Goal:

This post is an idea for how to handle forking in Schelling point based systems. One should compare it to the forking system of Augur (see Section 9 of their whitepaper); however, this proposal will be adapted to a generality/context that is not present in Augur.

Specifically, we still have the general Schelling game framework where one is rewarded for voting coherently with the majority and penalized for voting incoherently, with the idea that if we ask questions about the real world, the "truth" will often be a Schelling point. However, we attempt to adapt to the situation when the questions we ask might be hard. Namely, we expect 1) reviewing a given case to take a non-trivial amount of effort, and 2) honest parties will arrive at the "wrong answer" - i.e. an answer other than that which would ultimately selected by an honest majority - some non-trivial percentage of the time. (This idea was conceived for use in Kleros, which attempts to be a fairly general dispute resolution platform. In real world disputes there will be situations where honest people will disagree sometimes. Compared to a Schelling point based oracle that is used to handle more objective questions, honest participants in a Kleros-type system are likely to have more variability in their returns, but that should be okay as long this averages over the long term. As it can take a lot of effort to review a case, we have a generally small, discrete number of token holders who are drawn to vote on each case, with the possibility of appeals, to minimize duplication of effort.) Recall that block producing can be thought of as a sort of Schelling game where people are voting on rival blocks/the order of transactions, so there may conceivably be some applicability of these ideas to forking in consensus algorithms, but I haven't thought much about that angle.

tl;dr:

When considering which fork they would want to wind up on, token holders may consider both the outcomes of the case associated with the two sides of the fork but also the percentage of the community that goes with each fork. To find the largest fork that is compatible with users' cutoffs for not being willing to "go it alone", one can use a sort of auction system.

Context:

The fact that cases can require a significant amount of effort to evaluate probably implies that a system such as the one Vitalik discusses here is not viable in our setting. The idea is that every decision results in a fork and a DEX could be used to determine which fork has the highest market value, taking that as the "true value" for the purposes of a smart contract using the result. While such a system shouldn't require participants to reevaluate every

case because usually only one outcome of any given case will give forks that have non-negligible value after a certain time, this system would probably nevertheless require all active market participants to evaluate most of the recent

cases, which will likely already be a significant duplication of effort.

We want a forking mechanism that both:

1. facilitates an honest minority forking out an attacker's holdings in situations where there is a 51% attack (or indeed one might also want a fork in case of some kind of ideological split such as the literalist/spirit of the rules kind of split that one might have seen with the Augur "who will control the US house after the midterms"; market if it had gone on long enough to force a fork).

2. while still allowing for the possibility of cases where, potentially after a long process of appeals, there is not a consensus among honest parties on which side should win. Then, a ruling should be made one way or the other, but if token holders recognize that there is not a need to split the community over this case, then such a split shouldn't occur involuntarily. (An important note here is that we image that, absent eventual forks, only a percentage of a given voter's tokens would generally be staked on the outcome of any given case. So, if there is no fork, a voter that is incoherent with a given case would still typically have tokens/be "part of the community".)

We imagine that a given token holder's utility for a given case with a potential fork as a function: utility=fct(case outcome, percentage of tokens that are on the same fork as me after the case).

This allows for a possible trade-off between how egregiously incorrect/unacceptable the winning answer is and the breakdown of how the community splits. We can imagine cases where someone would think that outcome A is pretty injust, and it would be worth forking to a universe where outcome B won, but only if a large percentage of the community forked with her. Otherwise, if only some marginal amount of the community would have been willing to fork over this case, she prefers tolerating outcome A and remaining in the main branch.

(This possibly abstracts both the price they might expect the tokens to get on markets going forward after the case as well as their morality/altruism and willingness to participate in a system that they view as just or injust.)

We expect that the utility function should be monotonic in the percentage of tokens going to the same fork as you. I.e. all else being equal, we assume that participants would not prefer that the fork they are going to be smaller, as this would be a sign that it would be less likely to catch on. These dynamics are reminiscent of the "battle of the sexes" coordination problem in game theory, where two parties try to coordinate on two possible outcomes, and while they have different preferred outcomes, their preference for landing on the same outcome as the other party is stronger than their preference for their better outcome.

Proposal:

User USR_i

submits (vote_i

, r_i

)\in\left{0,1\right}\times [0,50]

. The user's choice of r_i

will essentially allow them to specify a minimum threshold for community support for a fork at which USR_i

would want to join the fork.

The "main fork"; is the one where the outcome_{main}

corresponds to the choice receiving the most token-weighted votes. The main fork is the fork that is used to settle any payments of ETH in existing contracts. For the moment, we only consider cases with binary outcomes. Then the "alternative fork" is one where the other outcome is adopted.

We want to organize voters, weighted by tokens, into the two forks such that

1. USR_i

remains on the main fork if vote_i

= outcome_{main}

1. USR_i

remains on the main fork if vote_i \neq outcome_{main}

, but the total percentage of tokens that go to the alternative fork is less than r_i

1. USR_i

goes to the alternative fork if vote_i \neq outcome_{main}

, and the total percentage of tokens that go to the alternative fork is greater than r_i

In Kleros, the users who would normally cast a vote are limited to those who were drawn to consider the case, which initially isn't very many people but grows with appeals. We could have a special forking vote as some sort of ultimate round like Augur does if a case gets appealed enough times. Alternatively, one could always give users the ability to submit a preference including an r_i

like this (even if they aren't drawn and their vote doesn't contribute to the outcome of the case in that appeal round), with the expectation that in early appeal rounds, the total percentage of tokens willing to fork will be small because only users who were drawn to vote will be likely to be paying attention to this case.

Note that which fork a voter is included on influences the percentage of tokens on each fork and hence the constraints on r_i

for the other voters. Hence, there are potentially multiple possible consistent assignments of the voters to the two forks. Indeed, assuming r_i>0

for all USR_i

, then the choice of putting everyone on the main fork satisfies these conditions. We would like to choose the assignment of voters that maximizes the percentage of tokens on the alternate fork.

The process of finding the maximum percentage of tokens that go to the alternative fork can be done efficiently.

Specifically,

1. Take the list L

of voters who voted for an vote_i \neq outcome_{main}

.

1. Sort L

by the user’s r_i

.

1. Take the sum of the number of tokens over all voters on L

. Denote this by r

.

1. Take the user USR_i

with the largest r_i

. Compare r_i

to r

. If r_i<r

, remove USR_i

from L

.

Repeat steps 3)-4) until L

stabilizes.

Then, the token holders remaining on L

go the alternative fork. A simple inductive argument shows that none of the voters removed in step 4) could have gone to the alternative fork under any subdivision. On the other hand, all voters that remain on L

after this process have r_i\leq r

; hence if they all go to the alternative fork together this is compatible with their choices.

Sorting L

takes O(NlogN)

time, where N

is the number of voters who vote for vote_i \neq outcome_{main}

. The repeated process of 3)-4) takes O(N)

time, so the entire process takes O(NlogN)

time.

Conclusion:

One wants to give maximum freedom to participants to fork as last defense against attacks and then facilitate the organization of the resulting groups. This idea attempts to balance that against not causing the community to split when splitting isn’t really the will of the community.