, and upgrades could be grieved, or made maliciously. Forcing locked tokens from the foundation or the core team to stake could help alleviate this.

The staking contract and incentives contract are immutable, the only way they can be changed is a hardfork of the protocol. This means the logic of those contracts must be correct or a future upgrade will have to migrate the community to a new

incentives contract / token. As expertly explaind by @LasseAztec , the functionality offered by this proposal, is a superset of being non governed. i.e you can still hardfork the network at any point if you can convince the community to migrate.

[

upload_6a43874d587d9a245e088eb4139f1f11

1920×1722 102 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/095be9aa177f3b5fb88b8cd1249fc3667ad79b4f.jpeg)