Assumes the existance of an Aztec Token (AZT).

Summary

The Republic is a two-part idea I have been working on that presents a new spin on governance.

It’s all about giving the user more control of her risk-profile, whether she desires to delegate the responsibility to some governance or accept it herself.

The first part of ‘The Republic’ is about self-governed portals. We’re dividing them into three main types: rollup-governed

, portal-governed

, and ungovernable

. Each one with different levels of control and flexibility, so you can pick what works best for you.

The second part is about creating a more streamlined process for those who want

to follow forks without the hassle of migrations. We’re proposing the idea of a ‘Senate’ to embody this decision making system. The Senate is a smaller group, whose members are elected by the Citizens through token-governance.

But don’t worry if you’re not into that - portals can reject governance, and the system is flexible enough to replace the Senate with something else if desirable.

What makes this proposal interesting is how it integrates the self-governed portals with the Senate. It creates a flexible structure that lets portals choose whether to accept or reject forks individually. This makes governance an opt-in

process, giving you the freedom to choose.

Lastly, we’ve got some ideas for tackling the issue of ‘pending messages’ - those messages that are sent but not consumed before an upgrade. We’ve also got a plan for mass migrations for ‘ungovernable’ portals. We think these ideas could be useful for other proposals too.

Looking forward to hearing your thoughts and feedback!

[

1600×651 74.2 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/8fa9acac20cf8a2863276ca2f448a51ae3f494c0.jpeg)

Comparison

Currently no other proposals exits, however, consult “Closing Thoughts” to see it marked on the spectrum.

Generic Background

This section is generic background that also ended up being adopted for the RFP proposal itself. Added here for self-containment.

In the land of blockchain, it is sometimes useful to perform network upgrades to fix bugs or include new functionality, be it new opcodes, precompiles or changing the fee structure.

The “problem” of an upgrade can be split into two sub-problems:

1. Deciding that we want to upgrade and what we want to upgrade to
2. How we enforce the upgrade

In the following sections, we will be looking at how these sub-problems can be solved at a very high level for different types of systems; L1 blockchains and L2 (with bridged assets).

Difference on migrations and upgrading

We make a distinction between a “technical upgrade” and “technical migration”. A migration is a separate deployment,

where users must take action to migrate their positions from A

to B

. An upgrade instead requires no interaction for the individual user.

Example of migration: Uniswap

: With the launch of V3, people would migrate funds from V2 to V3 to supply liquidity.

Example of upgrade: Aave

: The pool contract is using a proxy and the implementation have been updated on occasion.

Before diving into the specifics, we have some notation to get out the way. Namely that we will differentiate between the words upgrade

and migrate

in a technical manner.

namely what is the technical difference between an upgrade

and a migration

. While the end goal for those are similar, “provide users access to a new version of the software”, it have some major differences it how this is acheived, and what kind of actions is required by the users of the software.

A migration is a separate version of the software, where foreign state must be “moved” from the old to the new version.

Ethereum

In the domain of Ethereum, the first part is often a mix of discussion by the community proposing EIP’s and client developers that work on implementing the proposed EIP’s. When a decision is made to include XYZ into a network upgrade, a time of upgrade is decided. Throughout this process any community member can raise issues with the proposed changes and you might see fractions in the community that disagree whether to support or reject the change.

When the time of the upgrade arrives any node in the network can decide if they want to run the new node software (including the change) or continue running the existing software. If there is a significant divide between the nodes this can spawn long-lived forks of the network with different rules. The value of the fork (and its claim to be the real

Ethereum) comes down to social consensus. A famous example of such a split spawned Ethereum Classic after the Dao Hack. Social consensus decided that classic was not the real

Ethereum, but both versions live on and users are able to decide for themselves what they think is the real

Ethereum.

Rollups with bridged assets

As Jon Charbonneau

nicely conveyed in his essays Rollups Are L1s (& L2s) a.k.a. How Rollups *Actually Actually Actually* Work and The Rollup Multiverse a rollup can use the same structure as Ethereum to figure out when to fork and what to fork to but there is one very large difference from a fully separate blockchain - the bridged assets

(foreign state).

If there is a proposed upgrade and community want to “fork off” they can deploy the rollup state transition contract, and start it from the current state. The assets can be used inside the fork as if they always were there, but the collateral for any bridged assets will still be in the same contract (not forked by this).

The bridge must decide which of the forks is the “real”

one, and only allow it to withdraw funds from the bridge. Only one of the forks can be seen as the “real”

one as it might be possible to double-spend otherwise. As an example, assume that you have 10 eth and the fork happens. On both of the forks you now have 10 eth, so you can from the L2 point of view exit those to L1. If L1 accepts messages from both forks you would be able to exit with 2*10 eth and practically have stolen someone else funds (double-spending your deposit).

If the bridge contract with the collateral is controlled by some governance that decided what it seen at the real fork it can impact what is seen as the “real” fork by anyone. In the case where a rollup has a canonical bridge which holds custody of a large part of the non-native assets, this bridge can practically decide what is seen as the real fork (or at least which of the forks that will have backed assets)

Above is a diagram showing how such a setup could look. We are showing message boxes as the means of communicating between L1 and L2.

Governance Spectrum

[

5904×1072 198 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/8d13af6e207cf0302b9c913174c5a44fc4c4f9bd.jpeg)

Ungovernable

If the contracts seen in the above diagram are all immutable without admin control or keys, the act of upgrading from one version to another requires that users are exiting funds from the rollup to enter in another system. This is sometimes referred to as a migration

as users need to migrate

the state from one system to another to employ the benefits of the new.

It is a “upgrade” mechanism that have been used in multiple projects in DeFi mostly prominent is probably Uniswap V3, which required liquidity providers to exit V2 and put their funds into V3.

Governed

Alternatively, contracts can be controlled by some governance organ which are able to decide what is seen as the “real” fork.

The exact mechanism for how this decision is enforced varies between different platforms, but can be done by having a proxy in front of every contract such that the implementations can be changed by govenrance if the decision is made. Graphically this would look like what we saw earlier, just with an extra party the “governance” that can update the implementations.

Upgrade delay and rage-quitting

To ensure that it is possible to rage-quit the system in case an undesired upgrade, it must be possible for the user to perform a forced-exit before the upgrade is executed. Otherwise malicious governance can trivially steal all bridged funds.

See Forcing a Transaction.

Tradeoffs

To outline some of the tradeoffs in a digestable manner, the below table shows what is required by the user to ACCEPT or REJECT upgrades and outlining what the user trust beyond correct implementation.

To summarise:

- Governed systems use delegates (direct or indirect) to perform decision making, making the workload smaller for users and thereby more convenient. While users and delegates are in agreement this method runs smoothly. However in case of disagreements, users need to act actively in order not to be forced to use the new upgrade.
- Imutable systems once again proves to be a double edge sword that protects users from automatically following upgrades - good or bad. Upgrades becomes an active choice for each individual, thereby avoiding coercion but also requires more work and research by users.

Details: Proposal Part A - How to upgrade

NOTE

Going to address it in opposite order.

First will outline how we can enforce upgrades,

Second outline how we decide on doing an upgrade

Design Goals

- Safety: It MUST be possible for an resourceful user to REJECT rollup upgrades and rage-quit with his funds.
- Safety: It MUST be possible for a prepared user to REJECT rollup upgrades without having to interact with the chain.
- Safe: It MUST be possible for a user to send messages L1<->L2 that can only be consumed by the specified version, even if an upgrade was performed
- UX: It MUST be possible for a user to ACCEPT upgrades without needing to migrate his state
- UX: It SHOULD be possible for a user to send messages L1<->L2 that can be consumed by the newest version, even if upgraded after message was sent

Social consensus light: By supporting the Safety goals, it is possibly for us to emulate a light version of social consensus, where users can REJECT updates if they desire and there is nothing a malicious upgrade can do to take their funds.

TL;DR:

- There is no canonical bridge.
- Portals are individually governed!
- Rollup governance is opt-in.
- Pending messages can be made to follow upgrade (opt-in)

Notation

Let \mathcal{R}_i

be the i

'th state transition contract. Let c

be the “current” index and \mathcal{R}_c

be the “current” rollup. Current being the one that is seen as “real” from the governance point of view.

Let N_i

denote indexed constant values, such thah N_1, N_2, \dots, N_n

are all constant values.

No Canonical Bridge - But self-governed portals

The Aztec Rollup will not

have a canonical rollup bridge that hold assets, but entirely rely on external developers using the message boxes to build asset containers (or bridges). This deviates from what most other platforms are doing at first glance, but is practially similar to a rollup where majority of collateral is held by bridges that are not controlled by the rollup governance.

We believe it is useful to seperate rollup-governance and bridge-governance (in order to combat centralization)

Since the bridges are contracts that are permissionlessly developed and deployed by anyone with the desire to do so, it is impossible for an “Aztec” governance to force its view upon the individual bridges. It might incentivise them, but cannot force it.

A portal can be fully immutable, or have its own governance mechanism that it used to decide which fork is real or it might fit somewhere between these options. Practically we could say that every bridge is self-governed

.

Portal Variants

Since the Aztec portal contracts are the holders of collateral the governance strictly don’t have full control of the assets that

have been bridged into the rollup. Their power is restricted by the implementation of the individual bridges.

As alluded to earlier, portals might be ungovernable

or governed

, but additional types exist. We distinguish between 3 primary types of implementations from the point of view of governance models, namely:

1. Rollup Governance

: This type of portal follows the whim of the Rollups governance. It relies on Rollup governance for safety.

1. Internal Governance

: This type of portal uses its own governance mechanism to decide which rollup to honor (what rollup is real). It can default to follow the Rollup Governance, but use its own governance to reject upgrades if desired. This portal has a lot of flexibility, but relies on 2 (possibly distinct) governance mechanism for safety.

1. Ungovernable

: This type of portal is immutable with fixed values for the current rollup. Assets in this type of portal must actively

be migrated from \mathcal{R}_i

to \mathcal{R}_{i+1}

by the users, see Appendix for initial idea on how this can be done. The design is trust-minimized and unfetched by the actions of the governance, but massively inconvenient for users that want to follow the upgrade as interaction (transactions on minimum L2) is required.

When thinking about upgrades by governance, one needs to consider that the portals individually decide what they believe to be the “real” fork. In the table below, we have the 2 extremes, but also middle cases, that have an internal governance and might rely on it for either all decision or only emergencies.

[

Model(72)

1920×1206 183 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/eb1a543f51c072f6bdd2e0540eaaec839b1d4f03.jpeg)

Note

Migration simply means that is requires action, but not necessarily individual exit and new deposit from every user. See the mass-migration. The case of exit and deposit is practically a simple migration.

That portals can employ any combination of these 4 makes upgrades a bit different from the norm. Also, it adds some constraint on us when we want to decide the upgrades as it makes handling “pending” messages a bit different.

What are we going to do?

To support our goals, most of the contracts that make up the Aztec Rollups L1 presence will be immutable without admin keys

. This includes the state transitioner and message boxes.

These contracts must be immutable, because an immutable portal can be tricked by cheating message boxes or state transitioner. By making them immutable governance cannot upgrade it to cheat these portals.

Note, that with state transitioner being immutable and not reading from external values, the implementations specify fully which sequencer selection mechanism is used and cannot be changed without a separate deployment of the state transitoner.

To facilitate an upgrade when most contracts are immutable, we propose to introduce a registry

which is controlled by the governance with the following tasks:

- Point to the governance implementation of choice.

- Point to the “real” implementation from the POV of the governance (state transitioner, message boxes)
- Store all prior “real” versions as snapshots (blocknumber of upgrade, state transitioner, message boxes) allowing archivers to easily figure out where to pull historic blocks from.
- Store a merkle root of a tree with the snapshots.

The registry is controlled by the governance, and they can insert a new state transitioners and swap message boxes as follows:

- Registry will run a migrate(address _newStateTransitioner)

function that * Reads the rollup state from \mathcal{R}_{c}

- Deploys new state transition contract (\mathcal{R}_{new}

) with the above state to initialise

- Adds \mathcal{R}_{new}

as the last “snapshot” of rollups (along with message boxes, which can be unchanged)

- Updates the current rollup in registry \mathcal{R}*c = \mathcal{R}*{new}
- Update a snapshot root computed on chain (merkle tree of snapshots)
- Reads the rollup state from \mathcal{R}_{c}
- Deploys new state transition contract (\mathcal{R}_{new}

) with the above state to initialise

- Adds \mathcal{R}_{new}

as the last “snapshot” of rollups (along with message boxes, which can be unchanged)

- Updates the current rollup in registry \mathcal{R}*c = \mathcal{R}*{new}
- Update a snapshot root computed on chain (merkle tree of snapshots)

The registry can only receive updates after long time-locks. The timelock ensures that users with assets in Governance Controlled Portals have sufficent time to performed a forced exit, see Forcing a Transaction.

Furthermore, the upgrade should not be possible to execute if there are pending forced transactions whose inclusion time is before the upgrade is at the earliest executable. This ensure that sequencers just halt halt block production until the upgrade can be executed to avoid escapes.

Hence the state transitioners are all immutable and without any access control, the old “rollups” \mathcal{R}_{i \neq c}

cannot be killed or paused. This allows users that have rejected the upgrade to continue using the previous implementation, assuming that there will still be someone to sequence it.

Due to the many-portal design, we can get a really interesting dynamic if assuming that the majority of portals are not simply following rollup governance. Namely, that rollup governance need buy-in from portal-governances and users in order to successfully “migrate” the collateral to the new rollup. If they cannot get this, due to contentious upgrade, users might decide that the “old” rollup is the real one and governance could be pushed into reverting their change and add the old rollup as the current one again. This might complicate expected state for users a bit as the upgraded state is practically discarded - The fork is pruned.

Note on sequencers

Only the “real” fork specified by the registry is able to mint a block reward, so sequencer selection implementations MUST be able to function without providing a minted token reward, as governance is otherwise able to “kill” existing implementations when the upgrade is happening. This can be supported by making the minting optimistic, and not revert if failing.

Sequencer selection implementations should not be constrained to only sequence for one instance.

There might be little incentives for sequencer to keep processing “old versions” if fees do not cover the cost as not additional AZT reward can be supplied.

It should be possible for “anyone” to act as the sequencer if the sequencer selection mechanism have been inactive for sufficiently long. Mainly applies to leader election proposals where leaders could be absent, and not B52 where anyone

might propose the blocks.

The diagram below gives an intuition of how the registry

is attached to the system outlined in earlier in generic background.

[

1501×804 123 KB

](https://europe1.discourse-cdn.com/business20/uploads/aztec/original/1X/91b206e24b5c303fac8d370a340bfe191c6c2420.png)

To support our goals for pending L1 and L2 messages, we have added changes in the appendix that allow us to both specify directly which state transitioner can consume the message from the Inbox, but also allows a user to send messages that can be consumed by upgrades through governance. The Latter being opt-in for safety reasons.

To fork or not to fork?

As we walked through above, upgrades are practically done through forking and convincing portal governance that they should use the new version, or for immutable portals that they should migrate to the new version.

This means that it is possible to reject upgrades by governance, WITHOUT foregoing the collateral on L1.

However, from earlier we have already seen the actions required for accepting or rejecting an upgrade proposed by the Governance under portal variations, so we will not dvelve too much in it here.

Nevertheless, governance is not the only that can play this game (even if they have the AZT incentives on their side). It is possible for anyone to fork the rollup by deploying their own variation. The main difference being that, no-one will by default be following the anon fork, they need to actively migrate to it. Note that this would fall into the “upgrade by governance” catagory if governance decide to follow the fork.

The specifics of migrations again depends on the type of fork, so lets draw a new version of our diagram, one showing how different portals handles upgrades to NON-governance appointed forks. Note that we do not have the REJECT option, as the action to REJECT a random contract deployment is no action.

With this at hand, we should have a decent idea around the portals and indirectly its users ability to control what version of the rollup is the “real” rollup.

Note that if rollup governed portals hold all but a negligable amount of collateral, the bargening power of other portals might insignificant, and you are back to the “standard” rollup case where the Rollup governance effectively controls the correct version of the rollup. Unless of course we have the special case where collateral on L1 is negligable as a whole, then we can socially decide what is the real rollup is!

Did we reach our goals?

- Safety: It MUST be possible for an resourceful user to REJECT rollup upgrades and rage-quit with his funds.
- Since we must forced transactions and upgrades can only be executed when timely exists have happened, it is possible for a user that follows the news to reject the upgrade and rage-quit with his funds.
- Since we must forced transactions and upgrades can only be executed when timely exists have happened, it is possible for a user that follows the news to reject the upgrade and rage-quit with his funds.
- Safety: It MUST be possible for a prepared user to REJECT rollup upgrades without having to interact with the chain.
- A prepared user can use ungovernable

portals instead of rollup governed which will require no action to reject the upgrade. Alternatively, a user with trust in “internal governed” portals can rely on a non-rollup governance to keep him safe.

- A prepared user can use ungovernable

portals instead of rollup governed which will require no action to reject the upgrade. Alternatively, a user with trust in “internal governed” portals can rely on a non-rollup governance to keep him safe.

- Safe: It MUST be possible for a user to send messages L1<->L2 that can only be consumed by the specified version, even if an upgrade was performed
- Through the use of addresses as the version (see Appendix), it is possible for the user to make his message consumable only by a specific instance of the rollup.

- Through the use of addresses as the version (see Appendix), it is possible for the user to make his message consumable only by a specific instance of the rollup.
- UX: It MUST be possible for a user to ACCEPT upgrades without needing to migrate his state
- A user that has funds in a governed portal will follow what the governance decides. If he choose the rollup governance or an aligned internal governance, he will follow along without any further interaction from his side.
- A user that has funds in a governed portal will follow what the governance decides. If he choose the rollup governance or an aligned internal governance, he will follow along without any further interaction from his side.
- UX: It SHOULD be possible for a user to send messages L1<->L2 that can be consumed by the newest version, even if upgraded after message was sent
- Through the special case of using type(uint160).max

as the version number the messages can be made to follow governance, supporting pending messages to move along with the upgrades.

- Through the special case of using type(uint160).max

as the version number the messages can be made to follow governance, supporting pending messages to move along with the upgrades.

Details: Proposal Part B - How To Decide

INFO

This decision can practically happen without impacting Part A much, and might even be changed through the upgrade process itself if the desire is so.

Who decides on upgrades

Token voting on every proposal scales horrible and users are often ill prepared for making decisions guarding the platforms.

As Part A introduced a way to govern the protocol without allowing governance to steal the funds of users that cannot fight back we believe that it can be beneficial to use a smaller Senate that strives to enforce the will of the people.

The Senate is used such that it is not required to have a full token vote for every upgrade where voters might be ill-equipped for taking a decision which in the end will centralise power anyway. We outline it explicitly hence it will often arrise anyways, and have the voting be in discrete chunks (elections) instead of continious delegation.

The Senate consist of members (senators) which are chosen by the Aztec token holders and it has the power to decide what “Rollup governance” believes to be the real

Rollup.

Actors

Actor-groups (a single actor might fit in multiple groups, e.g., holder and technical council):

- Token Holders (referred to as Aztecs or Citizens)
- Decide who the members of the senate should be
- Can force a re-election of the Senate
- Can declare “state of emergency” and “pause” the power of the Senate by doing a significant burn of Azt

. Inspired by Makerdao Emergency Shutdown.

- Decide who the members of the senate should be
- Can force a re-election of the Senate
- Can declare “state of emergency” and “pause” the power of the Senate by doing a significant burn of Azt

. Inspired by Makerdao Emergency Shutdown.

- Senators

- Voted in by the Aztecs (token holders) for a period
- Votes on AZIPs that upgrade contracts
- Voted in by the Aztecs (token holders) for a period
- Votes on AZIPs that upgrade contracts
- Technical Council
- Voted in by the Aztecs (token holders)
- To analyse feasibility of AZIPs and their impact on existing portals/contracts
- To analyse AZIPs from a security point of view and its impact on privacy of both new and existing users
- To publish their findings in the public domain as part of the review

phase of the AZIP.

- Voted in by the Aztecs (token holders)
- To analyse feasibility of AZIPs and their impact on existing portals/contracts
- To analyse AZIPs from a security point of view and its impact on privacy of both new and existing users
- To publish their findings in the public domain as part of the review

phase of the AZIP.

Upgrade Process from governance point of view

The flow of going from idea to new version of the rollup.

1. The community proposes an AZIP that require code changes to the Ethereum contracts
2. This could be changing the proof system, updating how the sequencer selection mechanism or other tasks that require changes to the core contracts.
3. This could be changing the proof system, updating how the sequencer selection mechanism or other tasks that require changes to the core contracts.
4. The senate gauges sentiment towards the idea, if there is strong sentiment against the proposal, drops the idea, otherwise continues to step 3
5. The technical council prepares a report/response and post it as part of the review

phase of the AZIP * The report should look into the feasibility of the proposal and its impact on the security of the system.

1. The report should look into the feasibility of the proposal and its impact on the security of the system.
2. Based on the report, sentiment might change, and the senate decides whether the proposal should be implemented
3. The implementation process itself can be performed by external or internal parties. If already provided, this step can be skipped along with 5.
4. The implementation process itself can be performed by external or internal parties. If already provided, this step can be skipped along with 5.
5. Implementation is done
6. After the implementation is done an new contract is deployed
7. Proposal that alters contract values can be put on chain with signates from \frac{1}{N_1}

of the Senate (agreeing to put it to the vote).

1. Proposal can be voted on by Senators for N_2

days

1. If m \ge \frac{1}{N_3} > \frac{1}{2}

Senators have voted for the proposal before N_4

days, it is enqueued, otherwise dropped.

1. After a delay N_5 > 14

days the proposal can be executed by anyone. After N_5

additional days the proposal is dropped if not executed yet.

Risks

- Long reaction time
- Lack of sequencers on forks
- Malicious Governance
- Governance Centralization
- Bribeable senate and vote buying
- Uneducated Citizens
- Low participation
- State of emergency abuse
- Governance Centralization
- Bribeable senate and vote buying
- Uneducated Citizens
- Low participation
- State of emergency abuse

Long reaction time

In the case of a vulnerability in the code, the reaction time for the governance to deploy a fix can be significant both due to the governance process, but also due to the additional delay to ensure that users have the ability to rage-quit. We think that this is a general issue of upgrades, and that any upgradable system that does not enforce rage-quits are insecure.

Lack of block producers on forks

Assuming that one of the proposals using some form of staking in the sequencer selection (block production) is used, the same group of actors that are producing blocks might be the majority of token holders that elected the senators pushing forward the divisive upgrade. This could lead to only few of the possible sequencers producing blocks on the fork.

This could lead to a case where users that intend to stay on “old” version when a upgrade happens might be unable to produce blocks with certain sequencer selection models.

For upgrades that want to fork off, these can specify a different sequencer selection algorithm and instead rely on bootstrapping a separate set similar to how other L1’s are forked.

Malicious Governance

Governance risks apply to both portal governance and rollup governance. For most portals, governance will likely have the full custody of funds, meaning that they are able to steal funds (similar to some DeFi protocols with proxies and governance). Because of this, we will mainly be interested in rollup governance for the remainder of this proposal.

If governance becomes malicious and want to perform an upgrade to a rollup where they are the only that can perform actions and simply take funds, how is the proposal addressing it? This could be because of greed, outside pressure or other reasons as we will look at shortly.

If a malicious upgrade is performed, portals that are blindly following governance (the aztec governed) are not to be saved. Of the internally governed portals, they either have to opt in to lose their funds or they can use the long delay before an upgrade to opt out. For ungovernable, no action have to be taken.

Governance Centralization

Senators

If the governance is centralized because of senators acting from greed or pressure, the citizens are able to throw the republic into a state of emergency or remove them from their duties to avoid an upgrade going through. This require coordination between citizens, but they are able to individually escape using the forced exit if necessary.

Token Distribution

If the centralization is due to token centralization where some entity have huge influence of the senate and can practically do whatever it wants the only real option to avoid the upgrade is to fork and rage-quit from the portals that follow governance blindly. In this case, there are no rollup governance with extended rights (special case in the message boxes) and all portal will need to be internally governed or setup a new governance structure and building on that.

Uneducated Citizen Or Senators, Low participation and bribes

Governance participation generally is very low for token holders, and it does not help us when proposals are hard to analyse completely. Some of this is hopefully addressed by only requiring citizens to participate in elections and state of emergency. However, chosing a senator that is unable to understand the effects of proposals (with the report by technical council) could be very detrimental to the system.

Buying votes as part of the election process or bribing senators can lead to similar results as above. The solutions are similar to centralisation due to other means - remove the senators or use portals that are not blindly following rollup governance.

Unable to upgrade due to “state of emergency”

If the value of the token burn required to enter state of emergency is low, of if there is a possibly payday from delaying an upgrade, it might be in the best interest of citizen to delay the upgrade by sacrificing a gift to the aztec gods.

There is no way to bypass the delay, so governance have to take the L and wait.