CCC Community-Actionable Thought #1

A Token Splitting Platform for Projecting Full CCC Vote Weight Across Many Chains and Launching Synergistic Projects Tightly Integrated with CCC

Disclaimer: None of the statements here are meant to promise that anyone will perform future efforts, nor are they meant to provide a warranty, nor are they meant to make a proposal where such a proposal would be illegal. Do not rely upon this document. Examples in this document are intentionally basic, and more advanced use cases are left as an exercise for the reader to foster discourse. Statements in this document are meant to exercise freedom of speech and community discussion of ideas.

Document Developed by Cat Church LLC.
ccc.meme, @Cccoinmeme, @Misolcom, @LandruCCC

Background:

The capability of a token to vote "on-chain" means that voters record their votes onto a blockchain where they are tallied. When the voting period is over, both users and contracts on the blockchain can read the results from the poll contract. When a transaction is executed that makes an "observer contract" observe the final results of a vote by querying the poll contract, the contract can choose to execute different outcomes based on the vote result. As one example, an NFT could have its URI change to one of two prepared URIs (e.g. one linking to a cat image, and one to a dog), that represent the winning vote option. Importantly, the observer contract can be developed in a manner where it is not possible to censor or prevent the actions of the observer contract from carrying out its response to the vote outcome. Although an external user must initiate the transaction that performs the observation, the contract can be written such that anyone can do this at any time after the vote is final. Thus, a special power of on-chain voting is that an observer contract can be programmed to perform anything that can be done on a blockchain, triggered by vote outcome.

Where token holders prove themselves especially reliable at making good voting decisions, they become a kind of decentralized oracle, where arbitrary questions or requests can be made to the voters, and their response can be increasingly relied upon. Where the reliability of a token's voting is a point of pride

for the community that holds the token, the community is incentivized to maximize the reasonableness and dependability of the vote outcomes. Such vote outcomes could be considered a decentralized oracle for the blockchain. Where humans are voting the tokens, the nature of the oracle is one that is endowed with the collective wisdom its holders.

In a future world where the tokens compete to serve as decentralized oracles for increasingly important decisions, the token whose holders demonstrate the most collective wisdom may win. Since delayed gratification, exemplified by the marshmallow test, has a high correlation with many dimensions of success (from health to wealth); and since such success is correlated with good decision making; a token whose holders are recruited from people who have passed a delayed gratification test might be expected to provide especially good voting reliability. Similarly, faith in the First Principles of Crypto, and the holding of fairly launched tokens, demonstrates a deeper understanding of the nature of truth and the wisdom to not be taken in by scams. For these reasons, CCC is a recruitment of the holders of XEN, eVMPX, and XONE for the purpose of maximizing an advantage in reliable vote outcomes.

At launch, on-chain voting tokens are only able to vote on the chain where they originate. To vote on another chain, the token may be bridged using a standard bridge, and the bridged token can be converted into a voting-token through a contract similar to the token contract on the original chain, with the difference being that the minting and burning of the voting-token on the alternative chain occurs whenever a user deposits or withdraws the bridge token into or out of the voting-token contract. In this way a voting token on one chain can be bridged and converted to enable voting on a different chain in a straightforward manner.

It is important to note that polls and votes use a timestamp at which point the vote weight is calculated for a given account. If a poll uses a vote weight as of noon January 1, and a user had no tokens at that time, then they will have zero vote weight for that poll even if they acquire tokens in response to seeing that poll and desiring more votes.

Problem #1: **One chain at a time**

A tokens' vote weight can only be counted on one chain at a time. For example, if a user holds their tokens on Ethereum January 1st, and then bridges all

their tokens to Fantom January 2nd, then that user will have no vote weight for an Ethereum poll that uses vote weight on January 3rd. The user would instead have vote weight available for a similar poll on Fantom.

Supposing that there are governance actions that support the health and ordinary function of the token on Ethereum that need to be done from time to time, a user would have to be sure to bridge back and forth, potentially prioritizing one vote over another where two polls on different chains use the same timestamp.

Solution #1: **All chains at the same time**

In order to enable full vote weight on multiple chains at once, the founding token (or "parent") must be "splitable" into multiple tokens each of which can be bridged to different blockchains. Users of such a token-splitting contract would execute the token splitter's "Split" function which would transfer the user-designated number of parent tokens to the splitter contract, and would emit that same number of child tokens each from multiple separate child token contracts. Each child token would have a different name and purpose. For example, fCCC could be the symbol of a token whose purpose is to be bridged to Fantom where it could be converted to a voting token on Fantom. Another child token, emitted to the user from that same Split transaction, could have a symbol pCCC, with the purpose of bridging to the Polygon network where it could be converted to a voting token on Polygon.

The original parent token (CCC) is no longer held by the user once it is split, but instead it is held by the token-splitting contract to enable future merges of the child token back to the parent token. To maintain the ability to vote on the founding chain, Ethereum in the case of CCC, the first child token emitted from the token-splitting contract could be eCCC, with the purpose of maintaining voting capabilities on that chain. eCCC could have inherent vote tracking functionalities, like the original CCC. To support users who do not want to split their tokens, poll contracts can be written that calculate vote weight as the sum of vote weights from two contracts, both CCC (or bridged and converted CCC in the case of blockchains like Fantom) and the child-token-based voting-token designated for that chain. The token-splitting contract prevents a user from holding both the original CCC (parent) and child token it was split into (if it was split) at the same time, so there is no risk of double-counting votes (however see a discussion below for issues arising from attempts to protect CCC from broken/hacked bridges).

The token-splitting contract is NOT one-way. A user can convert the child tokens back into the parent token. To convert back to X number of CCC tokens, a user must hold at least X number of each of the child tokens. The child tokens are burned during the conversion process back to CCC.

Problem #2: **Bridge Failure**

Bridges are not infallible and it is possible that they become defunct. Bridges are executed by off-chain resources that could cease to function. If such a thing were to happen, child tokens could be stranded on the blockchain to which they were bridged. When child tokens are stranded off of Ethereum, it becomes impossible to merge them back into CCC.

Solution #2: **Votes can excommunicate a child token**

Where the purpose of a token is voting, it makes sense to use that voting power to control the token-splitting contract itself to mitigate risk. This is especially true when the risk is off-chain and human understanding and evaluation of the issues is needed.

The token-splitting contract can include a polling system that asks users if any child tokens are stranded due to defunct bridges, or if any other issues have caused it to become unreasonable to expect users to provide a particular child token in the merging process back to CCC. If such a vote passes for a particular child token, the merge function of the token-splitting contract can "excommunicate" it, which would remove the requirement that the user provide a particular child token to complete the merge.

Note that if the child tokens are being valued by some users by some criteria, such as their use in merging back to CCC, and a vote passes that removes that use-case from a particular child token, then those users may see the value of the child token differently. Therefore, it should be considered highly speculative to rely on perceived values of the child tokens since the excommunication votes could possibly affect that perception.

If a bridge were voted as defunct or a child token were excommunicated for any reason, but the bridge was to operate again in the future, then it's possible that

the total available vote weight for votes on the chain corresponding to that child token (which are capable of using the combined bridged-CCC vote weight, and child-token vote weight, as previously described) have more than a total of 1 billion vote weight for a given timestamp. The maximum vote weight of 1 billion is only guaranteed by the non-coexistence of a child token and the CCC that created it. Therefore, contracts that sum the vote weight of both CCC and a child token could see total vote weight be as much as 2 billion if the child token is excommunicated (since the existence of the child token is no longer exclusive with the parent token). An adjustment period during which users naturally migrate to using poll contracts that utilize CCC vote weight exclusively on that chain, or the child token vote weight exclusively on that chain, might be expected, during any interim time where a new child token for that chain is not yet designated.

Problem #3: **Competing use-cases and new chains**

Projects may want to tightly integrate with CCC in a similar manner to a bridge token, to enable new use-cases for CCC. New chains may come into existence that are not out yet, and a way to name a child token after its purpose and target chain (for bridging) is desired.

Solution #3: **Voting for naming and releasing child tokens**

The token-splitting contract can emit all of its child tokens to the user during a split, but only allow the tokens to be transferable once the name of the child token has been approved by voters. This helps unify the purpose of each child token to avoid conflicting uses (e.g. where a token is used for voting on two different chains because one chain did not get a child token designated to it). Since future votes can control naming a child token, and thereby suggest a unified understanding of which blockchain a child is meant to bridge to, the names and purposes of all the child tokens do not need to be known when the token-splitting contract is deployed.

Child tokens can be approved that represent use with a particular project. As one example, a staking contract could be created that yields a separate token at a decaying rate, such that ownership of that token represents in some sense being an early adopter of the project (since early adopters will experience higher rates on their stakes). To work with CCC, a project could petition voters to vote to approve a child token for naming and release, called “Super Cat Token”, which can then be

staked into that project's staking contract to yield "Super Kitty Tokens." Notably, if that project's staking contract were to become defunct (e.g. not releasing the Super Cat Tokens when a user tries to unstake), then the token-splitting contract's excommunication vote can allow users who staked their Super Cat Tokens to get back their CCC if they have all the other non-excommunicated child tokens.

One can imagine a scenario where a single vote to excommunicate a token without reasonable cause could be seen by the entire crypto community as discrediting the reliability of CCC votes. Because of the on-chain nature of CCC votes, it should be clear to the reader that the consequences of CCC votes can be quite serious, with the potential to both build and injure the reputation of CCC.

The token-splitting contact may emit tokens to the user during the split function, although they are soul-bound (non-transferable) before a vote succeeds that names and releases them. This may enable the splitting and merging process to be simplified so that merging and splitting doesn't need to know whether a particular child token has been released or not. To reduce gas costs for merging, the user may be required to set a certain flag in the merge function to indicate that a particular child token has been excommunicated. If the flag is not set, then the user's child tokens will be used to merge even if the token was excommunicated. Off-chain transaction-preparation should account for this and check whether to set the flag (i.e. the transaction parameters are different if merging is being done that is expected to not burn excommunicated child tokens).

Problem #4: **How to tell if a vote to name a new child token, or excommunicate an existing child token, is valid?**

If polls that excommunicate or release new tokens are allowed at arbitrary rates then they may receive too few votes to be reasonable. When algorithms are involved to determine whether a poll has sufficient votes, the validity lies in a grey area. Clear rules are needed.

Solution #4: **Voter-controlled parameters for future votes.**

A vote comprises multiple phases and multiple voting fields. The fields include:

1) Next-vote-months-delay (0-13 months, each "month" is 4 weeks).
2) Approve or disapprove of excommunicating any token.
3) Which token to excommunicate if an excommunication is passed.

4) Approve or disapprove of releasing a new child token.
5) Name of the new child token if the vote to release a new child token passes.
6) Symbol of the new child token if the vote to release a new child token passes[1].

The phases of the vote are:

1) Voters create and vote for write-in options. This phase might be 3 days.
2) Voters vote for a name and symbol written-in during phase #1. This phase might last 3 weeks.
3) If at least 50% of all votes are against releasing a new child token, the vote is over. Otherwise, the new child token is approved with the winning token name assigned, and a 4-day voting period is available for people that voted for a different child token name and would like to participate in selecting the symbol of the new token. If the vote to excommunicate is passed (greater than 50% of votes in favor), the child token that receives the most votes to be excommunicated becomes excommunicable in phase 4.
4) If a new token was approved, the winning token symbol is assigned.

Each vote contains a vote of whether to approve a new token at all. A new token will only be released if greater than 50% of all votes approve the token. A voter can vote to not approve a new token while still voting for a vote name and symbol in case the new token is approved anyway.

Each vote contains a field for whether to excommunicate a child token, and if so, which child token should be excommunicated.

Votes can use both CCC and eCCC (the first child token, for Ethereum voting) for vote weight so that users who have split their CCC can vote without having to merge back to CCC.

A transaction for step 4, executable by anyone, is necessary to perform the naming of the token and enablement of transferability of the new child token. (Prior to naming and approval, the child token was not transferable). This transaction also

[1] Where a voter votes for a token name and symbol, and a new child token is approved with a different name, then the voter will have a short period to adjust their symbol vote in the context of the chosen token name if they choose to.

performs the excommunication, if it was approved, and the calculation of the timestamp that begins the next phase 1 vote start (see below).

Votes include a field representing a number of months (1 month = 4 weeks for simplicity) to wait until the next vote begins. Each voter specifies an integer between 0 and 13. If 0 wins, the next vote (write-in phase) begins immediately. One method of choosing how may months to wait before the next vote starts is to use the median value of the months vote[2].

Plurality voting (selecting the winner as the one that gets the most votes) is not the only voting scheme. Approval and score voting are reasonable alternative vote methods for the blockchain but, in the context of on-chain voting, both give favor to voters willing to pay more gas. Approval voting lets voters vote for more than one candidate that the voter "approves" of. Score voting is similar except that a rating from e.g. 1-10 can be given for each vote so that not all "approvals" are of the same weight. Both methods are compatible with CCC vote weight tracking. Both methods prevent spoilers that split the vote for a particular option, and both methods can help select a common middle ground. One issue is that gas costs go up for on-chain voting when voters vote for any more than one option as is possible in both approval and score voting. One option is to present a poll prior to deployment of the token-splitting contract where the community is asked whether approval, or score voting should be considered. Other voting methods tend to be prohibitively complex for on-chain voting results calculation (unless many transactions are used to resolve a vote, and gas costs for voters are increased, which is undesirable).

The community is on notice that, if a member of the community would like to propose and lobby for a method other than plurality voting, a poll should be created that voices this so the results of the community's preference can be seen.

Problem #5: **Running out of child tokens to name and release.**

It's possible after the project has been running for a long time, that the token-splitting contract cannot split all the child tokens in a single transaction. A user might want to split their CCC only into the child tokens they intend to use, if the number of child tokens is too numerous.

---

[2] Using the median simulates an instant-runoff vote, and works well on the blockchain when the number of integer vote options is small, such as the14 options in this case.

Solution #5: **Nested token-splitting contracts**

The first 20 child tokens of the token-splitting contract may be normal child tokens for use in bridging or projects. The next 20 might each be parent tokens used by separate token splitting contracts. The ability to deploy trusted token-splitting contracts means that a token-splitting contract factory may be required. Special deployment transactions may be required to unlock new token-splitting contracts when a new child token is released, which any user can execute.

The top-level token-splitting contract needs approval from the user to transfer CCC to itself during the merge transaction. Similarly, each sub-level token-splitting contract would ordinarily need approval to split its parent token (that parent token is a child token of the token-splitting contract just above it). The nested token-splitting contracts create a tree, where the leaves are child token contracts. Pre-approvals can be integrated into the relationships between the token-splitting contracts to facilitate splitting and merging without requiring manual approvals for each link in the tree.

Conclusion

By nesting token-splitting contracts, the number of bridge and project tokens becomes unlimited. Each represents a shard of CCC which can be combined back into the whole. CCC voters can vote with full vote weight across many blockchains. The core CCC token is protected against permanent bridge outages through excommunication. Project tokens that have their own child token approved, through voting, do not compete with CCC since they are literally a piece of CCC.