Panvala: Self-Organizing the Provision of Non-Rivalrous Goods

Niran Babalola, Akua Nti, Daniel Schifano, Jacob Cantele, Romana Basilaris, and Isaac Kang

June 26, 2019

Introduction

No one knows who funded the creation of Bitcoin, but for every project that has stood on its shoulders, the path to take has become clear. The earliest new projects were trivial forks of Bitcoin, and were rewarded by mining early in the chain's history. Later projects wrote their own codebases, and often "pre-mined" tokens to keep for themselves or sell to their funders. Ethereum itself held one of the first crowdsales to fund the creation of a new currency.

Ethereum's ability to execute arbitrary smart contracts led to an explosion of new token systems that used the same funding model, and several promising systems that started this way now seem on a credible path towards user adoption. Ethereum, however, has run into the limits of this model. While it's a very effective strategy for launching new systems, sustaining these systems is still difficult.

In the early days of Ethereum, many people worked on improving and building on top of the network because they felt that the Ether they held would become more useful as the network improved. Since then, the price has increased over 1000x, and that incentive has become less perceptible. At this point, it's hard for any single team to feel like their work has any bearing on the demand for Ether, and the effects on the community have been significant. James Prestwich, the founder of a company that enables cross-chain transactions, has observed that "one of Ethereum's poorly understood weaknesses is it's inability to retain talent and experience over time."[1]

On December 18, 2018, Preston Van Loon, an engineer working on implementing Ethereum 2.0, aired his concerns on Twitter. "Our biggest distraction [at Prysmatic Labs] is that we are still working full time for other jobs. Even with recent grants, it's hardly enough to take the whole team full time with significant pay cuts and it's certainly not even for us to scale the team to where we need it." In response, Vitalik Buterin tweeted "Just sent 1000 eth. Yolo."[2] While the contribution was appreciated,

[1] https://twitter.com/__prestwich/status/1129101755424362498
[2] https://twitter.com/vitalikbuterin/status/1075181710730506240?lang=en

this “YOLO grant” of ether has become the clearest illustration of the absence of institutions in our community to provision public goods. If our roads were paved or our updates on each flu season were funded by one-off grants from individuals, we’d be rightly terrified.

But beyond the risks posed by uncertain development funding, the threat of competition has loomed over our community for years. We’ve always known that Ethereum needed changes for scaling to accommodate the demand for smart contracts, and Ethereum loyalists haven’t been the only people with good ideas about how to do that. When faced with the choice of whether to cooperate with Ethereum or to compete with it, many teams with good ideas choose to compete. It’s far easier to reward workers and investors if you start your own blockchain—you can issue new tokens to whoever you want. If you improve Ethereum, how do you reward your team?

Panvala was designed to solve this problem. We want it to be more rewarding to cooperate with the Ethereum community than it is to compete with it. To do this, we’ve built a system you can think of as a decentralized version of the Ethereum Foundation that runs on its own token.

The Ethereum Foundation has faced regular barrages of criticism over the years, but Panvala was not built because the Ethereum Foundation has failed to achieve its goals. Panvala was built for the Ethereum Foundation to *succeed* at its goals. On May 21, the Ethereum Foundation updated the community about how it intends to use its funds, including a path towards future funding:

> We are also working to grow the Ethereum ecosystem’s funding base. This means encouraging other organizations besides the Foundation to support high-priority projects, and supporting innovative mechanisms for funding ...

> Efforts like these give us better leverage from our existing resources, and help build a sustainable path for funding vital projects far into the future.[3]

The Ethereum Foundation's remaining 600,000 ETH will not last forever, and the $30 million that it intends to spend over the next year is roughly the same level of funding that competing projects are allocating even though those projects have not even launched live networks[4]. That's why we need Panvala's ability to raise new funds and act as a decentralized Ethereum Foundation.

More abstractly, Panvala is a system that self-organizes the provision of non-rivalrous goods. Non-rivalrous goods are cheap to provide for each additional person, like a movie theater that isn't capacity or new research discoveries. For comparison, consider private goods: products that are hard to share, like a mobile phone or a toothbrush. Each individual pays for the private goods they can afford. Aided by government regulations, the marketplace self-organizes the provision of goods like these, so the supercomputers in our pockets and the cars we drive constantly get better, cheaper, and more efficient. But for non-rivalrous goods that can be shared, *Panvala* pays for what it can afford. It self-organizes the provision of shared goods, and constantly improves them to earn and retain the loyalty of the people who donate to it.

Economic Model

Grants

Panvala grants are awarded to teams who work to fulfill the Ethereum vision with infrastructure and applications that the whole ecosystem depends on. Grants are issued using the system's own token, so they can be issued from the first day of the system. The value of these early grants cannot be determined until there are people

[3] https://blog.ethereum.org/2019/05/21/ethereum-foundation-spring-2019-update/
[4] https://etherscan.io/address/0xde0b295669a9fd93d5f28d9ec85e40f4cb697bae

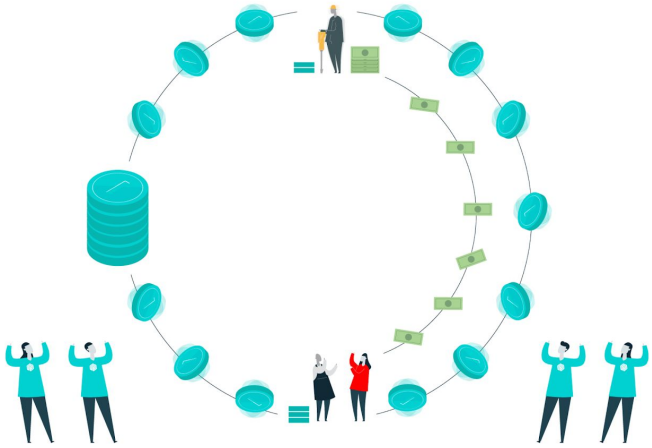
who want to buy the tokens. The intended buyers of the granted tokens are donors who want to fund these ecosystem development projects.

Panvala puts very few restrictions on grant issuance.

1. The token has a fixed supply of 100 million tokens.
2. The rate that donated tokens can be released is limited by the token capacitor.
3. The token holders must govern the issuance of grants using slate governance.

Other than these three restrictions, the token holders are free to structure their spending of released tokens however they would like. Since most non-equity funding in our community operates using grants, we've adopted that model as the default. However, the system can also accommodate a budget model, where teams receive recurring funding to perform an ongoing function, in addition to project-based grants to teams that pursue a variety of funding sources.

Donations



Panvala is designed to organize a flow of funds towards work that our community wants to support. While Panvala's grants are issued to individual teams, donations are made to the system as a whole, not to individual projects. These donations are made using Panvala's native tokens. Panvala's tokens can be acquired with more liquid currencies like USD, KRW, and ETH, but Panvala has no ability to hold any currency other than its own. Donors buy tokens from teams who did work (directly from teams, or indirectly via exchanges) then donate those tokens back into

the smart contract they came from. That's what completes that cycle that makes Panvala sustainable.

Buying tokens from a team that received a grant is necessary, but not sufficient to make a donation. A team that sells its tokens now has money to fund its work, but the buyer of the token has three options: donate the tokens back to the system, hold on to the tokens indefinitely to vote on Panvala's decisions, or hold on to the tokens in order to sell the tokens later. When a donor buys resold tokens, their purchase does not fund any work. That's why the purchase of tokens from a team is not considered a donation. The donation occurs when you deposit the tokens you acquired back into the system so they cannot be resold.

The Panvala community recognizes the individuals and businesses who make donations to encourage more people to join them. Businesses who sponsor Panvala earn the community's gratitude and attention based on the annual pledge required for their sponsorship package (see *Appendix B: Initial Sponsorship Program*). Individual Panvala patrons are recognized by the size of the recurring donations they have pledged to make (see *Appendix C: Initial Patron Program*). The benefits of these programs are fulfilled by token holders, grant recipients, and the whole Panvala community, not by the Panvala Launch Team.

The Token of Panvala

The token of Panvala is the *pan*. When specifying amounts of pan, the 🍳 symbol ("cooking" emoji, U+1F373) is used before the amount. For instance, 🍳5000 might be required to stake on a slate, and a batch of grants might have 🍳2 million available. "PAN" is used to represent the token of Panvala when trading on exchanges or in any other context where a ticker symbol is useful, alongside USD, KRW, and ETH. When the full name of the token is needed (e.g "United States dollar" or "Korean won"), the token is the *Panvala pan*. Pan is both singular and plural.

Instead of coordinating donations by pooling money and spending it on workers, Panvala coordinates donations by issuing grants of its own tokens to workers. Donors

buy those tokens and donate them back to the token capacitor. There is no way to donate dollars, won, or Ether directly to Panvala. Those currencies are used to acquire pan from workers and donate it back to the system.

Purposes of the Token

Property Rights

Using property rights to organize cooperation makes it easy for people to do work and be rewarded for it without needing anyone's approval to do so. As a result, people capable of improving property can identify themselves without needing to be recognized by a central planner. We're familiar with how this plays out with land or intellectual property, and these same dynamics can be harnessed to organize the provision of public goods.

A normal foundation hires donor development staff to increase the flow of donations into the organization. Instead of having a handful of donor development employees who are rewarded for increasing donations, Panvala can tap thousands or even millions of token holders, who can all be rewarded for increasing donations to the ecosystem. The more donations are made to the system, the more demand there is for the tokens held by the people recruiting more donors. Token holders have an incentive to tap their social networks to recruit more donors to fund the work we all care about.

Principal-Agent Alignment

Many donor-funded organizations are ineffective. The management of these organizations act as an agent for their donors, who expect them to maximize the good that can be done with their donation. However, since their effectiveness is hard to measure and often defined subjectively by a handful of large donors, the management of the organization can be far removed from any increases or decreases in their effectiveness compared to for-profit organizations where there are clearer measures of impact. It is notoriously difficult to solve principal-agent problems and get agents to act in the interests of their principals.

In Panvala, pan holders are the agent for Panvala's donors. Pan gives its holders a stake in the system's future. If pan holders vote to issue grants effectively, they will grow the number and size of donations made to Panvala, which increases demand for the pan they hold. If they issue grants poorly, donations will decrease, as will demand for their holdings. As a result, we expect pan holders to be very responsive to the desires of current and potential donors, even though donors themselves don't have votes in the system.

Subsidiarity

While each token gives influence over all of the system's actions, they also create a locus for subsidiarity to allow decision-making to be pushed down to lower levels of Panvala. Some organizations divide themselves by geography or by function, but we expect that the most effective way to divide Panvala will be by pools of staked tokens. Today, the individual donors to Panvala are recognized at the top level of the system, but in the future, it might be the staking pool those donations are assigned to that matters the most. Those staking pools can then be evaluated and recognized based on a function of the number of tokens in their pool and the size of the donation flow they have organized.

Unit of Account

Donations are measured in Panvala's unit of account, not in pan. At launch, Panvala's unit of account is equivalent to 1 USD, and while this remains true, we avoid mentioning the unit of account in favor of just using USD. However, we expect that Panvala's unit of account will be adjusted over time to achieve stable values for as much of Panvala's global community as possible. When that happens, the unit of account will be named by the token holders.

Initial Distribution

Panvala started issuing grants on February 1, 2019. At that point, 50 million pan were held in the token capacitor, and the remaining 50 million pan were reserved for distribution by the Panvala Launch Team. Through August 2, a total of 6,093,697 pan

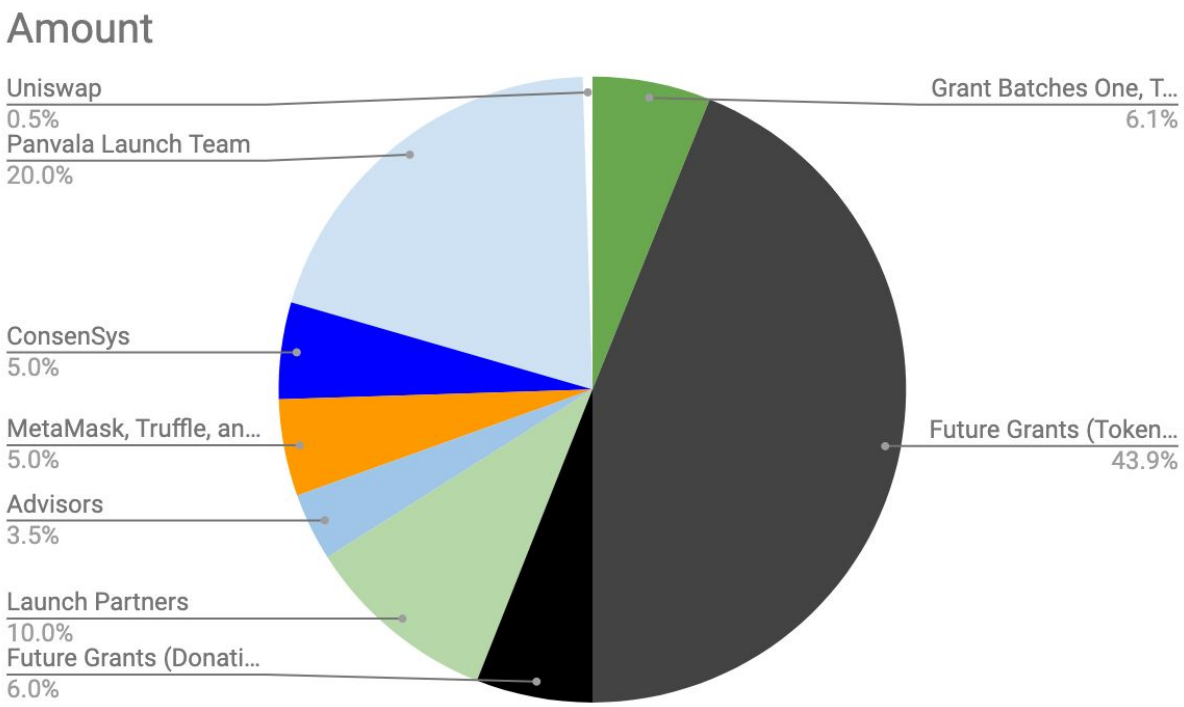
were issued in Batches One, Two and Three of grants, leaving 43,906,303 pan in the token capacitor.

From the 50 million pan for the Panvala Launch Team to distribute, the team retains 20 million pan. ConsenSys holds 5 million pan, plus another 5 million pan for projects at ConsenSys that the whole community can use, like MetaMask, Truffle, and the Burner Wallet. ConsenSys was allocated an additional 6 million pan, but chose to donate them back to the token capacitor for future grants. (As a result, the token capacitor will be initialized on chain with a balance of 49,906,303 pan.) 500,000 pan will be deposited in Uniswap, which is the default method for donors to acquire the tokens to donate. Advisors hold 3,500,000 pan.

Launch partners hold the remaining 10 million pan. Launch partners are selected grant recipients from Batches One, Two, and Three who have committed to donate by earning or purchasing pan over the first two years of Panvala. They each have monthly donation targets that must be met for these tokens to be released, which can be made up to three months in advance. If they fall more than one month behind, the remaining tokens assigned to them will be donated to the token capacitor.

All pan begins in the hands of people who have done work to fulfill the Ethereum vision. All pan other than grants are subject to vesting: one pan vests for each pan released from the token capacitor.

Destination	Amount
Grant Batches One, Two, and Three	6,093,697
Future Grants (Token Capacitor)	43,906,303
Future Grants (Donation from ConsenSys)	6,000,000
Launch Partners	10,000,000
Advisors	3,500,000
MetaMask, Truffle, and Other Infrastructure	5,000,000
ConsenSys	5,000,000
Panvala Launch Team	20,000,000
Uniswap	500,000
	100,000,000



The Token Capacitor

The *token capacitor* is the smart contract that releases tokens for grants and accepts tokens as donations. The tokens in the capacitor are released at a rate that decays exponentially over time. Panvala's token capacitor is configured with a half-life of 1456 days (four 52-week years), like Bitcoin's block reward decay. This half-life is informed by the practices of other digital currencies, as well as common practices for issuing shares of corporations. However, it's still just a guess. We've hardcoded this value not because it's definitely the right choice forever, but because we believe that making it easy to alter the release curve would deter participation.

Withdrawing tokens from the token capacitor requires permission to be granted through the slate governance process. That process has its own timeline for granting permissions, but the token capacitor itself does not enforce restrictions on the timing of withdrawals. It only restricts the amount of tokens that can be withdrawn based on

the balance after the last withdrawal or donation, the time of that change, and the amount of time that has elapsed since then.

Exponential Decay

The token capacitor releases tokens at rates such that its balance decays exponentially. Ideally, this decay would follow the formula for exponential decay:

$$N(t) = N_0(\tfrac{1}{2})^{\frac{t}{t_{\frac{1}{2}}}}$$

$N(t)$ is the new balance,

N_0 is the previous balance,

t is the amount of time that has elapsed since tokens were last released, and

$t_{\frac{1}{2}}$ is the half life of the token capacitor, 1456 days.

However, since the floating point operations needed to implement this formula have determinism issues, it's a poor fit for execution on a blockchain, where thousands of nodes need to agree on the result. The Ethereum Virtual Machine does not include floating point instructions for this reason. This leaves us two attractive approaches for implementing exponential decay: store a *lookup table* for pre-calculated values of $(\tfrac{1}{2})^{\frac{t}{t_{\frac{1}{2}}}}$ for selected values of t, or create a *schedule* of release rates to approximate exponential decay with a piecewise function.

It is easier to verify that a particular implementation of a piecewise schedule is free of any flaws that could throw off the supply policy of the system. Piecewise functions are deterministic, while attempting to approximate the curve more closely leads to behavior that depends on the prior sequence of balances and multipliers used from the lookup table. In addition, since the goal of these smart contracts is to build consensus within a large community, it's useful to be able to communicate exactly how many tokens should be released when using math that the public can do in their heads. Bitcoin's block reward schedule also approximates exponential decay in this manner.

However, Panvala's token capacitor releases are based on the current balance, not the current time like Bitcoin. Bitcoin can read from the clock to determine how many halvings have occurred, but Panvala would have to store or calculate the balance boundaries for each release rate. With donations, the balance can fluctuate unpredictably, and any piecewise schedule implementation would have to account for releases that cross boundaries of the schedule. Together, these concerns increase the complexity of the implementation to a degree that accepting the flaws of the lookup table approach is the right tradeoff to make.

Creating the Lookup Table

To create the lookup table, we must first select the smallest time interval that the table will support. The smaller the interval, the larger the error from truncation that compounds with every iteration. To use these multipliers with integers, we must choose a precision level to multiply by before using the multiplier, then divide out the precision factor when we're done. We've chosen one day as the smallest interval and 1×10^{12} as our precision factor. Together, they produce an error of about 531 tokens out of 50,000,000 over one half life.

We fill the rest of the lookup table with powers of two to be able to maintain more accuracy when more time has elapsed between the capacitor's balance changes. However, we expect to achieve a flow of donations that exceeds one per day, which would cause the multiplier for one day to be used far more often than any other.

Each time we multiply by a multiplier, any present error compounds. As a result, using multipliers for fewer elapsed days over and over releases slightly more tokens than performing fewer multiplications using multipliers for more elapsed days.

Days Elapsed	Multiplier	Integer Multiplier	Balance at Half-Life
1	0.9995240507	999524050675	49,999,469
2	0.9990483279	999048327879	49,999,733
4	0.9980975614	998097561438	49,999,872
8	0.9961987421	996198742149	49,999,937
16	0.9924119339	992411933860	49,999,968

32	0.9848814465	984881446469	49,999,981
64	0.9699914636	969991463599	49,999,990
128	0.9408834395	940883439455	49,999,995
256	0.8852616466	885261646641	49,999,997
512	0.783688183	783688183013	49,999,999
1024	0.6141671682	614167168195	49,999,999
2048	0.3772013105	377201310488	N/A

Locked and Unlocked Tokens

The total balance of the capacitor is divided between *locked* tokens and *unlocked* tokens. Unlocked tokens are the only ones that can be withdrawn, and locked tokens are the only tokens involved in decay calculations. Each time tokens are received or sent by the capacitor, we move tokens from the locked balance to the unlocked balance *before* adjusting to any request to deposit or withdraw tokens. If the unlocked balance were updated *after* a deposit, that new donation would be included in the balance of tokens to be decayed as if the tokens had been there since the last balance update. It the unlocked balance were updated after a withdrawal, withdrawal might be incorrectly rejected if the tokens to withdraw weren't unlocked yet.

A standalone function is available to sweep the appropriate number of locked tokens to the unlocked balance by the following method:

1. Calculate the elapsed days since tokens were last unlocked.
2. If the number of elapsed days is odd, multiply the locked balance by the lowest multiplier. Divide by the precision factor to determine the number of tokens that remain in the locked balance.
3. Add the difference between the previous and new total of locked tokens to the balance of unlocked tokens in the contract's storage.
4. Divide the elapsed days by two, shift to the next multiplier, and repeat steps 2-5 until no time is remaining. This will take $log_2(t)$ iterations, and will release tokens for up to 4095 days in one transaction.
5. Store the difference between the total token balance of the contract and the balance of unlocked tokens as the new locked balance.

6. Add the elapsed time to the last unlocked time.

To illustrate the process more precisely as code, here is an excerpt from `TokenCapacitor.sol`:

```
    function calculateDecay(uint256 _days) public view returns(uint256) {
        require(_days <= (2 ** decayMultipliers.length) - 1, "Time interval
too large");

        uint256 decay = scale;
        uint256 d = _days;

        for (uint256 i = 0; i < decayMultipliers.length; i++) {
           uint256 remainder = d % 2;
           uint256 quotient = d >> 1;

           if (remainder == 1) {
                uint256 multiplier = decayMultipliers[i];
                decay = decay.mul(multiplier).div(scale);
           } else if (quotient == 0) {
               // Exit early if both quotient and remainder are zero
               break;
           }

           d = quotient;
        }

        return decay;
    }
```

Anyone can send a transaction that unlocks tokens and advances the last unlocked time by a given number of days that is less than or equal to the total elapsed time since

tokens were last unlocked. Multiple transactions will be needed to bring the contract up to date if 4096 days or more have elapsed since tokens were last unlocked. Before processing a donation or a withdrawal, this function is called within the same transaction to minimize the number of transactions needed to keep the capacitor state up to date.

When permission has been granted to withdraw tokens, the number of unlocked tokens is reduced by the withdrawal amount. Withdrawals greater than the number of unlocked tokens revert the transaction. Donations are added directly to the locked token balance.

Token Release Schedule

For reference, an approximate schedule of the first eight years of token capacitor releases is included below. This schedule assumes that the calculations to release tokens from the capacitor are done once per quarter, when they will likely be done more often, leading to slight variances in the numbers of tokens released. Donations to the token capacitor are not considered in this time-based chart. As a reminder, the token capacitor actually releases based on its *current balance*, not based on the *current time* as in Bitcoin.

This chart can be used to consider donations by thinking of each donation as rewinding the token capacitor back in time by the equivalent number of tokens. Without donations, the balance of the capacitor moves down the curve at a predictable rate. Each donation restores the balance to an earlier point on the curve, so while the dates on this chart won't match the balances in real life, you can look up the current balance on this chart to understand the current status of the system.

	Epoch	Locked Tokens	New Unlocked Tokens	Allocated supply	Quarterly Inflation	Annualized Inflation
2/1/2019	1	47,880,164	2,119,836	52,119,836	4.24%	18.07%
5/3/2019	2	45,850,202	2,029,962	54,149,798	3.89%	16.51%
8/2/2019	3	43,906,303	1,943,899	56,093,697	3.59%	15.15%
11/1/2019	4	42,044,819	1,861,484	57,955,181	3.32%	13.95%
1/31/2020	5	40,262,256	1,782,563	59,737,744	3.08%	12.88%

5/1/2020	6	38,555,268	1,706,988	61,444,732	2.86%	11.93%
7/31/2020	7	36,920,651	1,634,617	63,079,349	2.66%	11.07%
10/30/2020	8	35,355,336	1,565,315	64,644,664	2.48%	10.30%
1/29/2021	9	33,856,385	1,498,951	66,143,615	2.32%	9.60%
4/30/2021	10	32,420,985	1,435,400	67,579,015	2.17%	8.97%
7/30/2021	11	31,046,441	1,374,544	68,953,559	2.03%	8.39%
10/29/2021	12	29,730,173	1,316,268	70,269,827	1.91%	7.86%
1/28/2022	13	28,469,711	1,260,462	71,530,289	1.79%	7.37%
4/29/2022	14	27,262,688	1,207,023	72,737,312	1.69%	6.92%
7/29/2022	15	26,106,839	1,155,849	73,893,161	1.59%	6.51%
10/28/2022	16	24,999,994	1,106,845	75,000,006	1.50%	6.13%
1/27/2023	17	23,940,076	1,059,918	76,059,924	1.41%	5.77%
4/28/2023	18	22,925,095	1,014,981	77,074,905	1.33%	5.45%
7/28/2023	19	21,953,146	971,949	78,046,854	1.26%	5.14%
10/27/2023	20	21,022,404	930,742	78,977,596	1.19%	4.86%
1/26/2024	21	20,131,123	891,281	79,868,877	1.13%	4.59%
4/26/2024	22	19,277,629	853,494	80,722,371	1.07%	4.34%
7/26/2024	23	18,460,320	817,309	81,539,680	1.01%	4.11%
10/25/2024	24	17,677,662	782,658	82,322,338	0.96%	3.90%
1/24/2025	25	16,928,187	749,475	83,071,813	0.91%	3.69%
4/25/2025	26	16,210,487	717,700	83,789,513	0.86%	3.50%
7/25/2025	27	15,523,215	687,272	84,476,785	0.82%	3.32%
10/24/2025	28	14,865,081	658,134	85,134,919	0.78%	3.15%
1/23/2026	29	14,234,850	630,231	85,765,150	0.74%	2.99%
4/24/2026	30	13,631,339	603,511	86,368,661	0.70%	2.84%
7/24/2026	31	13,053,414	577,925	86,946,586	0.67%	2.70%
10/23/2026	32	12,499,992	553,422	87,500,008	0.64%	2.57%

Recording Donations

Donations are recorded along with metadata to let the public know the context of the donation. In particular, it's valuable for the public to know the donor's intent and their view of the market when the donation was made. The current price of ETH/USD, current price of PAN/ETH, the intended donation in USD, and the terms of the pledge the donor intends to fulfill are useful context to record as metadata for donations.

Donation Strategies

The token capacitor coordinates donations in a new way that is difficult to reason about since no one has done it before. We've thought through a hypothesis of the dynamics that we expect to play out.

Large, one-time donations have no effect on the long-term flow of donations, so they do not increase the system's ability to fund work over the long run. It's not much different from directly funding work as an individual, but Panvala's goal is to build up a flow of funding that teams can count on. On the other hand, long-term commitments to recurring donations can change the flow of donations for an extended period of time, which allows more work to be rewarded using fewer tokens. Teams that expect Panvala to receive a steady flow of donations can stabilize their expectations of what the tokens can fund, which allows them to plan ahead to do work for the Ethereum ecosystem rather than finding private companies to hire them.

Donors who are long-term token holders are faced with a choice: should they buy new tokens to donate, or should they donate from the tokens they already hold? While both options are valid donations, donating by reducing your long-term holdings of Panvala tokens has counterintuitive effects. The purchasing power of the next batch of grants is dependent on the flow of tokens acquired, not the balance of tokens in the token capacitor. Donations that add tokens to the token capacitor but don't involve newly acquired tokens have no effect on the purchasing power of the system: the tokens released each quarter just allocate the value flowing into the system from workers and token buyers. The increase in tokens granted will be accompanied by a decrease in the token price if the flow of value to acquire tokens stays the same.

To make the largest impact with your donations, view them as coming from your income rather than your holdings. Donate tokens immediately after acquiring them, whether you earned them directly for work, or purchased them from someone else. To make an impact with tokens you held onto, hold them indefinitely and use their voting power to steer the system. Ideally, make an impact in both ways: holding tokens to

vote while donating regularly from your income has the largest positive impact on Panvala's capacity to fund the work that you enjoy donating to.

Slate Governance

Panvala makes decisions using *slate governance*. Each quarter, the system approves one slate of actions and all of the individual proposals it contains. Pan holders who don't believe that a slate represents the consensus of the community can propose a competing slate of proposals. Pan must be staked on each proposed slate, and the tokens staked on losing slates are donated to the token capacitor.

Each slate is associated with the *recommender* who authored the slate. While most on-chain decision-making systems involve approving or rejecting individual proposals, the main question we answer each quarter is "which decision-making process should we use?" The recommender of a slate always represents a particular decision-making process, even if it is poorly defined. When done well, recommenders make their decision-making process clear, and their recommended slate represents the output of that process. Some example processes for recommenders include voting off-chain, electing representative bodies, or relying on reputable authorities. Challenging individual proposals in done within the process that the recommender has already defined. Challenging a slate is like amending a constitution: you change the rules when they no longer serve the community's goals, but not because you're unsatisfied with a particular outcome.

Design Goals

Slate governance was designed to avoid common pitfalls we've seen in decentralized systems. The systems that have been deployed so far often see large numbers of decisions made, but low voter turnout has been a signal that token-based voting might not actually be rewarding the effort it takes to evaluate decisions. Other systems see too few decisions: Bitcoin has been famously resistant to change over the years, for better or for worse. We see this inertia as an emergent outcome of Bitcoin's

fork-based governance rules. The principles that justify resistance to change have been post hoc rationalizations of an emergent phenomenon.

Fork-based governance has also made its way to on-chain systems. TheDAO was a fork-based organization in which token holders could fork off a new organization after any decision they didn't want to cooperate with. (TheDAO was hacked before we could see whether its design would work[5].) Moloch DAO follows in TheDAO's footsteps with its "ragequit" functionality: during the waiting period after each approval, anyone can withdraw their remaining Ether if they don't want to support the proposal. These designs make it easy to decide to join since there's no real commitment, but their potential is constrained by the need to play it safe to keep people from leaving. Panvala avoids fork-based governance with the goal of building a committed community that cooperates even when they don't get their way.

On-chain voting has a poor track record. We believe it should be used as a mechanism of last resort rather than in the typical operation of a system. Our approach is similar to Plasma, a blockchain scaling strategy that builds child chains that can be entered and exited via a root contract. When a Plasma child chain is operating normally, very few transactions are sent off chain. When something goes wrong, that could trigger thousands of transactions to the root contract on the blockchain so people can remove their assets. Similarly, when everything is operating normally in Panvala, very few governance transactions are sent. During times of discord or attacks, thousands of transactions can be triggered for votes to be tallied to resolve the problem.

Resources and Permissions

Many designs for on-chain governance are oriented around approving transactions for a shared account to send, allowing token holders to collectively perform the same actions that a person can send a transaction to execute. Traditional multisignature wallets are the simplest form of this design, and Aragon DAOs are complex, token-based designs that control a single Aragon Agent that sends arbitrary transactions. Panvala avoids this design primarily to avoid becoming a shared pool of

[5] https://www.bloomberg.com/features/2017-the-ether-thief/

assets. Since Panvala cannot send transactions, it can't hold any assets other than its own token.

Instead of sending arbitrary transactions, Panvala's *resources* allow anyone to request *permission* to interact with them. A resource is any smart contract (like the token capacitor) that defines permissions, which are then fed into the *gatekeeper* for approval. The gatekeeper contract is where token holders create slates of permission proposals, and vote on them if necessary. Resources call two functions on the gatekeeper:

```
function requestPermission(bytes memory metadataHash) public returns(uint)
function hasPermission(uint requestID) public view returns(bool)
```

Resources store the permission identifier returned from `requestPermission` along with bookkeeping information about each permission request so they can check if the permission has been granted, ensure that one-time use permissions haven't been used already, and execute the desired action. For instance, the token capacitor stores `Proposal` structures for each request to withdraw tokens:

```
struct Proposal {
    address gatekeeper;
    uint tokens;
    address to;
    bytes metadataHash;
    bool withdrawn;
}
```

Keeping track of the gatekeeper instance that was used for each proposal is particularly important to ensure that upgrades of the governance contracts go smoothly (see *Upgradeability*).

Slates

A slate contains zero or more permission requests for a single resource. Slates are authored by recommenders, who can choose whether to stake pan on their slate to add it to the contest. If the recommender doesn't stake on the slate, someone else must stake on it or the slate will be ignored.

Each resource has its own set of slates that compete to approve permissions each quarter. As a result, each resource has a separate contest occurring each quarter. Some resources might trigger a vote during the same quarter that other resources have no contest.

If a resource only has one staked slate during a quarter, that slate automatically wins, and its permissions are approved.

Slates submitted without any permission requests to approve are called *blank slates.* Recommending a blank slate is appropriate when the consensus of the token holders is to take no action for the quarter.

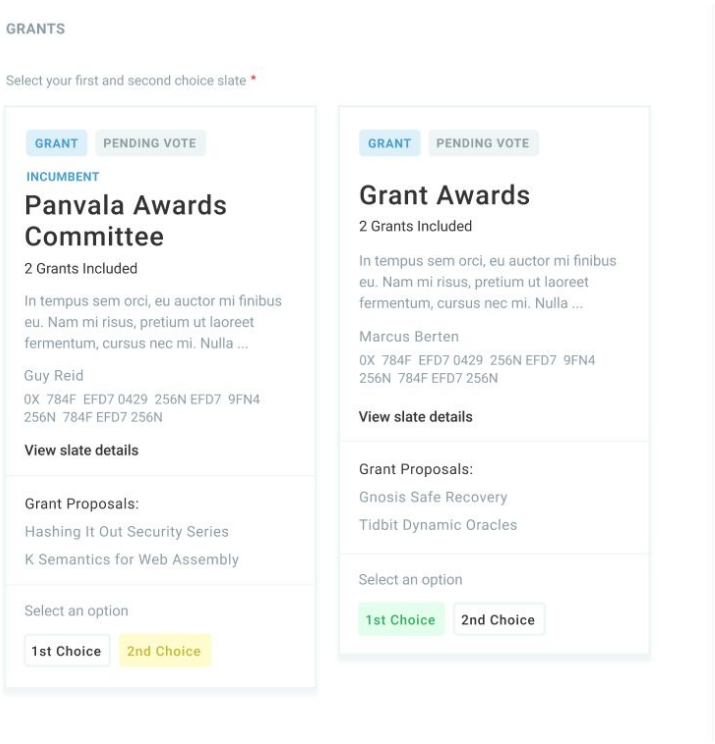
Incumbency

Recommenders represent an off-chain process for reaching consensus about which permissions to approve. Panvala highlights the identity of slate recommenders to focus the on-chain decision away from the merits of the permissions on a slate and towards the process by which those permissions were added to the slate. The recommender of the last successful slate for a resource is the *incumbent* for that resource. The incumbent is effectively the embodiment of the bylaws that are currently in effect. If the incumbent's slate loses a contest, it's not just their proposals that were rejected, it's the implicit bylaws they represent that were rejected. Hopefully, they were replaced by better bylaws.

If no one submitted a slate for a quarter, the last incumbent persists. Incumbency can never be vacated.

Voting

Panvala uses a commit-reveal process to tally votes. During the commit phase, voters submit a hash of their vote, while keeping the vote itself and a random salt secret. During the reveal phase, no new commitments can be made, and earlier commitments can be revealed. This approximates the experience of typical elections where no running tally of votes is available until the polls close.



Each token can be used to acquire one vote. To acquire voting rights, your tokens must be deposited in the gatekeeper contract before or during the commit phase, and cannot be withdrawn until the commit phase is over. This prevents the same tokens from being used to acquire multiple votes. Voters can delegate their votes to another Ethereum account. This allows voters to store the keys that control their tokens safely while delegating to a frequently used key that cannot withdraw the tokens.

If there is an active contest but no one votes, the default action is to reject all slates for that resource.

Ranked Choices and Runoffs

When a contest has two competing slates, the slate with more votes wins. When a contest has three or more competing slates, voters can indicate their first and second choices for slates. If any slate gets more than half of the first choice votes, that slate wins. Otherwise, all slates but the top two recipients of first choice votes are

eliminated. Any second choice votes for the top two slates are counted for voters whose first choice was eliminated. The remaining slate with the most votes wins.

Candidate	Round 1	Round 2
Slate A	45 million	52 million
Slate B	~~20 million~~	
Slate C	35 million	48 million

An example runoff

Epochs

Each period of governance is called an *epoch*, and lasts thirteen weeks. Epoch zero started on November 2, 2018 at 1700 UTC and ended with the issuance of Batch One of grants on February 1, 2019.

Epoch Number	End of Epoch	Time (UTC)	Time (Austin, TX)
0	2019-02-01	1700	11 am CST
1	2019-05-03	1700	12 pm CDT
2	2019-08-02	1700	12 pm CDT
3	2019-11-01	1700	12 pm CDT
4	2020-01-31	1700	11 am CST
5	2020-05-01	1700	12 pm CDT
6	2020-07-31	1700	12 pm CDT
7	2020-10-30	1700	12 pm CDT
8	2021-01-29	1700	11 am CST

The first deadline within an epoch is the slate submission deadline. After this deadline, no more slates can be staked or recommended for the given resource. The hard deadline for slate submission is eleven weeks into an epoch, but to prevent slates from being snuck in at the last minute, a soft deadline starts 5.5 weeks into the epoch,

and adjusts as slates are submitted. Each time a slate is staked for a resource, the soft deadline is reset to be halfway between the current time and the hard deadline, which makes the soft deadline approach the hard deadline with each slate submission. As a result, each resource can have a different soft deadline for slate submission.

At the end of week 11, the voting commit period begins and lasts for one week. At the end of week 12, the voting reveal period begins and lasts for one week. Once the epoch has ended, no more votes can be revealed, so the contests can be finalized and permissions approved for the winning slates. Each permission request expires at the end of the following epoch.

The Parameter Store

The *parameter store* holds key-value pairs that are subject to Panvala's governance process. The parameter store gives us a common API for proposing, approving, and executing changes to parameters instead of needing different functions to be called to change different parameters. This pattern was inspired by the "Parameterizer" contract from the original token-curated registry implementation.[6]

Initial Parameters

Parameter Name	Parameter Type	Description	Value
`gatekeeperAddress`	`address`	The address of the active gatekeeper contract used to govern Panvala.	*Determined at deploy-time*
`slateStakeAmount`	`uint`	The number of pan that must be staked in order for a slate recommendation to be considered by pan holders.	50,000 pan
`archives`	`bytes32`	The git hash of the Panvala archives, which contain documents that have been endorsed through	TBD. The initial repository will be hosted at https://github.com

[6] https://github.com/skmgoldin/tcr/blob/master/contracts/Parameterizer.sol

		Panvala's governance process.	/Panvala/archives.

Upgradeability

While the token capacitor and parameter store contracts cannot be changed, the gatekeeper contract is designed to be upgraded over time. Rather than the upgradeable contracts pattern popularized by OpenZeppelin and Aragon where a contract maintains its address and storage while the code changes, we use an older "EternalDB" pattern popularized by Peter Borah and the Colony team[7][8]. In the latter pattern, state is stored separately from code, and the address of the code that controls access to the state changes with each upgrade.

Our "EternalDB" is the parameter store. Rather than having a modifiable owner that pushes authorized changes into the contract, the parameter store pulls changes from the gatekeeper contract, which is specified by a parameter as well. As long as the new gatekeeper contract follows the permissions API from the original contract, the new version can implement whatever decision-making logic that is needed.

Updating the gatekeeper points all relevant contracts away from the old gatekeeper and towards the new one. Since the epoch in which the gatekeeper is changed can contain many other decisions as well, the upgrade approach taken by the community has significant potential effects. Permissions will function as expected during the transition as long as resources store the address of the active gatekeeper alongside each permission to ensure that permission lookups aren't misdirected by gatekeeper upgrades.

Token balances and delegations can be transferred by individual voters, but care must be taken to inform voters of the transition with enough time to prepare. Incumbents are harder to transfer: since the new gatekeeper must be deployed before the

[7] https://github.com/ConsenSys/dapp-store-contracts/blob/master/contracts/EternalDB.sol
[8] https://blog.colony.io/writing-upgradeable-contracts-in-solidity-6743f0eecc88/

permission has been granted to upgrade, the new gatekeeper won't know the incumbents from that epoch unless it was written to be able to fetch them.

In this context, gatekeeper upgrades have three options when it comes to transitioning state: they can do a *clean break* that loses incumbent data, they can *fetch* the incumbent data and anything else they'd like to transition in an initialization function on the new gatekeeper, or they can *preserve all state* using Merkle proofs or a new gatekeeper contract that uses an upgradability pattern that updates the code within a contract. If no other known contracts are using the incumbent data, clean break upgrades should be fine, but third-party contracts may have begun relying on the data without notifying you. Fetching the desired state to transition avoids breaking known or unknown contracts that rely on incumbent data.

Governance Demonstrations

The first demonstration of slate governance was performed by the Panvala Mark Council, a group of Ethereum community members that was appointed for this purpose. On October 25, 2018, the Panvala Mark Council convened to recommend issuing a Panvala Mark for a simple multisignature wallet smart contract.[9]

The Panvala Awards Committee recommended slates of grants for three batches. They met on January 28, 2019 for Batch One[10], and on April 29 for Batch Two[11]. They will meet on June 25 to recommend grants for Batch Three.

Videos are available for each of these meetings.

---

[9] https://medium.com/@Panvala/how-we-issued-a-panvala-mark-3f716b5fa910

[10] https://medium.com/@Panvala/twelve-grants-awarded-in-batch-one-of-panvala-token-grants-59b8df7422fe

[11] https://medium.com/@Panvala/seven-grants-awarded-for-ethereum-2-0-and-scaling-teams-in-panvalas-second-batch-626f74f0a3bb

Error Recovery

Smart contracts are rigid programs that are difficult and risky to modify after they have been deployed. Many contracts in other applications have had bugs and security flaws. While we've taken the best practice precautions to avoid these issues, it's still possible that problems will arise that we haven't foreseen.

In the event of a serious error, the incumbent recommender for the token capacitor resource should make a recommendation for recovering from the error. In some cases, it might be easy to deploy fixed versions of the contracts with a new token that copies the balances at the time the error occurred. More complicated errors might be more difficult to recover from.

Since Panvala only holds its own token, all errors can be recovered with a new instance of the system. Determining the initial state of that new instance is the hard part, and the incumbent recommender for the token capacitor should lead the community towards a consensus.

Future Work

Generalized Staking

Panvala currently uses staking to allow access to quarterly ballots and penalize those who stake on losing slates. Staking is generally useful any time a system or its participants want to make it easy for people to earn their trust. In most existing systems, staking is governed by software, and your stake might be lost if you break the rules that the software enforces. Since Panvala's staking is governed by people, they can create any rule sets they want and enforce them by voting on whether to burn the tokens, donate the tokens, or unlock the tokens for their owner to withdraw.

We expect that one early use case for generalized staking will be certifications. Since this wave of decentralized systems is still young, it lacks much of the curation that users are accustomed to. The era of mass production brought about Consumer Reports to let people know what they could actually trust. The early internet era needed the Yahoo! Directory and the search engines that followed for people to know what was worth reading once everyone could be a publisher.

Now that anyone can write smart contracts to manage and trade users' assets, we need Panvala Mark, a Consumer Reports for smart contracts. Token holders can vote to approve certifications for contracts, require tokens to be staked, and slash those tokens if flaws in the contract are found.

Beyond certifications, generalized staking can be used whenever the token holders desire the option to sanction someone as a condition for cooperating with them.

Donation Credits

The only way to earn status in Panvala today is by sacrificing some of your wealth to make a donation. We can introduce a new option by allowing businesses to donate on behalf of their customers. Businesses often offer their customers some of the value of their loyalty in an attempt to earn it. Instead of giving customers their tenth coffee free or frequent flyer miles to use on their next trip, businesses could offer Panvala donation credits using that same value.

Businesses could acquire donation credits by making a donation directly, earning them from businesses they buy their raw materials from, or earning them when they buy from Panvala itself. Ads, sponsorships, or other opportunities offered by Panvala can give donation credits in return so Panvala donors can earn status by shopping with those businesses. In the end, all donations would be credited to individuals rather than businesses, as the actual burden of these donations (or any expense paid to attract customers) falls on individuals.

The Panvala Community

Panvala Launch Team

Since July 2017, the Panvala Launch Team has been building Panvala at ConsenSys. The team consists of Niran Babalola, Romana Basilaris, Daniel Schifano, Jacob Cantele, Akua Nti, and Isaac Kang.

ConsenSys

ConsenSys is the home of the Panvala Launch Team. ConsenSys is a global blockchain technology company building the infrastructure, applications, and practices that enable a decentralized world.

Panvala Awards Committee

The Panvala Awards Committee made recommendations for the three batches of grants we've issued while building and testing the system. Its members were Pol Lanski, Evan van Ness, Bryant Eisenbach (Batch 1), and Eric Conner (Batches 2 and 3).

Panvala Mark Council

The Panvala Mark Council was convened to demonstrate slate governance by deciding whether to recommend a Panvala Mark for a smart contract. Its members were Mark Beylin, Ameen Soleimani, Alex Chapman, Joe Urgo, Chris Smith, and Jonathan George.

You

The most important part of the Panvala community is people like you. We want to hear why you're going to donate, what's stopping you from donating, or what you'd change about the system. We built this system for you: without donors, Panvala is

lifeless code that will be abandoned. With you, Panvala is the tool that will unlock the power we've always had, and use it to fulfill the dreams we share for our future.

Vision

By Niran Babalola

In February of 2018, ConsenSys held its second company retreat in Albufeira, Portugal. That week, we played a game that was a prototype of Panvala. Instead of tokens on a blockchain, we used poker chips, and instead of smart contracts enforcing the rules, we had a gamemaster. But the core of the system was the same: tokens were issued to people who pursued our shared goals, and donors bought those tokens in the hallways of the resort so they could deposit them back into the token capacitor: a metal can full of poker chips.

What excites me most about launching Panvala is that more people will be able to feel what a few players felt that week as they played the game. There isn't a word to describe how the game makes you feel—it's not a common enough feeling to have earned a name yet. But after you feel that nameless feeling, the feeling that follows will be familiar, and it has a name we all know: that feeling is hope.

Ethereum can be—and should be—the settlement layer for the world's open financial system. While that future isn't guaranteed, I'm confident that with the right tools, this will be one of many goals our community achieves together. After all, we're the people who demonstrated that smart contracts could be usefully deployed on public networks when many doubted that it was possible. When TheDAO hack put 14% of all

Ether at risk, we're the people who endured the turmoil of the hard fork to protect the future of the network. And when everyone thought scammy ICOs were all that Ethereum was good for, we kept building through the crypto winter to make sure that the next time the public pays attention, they'll see things that used to be impossible.

I expect that Panvala will work, but that doesn't mean that it should never be changed. Panvala wasn't built to serve any ideology, or to unintentionally create a new one based on the quirks of the system we've built. Some of the system is based on assumptions and shortcuts that seem fine now, but might not be in the future. In some areas, we've intentionally made the system hard to change, not because we know better than you, but because no one wants to start playing a new game where critical rules are easy to change. I hope that these constraints serve you well, and if time and experience teach you more than we know now, that you free yourself from those constraints.

I believe (perhaps irrationally) that Panvala truly is a more effective way to coordinate voluntary cooperation than we've ever seen. If that is true, there's no way we'll stop once we've solved the problems that currently face the Ethereum community. We'll still be holding pan, so we'll still want more people to want to make donations with pan. Rather than stop after our first goal, we will expand our mission to attract more patrons, more sponsors, and more donations. From where we are today, we'll reach out to find anyone out there that might share the same goals and dreams for what we can accomplish together, and each day we'll write the story of how we solved the problems that once seemed unsolvable.

That's what the promise has been for the last technology that enabled permissionless innovation over the past few decades. Before the internet, you needed the money and connections to get AT&T's permission to launch any new idea that relied on a network of connected people. After the internet, you could be any geek off the street coding in your poorly lit dorm room while wearing your latest free t-shirt, but if people really wanted the idea in your head, you could reach the world without any gatekeeper's approval.

That's what I believe this technology will do for our economy, for our politics, and for our society. If you feel like you don't matter in today's world that is dominated by unprecedented concentrations of power, I hope Panvala will be a revelation.

You've always mattered.

Appendix A: The Myth of Panvala

In the beginning, there was a village on a river. Every aspect of life in the village depended on the river's flow of water—but in the local language, they called water "*pan*." They used pan to grow their crops, and to cook them into meals that were tasty to eat. They used pan to quench their thirst, especially while they played on hot days. They even played with pan itself: they splashed it, poured it, and threw it on each other when they least expected it.

Pan flowed down to the village from the top of a mountain. The mountain was tall: so tall that it was frozen at the top. As the ice melted, it became the start of the river. The river grew and grew as it flowed down the mountain. Downstream, clouds picked up the pan and brought it back to the mountain. Clouds dropped snow at the top of the mountain, where one day it would melt and flow into the river again.

One day, the weather changed for good. The clouds stopped coming back. Without the clouds, there was no snow at the top of the mountain. Without the snow on the mountain, there was less and less to melt for pan to flow into the river. As the river shrank, so did life in the village. Crops were harder to grow, and there was less tasty food to eat. There was less pan to drink, so on hot days, they had to stay inside. They couldn't even play very much without pan: since they couldn't splash it, pour it, or throw it on each other as often as they could before, there were fewer surprises to bring them joy.

The leader of the village was determined to solve this problem. He headed to the top of the mountain day after day to try to figure out what was wrong, but each night he headed back to the village with nothing to show for it except more frustration on his face. The villagers began to feel sorry for him. One day, a group of villagers decided to surprise him on the mountain to cheer him up. It was a hot day, so they brought lots of pan with them for the climb up the mountain.

When they got to the top, the leader of the village didn't hear them coming. A villager quietly snuck up behind him. Once he got close enough, he poured pan all over his head. Everyone began to laugh as they threw more and more pan at the leader. The pan flowed down his face, soaked his clothes, and dripped to the frozen ground at the top of the mountain. As the pan they poured hit the ground, it began to freeze, too.

That's when they realized how they could solve their problem. They could do the same thing the clouds used to. They could bring pan back to the mountain where it would freeze and eventually melt into the river below. If they could bring pan up the mountain as fast as it flowed down, the river would never dry up.

From that day forward, each villager did their part to keep the river flowing. After they grew crops, quenched their thirst, or even just played with pan, they'd collect as much pan as they could and take it back up the mountain, just like the clouds used to. Most people carried as much pan as they could fit in a backpack. The strongest villagers pulled carts loaded with pan to the top of the mountain. But everyone did as much as they could. They knew that if each villager kept going on their *panvala*—their water journey—joy would never leave the village ever again.

🗻

Appendix B: Initial Sponsorship Program

The details of the Initial Sponsorship Program will be finalized before Panvala is launched on the mainnet. To become a sponsor today, email info@panvala.com.

Appendix C: Initial Patron Program

Become a Panvala Patron today at Panvala.com. For future reference, screenshots of the donation page have been included here.

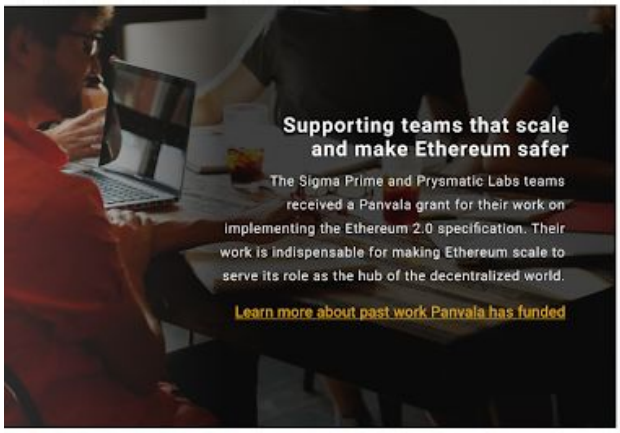
PANVALA

Blog Grants Team Contact Donate

In Panvala, donors help fund the work that the Ethereum community depends on.

Countless projects and people depend upon the Ethereum blockchain for their success. Contributing to Panvala rewards the teams who solve those problems.

Pledge Now



Supporting teams that scale and make Ethereum safer

The Sigma Prime and Prysmatic Labs teams received a Panvala grant for their work on implementing the Ethereum 2.0 specification. Their work is indispensable for making Ethereum scale to serve its role as the hub of the decentralized world.

Learn more about past work Panvala has funded

Our Founding Patrons support Ethereum

Advisor Patrons

Brooklyn Edwards	Rosa Andrews	Rene Howard

Patrons

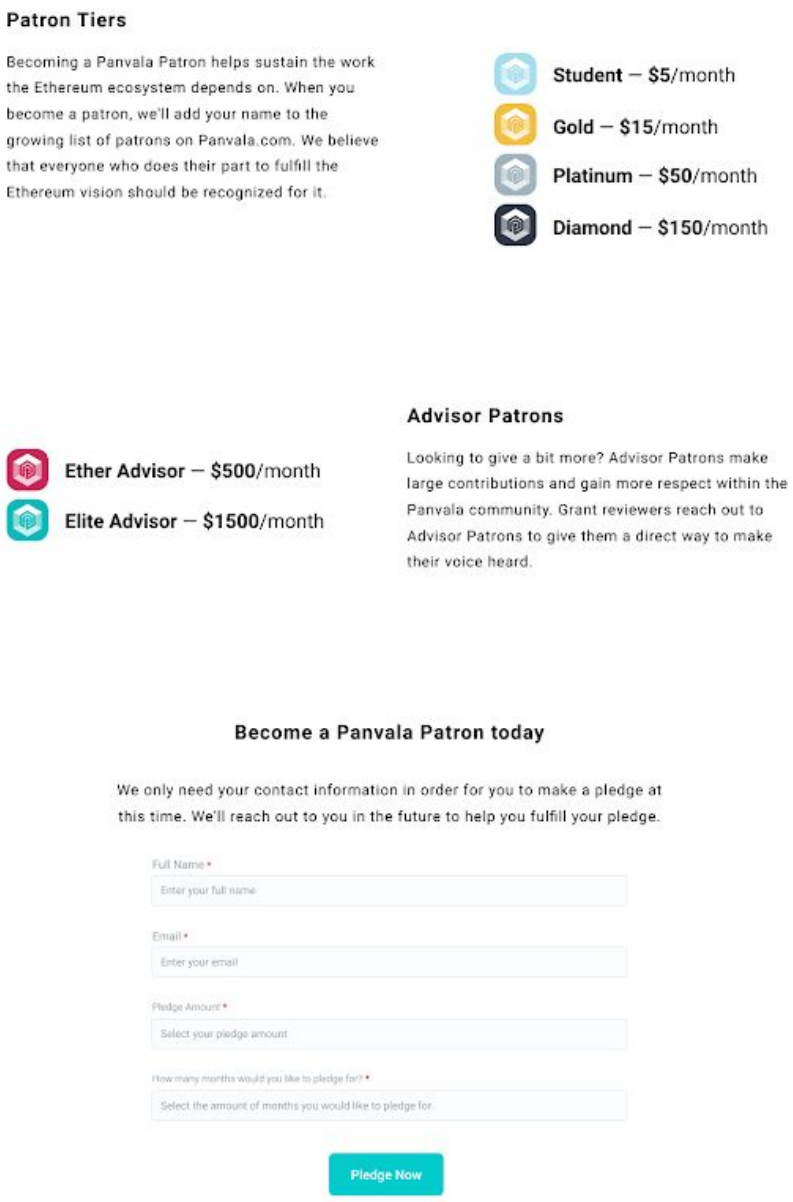
Wallace Walker	Carolyn Gardner	Perry Chapman
Kenneth Pena	Bella Hart	Brian Pena
Hector Gonzales	Kristin Smith	Mattie Perkins
Tomothy Steward	Dennis Franklin	Colleen Roberts
Sharlene Knight	Mathew Ellis	Keith Collins
Francisco Harvey	Cecil Hunt	Rita Reed
Charlie Daniels	Francis Weaver	Cecil Boyd
Melinda Roberts	Chester Hale	Jeremy Harper

View the full list of Panvala Patrons

16
Projects Funded

76
Patrons

4,149,798
In Token Grants



Appendix D: Related Work

Familiarity with the following ideas and projects will be useful for putting Panvala in context, but is not a prerequisite for understanding Panvala itself.

Fiscal Money

The 2007 financial crisis reverberated from America throughout the world, and triggered several aftershocks. In Greece, the financial crisis triggered a recession that stopped the flow of foreign capital into the country that had begun rising in earnest with its 2001 entry into the Eurozone. Its large, underreported debts became an even bigger problem with a slowing economy, and credit rating agencies downgraded Greek bonds in 2010. To avoid Greece defaulting on its bonds, the European Commission, European Central Bank, and the International Monetary Fund bailed out Greece with a loan conditioned on reductions in government spending.

After a second bailout and a worsening recession, Yanis Varoufakis outlined a Bitcoin-powered payment system to ease the crisis in 2014[12]. He criticized the Bitcoin currency as a path towards a deflationary spiral reminiscent of the Great Depression, but praised the blockchain technology that powers Bitcoin. Rather than using a blockchain to track a native, limited supply currency, this blockchain would manage "FT-coin", where "FT" stands for "Future Taxes." These coins would help Greece exit the recession imposed upon it by its creditors and the Euro by allowing the government more money to spend into the economy. Each FT-coin could be redeemed at the national treasury for its purchase price, or redeemed two years after issuance with the tax office for a premium on top of the face value. A €1000 FT-coin might be redeemable for €1500 of taxes two years later. FT-coins are removed from circulation when they are redeemed to pay taxes.

This system reflects the viewpoint of Modern Monetary Theory, a way of looking at government finance that sees taxation not as a way of funding the government, but as a way of removing money from the economy to control inflation. When you pay taxes in FT-coin, you don't directly fund the government: it will never spend the FT-coin you paid. You're just providing the demand that prevents FT-coin from losing value.

---

12 https://www.yanisvaroufakis.eu/2014/02/15/bitcoin-a-flawed-currency-blueprint-with-a-potentially-useful-application-for-the-eurozone/

In January 2015, Greece held snap elections that swept the Syriza coalition to power for the first time on a platform of tearing apart the existing bailout deal to fight for better terms. Varoufakis was appointed as the Finance Minister. Over the months of negotiations for a new bailout deal, Varoufakis prepared a plan similar to FT-coin (but using a traditional database instead of a blockchain) in case talks failed. Instead, the prime minister decided to accept the creditors terms, and Varoufakis resigned.

Dominant Assurance Contracts

Assurance contracts have been popularized by Kickstarter and Groupon. They're an agreement to make a purchase if enough other people make the same purchase. Assurance contracts are particularly useful for goods that have high fixed costs but low marginal costs, so they're only worth buying for most people if the fixed costs can be spread among enough people. Many non-rivalrous goods fit this description.

Dominant assurance contracts are a variant of that concept by Alex Tabarrok[13]. In this kind of "Kickstarter," when the goal isn't met, you are paid a specified amount. That payout is funded by an entrepreneur who bets on the goal being met. If he's right, some of the funds raised go to pay him for the risk he took.

Dominant assurance contracts avoid the wasted effort of contributing to a Kickstarter that doesn't meet its goal. If the goal isn't met, you get paid by the entrepreneur. If the goal is met, you got something you wanted to buy at a price you were willing to pay. In theory, this model gives entrepreneurs a playbook for organizing the provision of public goods while making a profit.

Moloch DAO

The unfortunately-named Moloch DAO is another effort to raise funds to spend on Ethereum development. Moloch DAO is an invite-only club that issues tokens proportionally to contributors of Ether. To "spend" Ether (ETH), tokens are issued to a worker without requiring any ETH to be deposited. The "ragequit" is key to its

[13] http://mason.gmu.edu/~atabarro/PrivateProvision.pdf

incentive model: every proposal has a waiting period for execution, and if you disagree with the proposal, you can withdraw your share of ETH before the proposal executes. When the proposal's new tokens are issued, there will be less ETH in the contract for them to withdraw.

At the time of this writing, Moloch DAO holds over $2 million worth of ETH.

Bonding Curves

Bonding curves were invented in 2017 by Simon de la Rouviere[14]. Bonding curves are tokens with an algorithmic market maker built in from the start. Tokens can always be minted at a price that increases depending on the current supply of the token. Tokens can be burned for ETH (or any other collateral) at a price that decreases based on the supply of the token. Minting tokens fills a reserve that is used to buy back tokens as needed, but if a different curve with lower prices is used to buy back tokens, the difference in prices can be used to fund the goals of the system. In the general case, bonding curves incentivize early adoption of a token and the growth of its system. When the reserve is used to fund goods, bonding curves can theoretically incentivize the growth of a fund for arbitrary goods.

Panvala does not use bonding curves, but comes from a similar line of thinking.

---

14 https://medium.com/@simondlr/tokens-2-0-curved-token-bonding-in-curation-markets-1764a2e0bee5