Proof of Participation Voting for On-Chain Governance

Terry Chung, Sandip Nair, Uttara Ravi, Pranav Kajgaonkar

December 2021

Abstract

Governance is the process of maintenance and implementation of changes to a system. Within the context of on-chain applications, this is the process by which different stakeholders come to an agreement on what changes to implement and the process of implementing those changes. Of particular importance in designing governance systems is taking into account its susceptibility to Sybil attacks (the use of multiple network identities to gain more voting power) and plutocracy (where those with more financial resources have significantly more voting power as in most coin-voting systems). In this piece, we propose a proof-of-participation based voting mechanism, with a particular use case in video games that is both Sybil resistant and anti-plutocratic.

Introduction

It is imperative for a decentralized protocol to have a way to upgrade, add features, change itself in ways that don't succumb to central authorithy - that being after all, its core value proposition. Ideally, one would want such a protocol to be able to take into account voices of its core users, core stake holders, without any value extractable via collusion in governance by large players who have interests outside of the development and advancement of the system itself. Though this itself is a hard concept to define, we generally assume that if the heaviest users and those with the most skin in the game have the means to shape the protocol that will result in the best outcome. Having said that, this may not always be the case.

Regardless of the obvious assertion that a decentralized system should not be governed by a central entity by definition (though recognizing that centralization comes in varying degrees), we also believe that decentralized governance may yield better results for certain protocols.

A centralized system is maintained by a centralized authority. In a protocol, this could mean that a core developer group belonging to a company has the ability to unilaterally alter the protocol. Not only does this require trust in the developers to implement what is in the users' best interests (and not the dev teams') it also introduces a centralized point of failure. Those who lobby and bribe have to expend less resources as they only have to buyout and convince a small number of people and the protocol is more prone to corruption. Decentralized governance is not only more anti-fragile but also necessary for truly decentralized protocols.

Motivation

1. Problems with existing decentralized governance

Realizing problems in centralized governance models, especially in environments where the team is anonymous, in different legal jurisdictions etc, the decentralized applications (dapps) of today have implemented different forms of governance.

One of the most widely used systems today by most dapps is coin-voting, simple and quadratic. In this scheme, the number of votes that you get is proportional in an increasing function to the amount of tokens in the project that you hold. These, as notably documented by Vitalik Buteirn in "Moving Beyond Coin Voting" [2] suffer from plutocracy, vote buying and coordination issues. They empower coin voters at the expense of other members of the protocol community who either use or depend on the protocol more so than the large financial interests (notably VCs and angels) who purchased the token early at a large discount [1].

Yet because it is hard to conduct Sybil resistant voting without coin-voting, because proof of humanity schemes infringe on privacy and proof of participation validation in current protocols is extremely hard, coin voting is the defacto governance model of most dapps today.

An example of the pitfalls of coin-voting might be Justin Sun's takeover of Steem.[2]

2. Existing Proof of Participation Governance Mechanisms

Today, a notable example of off-chain proof of participation schemes are in multiplayer video-games. In Old-School Runescape (OSRS) players are able to vote on updates to the game, and the developers (by social contract) only implement updates that pass with a plurality (75%+) of the vote[5]. In the popular mobile game Pokemon Go, the right to suggest the creation of new 'PokeStops' is only given to players above Level 38, which requires a significant investment of time into the game [3]. All these governance models reward players who have played the game and participated in the ecosystem, albeit at a surface level. Governance models based on proof of participation are able to create protocols governed by their most avid users, irrespective of their financial status, in a fair and verifiable way. One issue in both cases is that of bots (defined below).

3. Emergence of new dapp protocols that allow verifiable proof of participation

With the emergence of crypto-games, levels, which can act as a proxy to active participation in the game-protocol, is information available on chain, especially in the case where they are stored in an NFT or associated with your wallet address. We now have access to a tamper-proof metric of "proof of participation". Using this information we can create new governance models that make different sets of trade-offs in sybil-resistance and plutocracy and that may solve existing issues with coin-voting. We can further extend this to other dapps that have such metrics available. We will explore a construction and cost-of-attack for a simple example of such model.

Outline

Our model for proof of participation governance determines the voting power $V_i(x_i)$ for a user i based on the user's *level* in a decentralized crypto-game. In this game, a player's level is stored on a blockchain, in either a mapping from address to level, or on an NFT that represents their character as a state variable of the NFT. Therefore any updtae to player information such as an increase in the level (which we verify through an assumption specified later) via submitting a transaction to the blockchain network, costs a certain amount of gas and transaction fees. This is the first mechanism we use to help counter botting attacks, where an attacker could potentially employ a large number of bots to take over a majority of the voting power in the system. This is because the cost of such a botting attack now scales with the number of bots, and depends on the gas fee for making a level update (which could be arbitrarily set by adding arbitrary operations in the function to level up accordingly so that it discourages bots, without making it very expensive for honest members of the network to level up).

The second mechanism we use to counter bots is via scaling $V(x)$ with x. We present analysis of a particularly chosen $V(x)$ but each game designer can choose this function dependent on the desired and observed properties of their system. As we assume bots are caught at a constant probabilistic rate, it is significantly more likely that real players who have played for a long time and hence have achieved higher levels x have more voting power, and can thus out-vote bots.

Finally we counter plutocracy via proof of participation, since the voting power of a player depends on the effort put into leveling up in a game (via actively playing the game), not just on the cost of leveling up (gas fees for level update transactions). Depending on sufficient conditions to level up within the game (which can be made complex enough to make it difficult to bot), and the efficacy of the bot-detection system, it is possible to achieve a state where the only reliable method of accumulating voting power in the game is to level up by actively participating in the network.

In this piece, we first start by introducing and defining terminologies from gaming that we will be using. We then propose a voting model based on player levels in a general form where the specifics of the level function $x(t)$ and the player-level distribution can be replaced with the observed and/or decided parameters of a specific game. We then proceed to analyze a specific instance of a level function and a player-level distribution which closely models real-life video games. We finally show the cost of obtaining a 51% voting power via a specific form of attack on this system as a function of different parameters.

Terminologies borrowed from video games

We define certain terminologies borrowed from video games below :

1. **Role Playing Game (RPG):** A video game in which the player is in control of one or more characters (in-game controllable units). The plot and mechanics of the game revolve around the growth and development of characters.

2. **Massively Multiplayer Online RPG (MMORPG or MMO):** An online RPG with a large number of players where they may interact with each other in-game.

3. **Experience:** Measure of the amount of effort that a player has put into the game. This may be for a specific game character or for their game account as a whole.

4. **Level:** Experience thresholds which determine how far ahead a player is in the game. For example, a character's level in a role-playing game (RPG) determines its strength and may also unlock several in-game features.

5. **Farming:** Repetitive tasks such as collecting materials or gaining experience by defeating in-game enemies or completing tasks/quests. This is one of the biggest motivations for botting in games.

6. **Botting:** Deploying some digital agents to farm on several game accounts to reap some benefit, such as milestone rewards achieved for reaching certain levels, etc without spending time to play the game yourself.

Model: Proof of Participation voting

Assumptions

1. The existence of a public blockchain where transactions have an associated gas cost.
2. The existence of a decentralized game where the game can be played locally on your own machine but account information such as character level is stored on a blockchain, and requires transactions to be submitted to update. We also assume that it is impossible to tamper with the level up process (i.e to level up a player must submit adequate proof that they have completed the pre-requisites to level up, and the player cannot forge or artificially accelerate the obtainment of this proof). This is not an issue we will remark on, but we believe games on rollups can perform this task without consuming too much gas.
3. The existence of some form of decentralized (or initially centralized) bot-detection mechanism that is available for this game (albeit probabilistic), to make bots distinguishable from honest players.
4. The power of botting lies in numbers, not in the time spent in game. This is because the more time an attacker's bots spend in game, the higher is the probability of them getting caught.
5. In the canonical attack vector we examine, we assume that the attacker at a time t releases all of his/her bots into the network (that is, they start the game at the exact same time). This is a reasonable assumption since the attacker would want to maximize the time their bots spend farming experience in the game. At the moment we don't consider other strategies (playing for a while and intermittently botting) though we recognize those may be much more cost optimal for an attacker that bots.

Definitions

Consider a decentralized game as defined earlier and a player P, who has spent time t farming experience in the game. Additionally, assume a global clock such that the current player distribution varies with time.

1. **Level function** $X(t)$**:** The current level of a player who has spent t time farming for experience in game. We assume that all players are playing at the optimal frontier. In practice $X(t)$ will vary for all players even for the same time spent playing t.

2. **Bot detection probability** b**:** Probability of detecting a bot at a given timestep.

3. **Start threshold level** $s(b)$**:** The minimum level required to participate in voting, such that the voting power below this level for a player is zero. Intuitively, b is a parameter that depends on the level of centralization in bot banning the protocol and community are comfortable with. This is decided by prevailing level of technological advancement in securing certainty that a player is a bot based on their actions.

4. **Voting power function** $V(t,b) = V(X(t), s(b))$**:** The voting power of a player at level $X(t)$ given a starting threshold $s(b)$.

5. **Player level distribution** $n(x)$**:** The number of players in the game who are at level x at a fixed time t. This function presents a snapshot of the distribution of players by level.

6. **Max-level** M**:** The maximum player level at fixed timestep t. Alternatively this can be defined by the protocol. The choice does not make a difference in the calculations.

Voting power as a function of player level

$$V(t,b) = \begin{cases} 0, \ X(t) \leq s(b) \\ \gamma\left(X(t) - s(b)\right), \ \text{otherwise} \end{cases}$$

where γ is a constant of proportionality (could be taken to be 1, for instance).
This is the general voting mechanism, where a player's voting power can be calculated based on their on-chain level (or recorded in an NFT that represents their account). Next, we proceed to analyze a specific instantiation of this model which we believe closely resembles real world games.

Analysis

We model the fact that in video games it is progressively more difficult to reach higher levels by the following level function:

$$X(t) = \lfloor \alpha \log(t+1) \rfloor$$

where α is a constant of proportionality, which can be 1 for instance. This choice is justified by the fact that the logarithmic function's rate of growth decreases as its input increases. Notice that at $t = 0$, $X(0) = 0$. But note that this is an arbitrarily designation and game designers should plug in their own level function here.
If we can detect bots more frequently, then we do not need to set s too high, so $s \propto 1/b$. Thus, we define

$$s(b) = \lfloor \beta \left(\frac{1}{b} - 1 \right) \rfloor$$

where β is a constant of proportionality. Notice that $s(1) = 0$, since in that case bots will always be detected and there will be no bots in the system. Also, taking the floor is a design choice, since we want integral levels.

Let every increase in level require an update transaction which costs g gas. Then the *cost* in gas to reach a specific level l starting from level 0 would be gl. Note that the game designer can include arbitrary calculations in the level up process to raise g. Therefore the upper bound of g is a design decision. We now calculate the number of timesteps to be spent farming in game that is required to reach level l. We have

$$\begin{aligned} l = X(t) &= \lfloor \alpha \log(t+1) \rfloor \\ \implies l &\le \alpha \log(t+1) \le l+1 \\ \implies e^{\frac{l}{\alpha}} &\le t+1 \le e^{\frac{l+1}{\alpha}} \\ \implies e^{\frac{l}{\alpha}} - 1 &\le t \le e^{\frac{l+1}{\alpha}} - 1 \end{aligned}$$

Hence, the time spent by a player P starting at level 0 to reach a certain level l by farming is

$$t \in \left[e^{\frac{l}{\alpha}} - 1, e^{\frac{l+1}{\alpha}} - 1 \right)$$

Next, we instantiate our player-level distribution. Again, it is based on the assumption that leveling up gets progressively harder. In practice this will be a function derived from observed states of the actual game and dynamically adjusted with time. Let

$$n(x) = ae^{-kx}$$

where k is a function of T which determines the distribution at global time T. Let us determine the value of the term a. We can do this by the fact that the total number of players at a given time T is N. We have

$$\begin{aligned}
\sum_{x=0}^{M} n(x) &= N \\
\implies \sum_{x=0}^{M} ae^{-kx} &= N \\
\implies a\frac{e^{-k(M+1)}}{e^{-k}-1} &= N \\
\implies a &= N\left(\frac{1-e^{-k}}{1-e^{-k(M+1)}}\right)
\end{aligned}$$

Thus, we have

$$n(x) = N\left(\frac{1-e^{-k}}{1-e^{-k(M+1)}}\right)e^{-kx}$$

This is the number of players with level x at time T, which is determined by $k(T)$.
Let $\mathcal{V}(x)$ be the cumulative voting power of all players at level x at global time T. Then

$$\begin{aligned}
\mathcal{V}(x) &= n(x) \times V(x) \\
&= n(x) \times \gamma(x-s)
\end{aligned}$$

where $V(x) = \gamma(x-s)$ is the voting power of one player at level x.

Attack model

We carry out a probabilistic analysis of a naive and simple attack (for botters), which is based on the assumption that botting is easy (it is inexpensive to control a large number of AI agents for farming). To accumulate voting power, the attacker deploys a large number of bots at the same time and has them reach the minimum level required to participate in voting, which is $s+1$. This is so that the time spent botting is minimized.

Let the attacker deploy y bots at the same time and let them farm till level $s+1$ to accumulate enough voting power to take over the system (possess more than half of the total voting power). The time required to reach level $s+1$, as calculated earlier is

$$t \in \left[e^{\frac{s+1}{\alpha}} - 1, e^{\frac{s+2}{\alpha}} - 1\right)$$

The probability that a bot was not caught for time t and was able to enter the voting pool is

$$Pr[\text{bot not getting caught}] = (1-b)^t$$

Let $B(t)$ = number of bots that were not caught for time t. Define an indicator random variable $I_i(t)$ for each bot i as

$$I_i(t) = \begin{cases} 1, & \text{bot i was not detected till time t} \\ 0, & \text{otherwise} \end{cases}$$

$$E[I_i(t)] = Pr[\text{bot not getting caught}] = (1-b)^t$$

then we have

$$B(t) = \sum_{i=1}^{y} I_i(t)$$

$$\implies E[B(t)] = y(1-b)^t$$

Hence, the cumulative voting power acquired by the bots is

$$\begin{aligned} \mathcal{V}_{\text{bot}} &= y(1-b)^t \times \gamma(s+1-s) \\ &= \gamma y(1-b)^t \end{aligned}$$

Note that the total voting power at this time instant will be

$$\begin{aligned} &= \sum_{x=s+1}^{M} \mathcal{V}(x) \\ &= \sum_{x=s+1}^{M} n(x) \times \gamma(x-s) \\ &= \gamma \times \sum_{x=s+1}^{M} N\left(\frac{1-e^{-k}}{1-e^{-k(M+1)}}\right) e^{-kx} \times (x-s) \end{aligned}$$

as only those at level $x > s$ are able to vote.

Since the attacker wants to have at least 51% of the vote (of course if the protocol adjusts this number upwards say to 75% like OSRS, then the protocol is more resistant to bots) of the voting pool, the voting power held by the attacker should be more than half the total voting power in the pool. After evaluating and simplification of the above sum we have,

$$\frac{\mathcal{V}_{\text{bot}}}{\sum_{x=s+1}^{M} \mathcal{V}(x)} > \frac{1}{2}$$

$$y > N \left(\frac{e^{-k(s+1)}\left(e^{k(s+1)}(M-s)+e^{k(s+2)}(-M+s-1)+e^{k(M+2)}\right)}{2(1-b)^t \left(e^k-1\right)\left(e^{k(M+1)}-1\right)} \right)$$

The expression on the RHS was calculated using an online tool called Wolframalpha. Thus y which represents the total number of bots the attacker must start with has a lower bound represented by the above equation.

The lower bound of the fraction of bots in the system can hence be represented by the equation

$$f = \frac{y}{N} > \frac{e^{-k(s+1)}\left(e^{k(s+1)}(M-s)+e^{k(s+2)}(-M+s-1)+e^{k(M+2)}\right)}{2(1-b)^t \left(e^k-1\right)\left(e^{k(M+1)}-1\right)}$$

Expected cost of botting attack

We provide an upper bound on the cost of this botting attack. Assume that none of the calculated y bots got caught (though in reality, they did, we are doing this to obtain an upper bound). The cost of getting all y bots to level $s+1$ is

$$Cost \leq y \times (s+1) \times g$$

Note that this directly depends on the number of bots introduced y, which is what we will be considering in our analysis.

Results

Below we have plotted a curve for the equation we obtained earlier for the lower bound of f (the fraction of bots released by an attacker into the system) against b which is the probability of the system correctly detecting a bot.

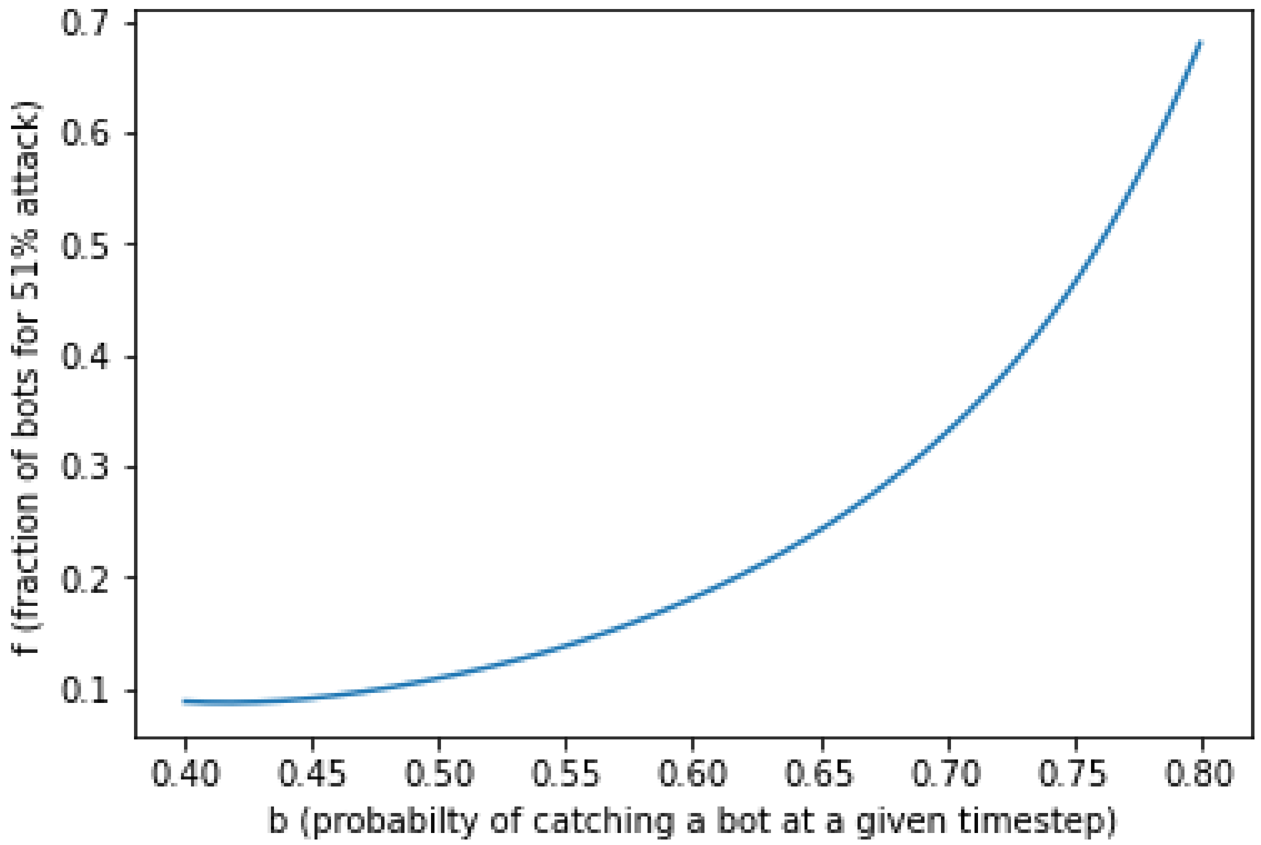

Figure 1: Fraction of bots required as a function of bot-detection probability

In order to plot the complicated equation obtained for f we fixed certain variables. For instance, we fixed the proportionality constants α, β, γ to be 1. We used the software desmos to plot this curve between f and b, by assuming some realistic values for highest level at the current moment in the game M ($= 150$) and the parameter for determining player level distribution k ($= 3$).

As expected, with a more powerful system (higher competence in detecting bots b) it gets increasingly harder for an attacker to successfully bot without being detected. Hence the fraction of bots in the system must be very large when trying to attack a powerful system with a powerful bot detection mechanism.

Future work

In our work, we assumed the existence of a decentralized game in which proving amount of participation should be possible in a secure manner. We're yet to see a direct implementation of this, and we hope to explore how rollups and existing games like Illuvium [4] navigate these issues.

We focused our analysis on a very specific and naive attack, and also for specific instantiations of the level function and player-level distribution. We hope to develop a general framework of analysis for general functions and see how our analysis holds up against real values in practice.

Conclusion

We see that the expected cost of a botting attack scales with the number of bots deployed. It is also resistant to plutocracy since actual time and effort has to be spent in farming experience in the game. Most importantly we hope we've presented a novel method of voting that hasn't been tried before and solves some of the pitfalls of coin-voting at least within the context of crypto-games.

References

[1] Vitalik Buterin. “Governance, Part 2: Plutocracy Is Still Bad”. In: (2016). [Online; accessed 17-Dec-2021].

[2] Vitalik Buterin. “Moving beyond coin voting governance”. In: (Aug. 2016). [Online; accessed 17-Dec-2021].

[3] Paul Tassi. “How To Submit ‘Pokémon GO’ Pokestop Location Nominations In Your Area”. In: (2019). [Online; accessed 17-Dec-2021].

[4] Illuvium White Paper. *A Fully Decentralised RPG and Collection Game Built On The Immutable X L2 Network*. [Online; accessed 16-Dec-2021]. Illuvium, 2021.

[5] Runescape. “Old School Runescape Polls”. In: (2021). [Online; accessed 17-Dec-2021].