ParallelChain Tokenomics Whitepaper

Roderick McKinley CFA[1], ParallelChain team

Version 1.1

ABSTRACT

ParallelChain is a new layer 1 protocol based on a *delegated proof of stake* (DPoS) consensus mechanism (ParallelBFT) that employs an innovative, multi-class validator design. This approach delivers substantive performance benefits including greater throughput and faster completion times, while saving attractive censorship-resistance characteristics that are lost in other more centralised designs.

ParallelChain's tokenomics have been developed to motivate sustained, high-quality validator performance within this unique, multi-class consensus design. Good validator actions earn rewards that are funded with the collection of end-user transaction fees, and new token issuance.

- Our transaction fee mechanism is closely modelled on that adopted by Ethereum for its recent "London Fork" upgrade. We make a modest adjustment to the design to secure a stronger economic link between validator rewards claimed, and end-user value created.

- Our token issuance mechanism follows a fixed, declining rate schedule. This approach creates supply certainty; lowers inflationary pressure over time; and sets the native XPLL token upon a clear, predetermined transition path to use-driven fundamentals at network maturity.

Validator rewards funded by these two processes are distributed according to a cap and margin model. The reward caps have been designed to create decentralising actions that counter concentration of stake between nodes of the same class. Additional incentive support has also been created at the class-level, to keep stake balanced between each validator class, as a whole. This paper will first cover all these mechanisms in detail.

We shall then conduct a review of pre-minted token allocations that are being set aside for special purposes (raising capital, seeding the first nodes on the network, and funding the future development), and how vesting contracts gradually release.

After that we examine what all these factors add up to as a whole: how they shape the economics of running a validator node, and how they impact ParallelChain's token supply dynamics overall.

A NEW MULTI-CLASS CONSENSUS DESIGN

ParallelChain Mainnet's three-class validator set targets a healthy middle-ground between the full anonymity and free participation of Bitcoin and the high performance of permissioned

[1] Strategic Advisor to ParallelChain.

blockchain systems such as Hyperledger Fabric and ParallelChain Lab's own ParallelChain Enterprise.

This feat is achieved by balancing the interests of security, and decentralization using the three-class design, each of which has differing levels of seniority on the network. "Seniority", in this context, corresponds to the average voting power exercised by the nodes of each validator class, which is used to confirm or censor the transactions of the next block being committed to the chain.

The security of traditional Proof of Stake (PoS) systems hinges on the assumption that crypto-economic penalties are sufficient to deter attacks. This assumption may have held so far for the most popular public blockchains, but stands to weaken as more and more of the things we value in our social and economic lives move onto blockchain.

ParallelChain Mainnet exceeds the security guarantees of current PoS systems by requiring network participants to undergo a transparent Know-Your-Customer (KYC) process as an additional prerequisite to accession into the validator sets. Furthermore, these KYC requirements become more stringent for the more senior validator sets.

Crucially, ParallelChain Lab will not be the sole arbiter of whether a participant passes KYC. Instead, over the long run *democratic governance* mechanisms built into the ParallelChain protocol will become the ultimate authority over matters of network rules and composition, including accession into the validator set.

The most senior validator class is that of the **Governing nodes**. This class has the smallest number of nodes out of the three, but the voting power of each Governing node is designed to be larger than the nodes from other classes. Governing nodes form the network's trusted security backbone and are required to go through the most stringent KYC process as a requirement for validator set accession.

At the other end of the scale, the most junior validators are the **Beta nodes**. They have been designed to exercise the least voting power individually, but there are many more of them on the network. Membership of this validator class is almost anonymous, making validator composition more democratic.

The third validator class are the **Alpha nodes**. They lie between Governing and Beta nodes in terms of security and anonymity.

ParallelChain then uses its reward distribution design to regulate the distribution of voting power between classes, and within them. The rewards are designed to target a state where each class as a whole has an equal share of the voting power, and where each class' voting power is also equally shared between its members.

When satisfied, this property allows for a quorum of two classes to be able to overrule the other remaining, in the event of disagreement.

VALIDATOR AGENTS AND REQUIREMENTS

Two types of agents are involved in the network's validation process:

- Operators
- Delegators

Operators are the agents who control and operate the hardware and software that performs the computational work for adding transactions to new blocks and voting on the inclusion of the next block.

Each node has a single operator.

Delegators are token-holders who are participating in the consensus process by delegating their stake to a node, thereby increasing the total amount of stake and voting power exercised by that node.

There is no limit on the number of delegators. Furthermore, individuals who act as operators, may also be delegators of other nodes at the same time.

In order to operate a node, the operator has to stake a determined minimum number of ParallelChain Token (XPLL) into the node that they operate. This minimum requirement varies according to node class in the following manner:

- Governing class: 2.5m XPLL tokens
- Alpha class: 1m XPLL tokens
- Beta class: 0.5m XPLL tokens

This design has been chosen in order to:

1. Seed the relative differences in voting power between node classes.
2. Increase the quantity of stake (and value) that's at risk for bad operator behaviour.
3. Create a cost that defends against the creation of multiple node identities by a single actor.

Within this paper, both operators and delegators are considered "validators".

Accordingly, "validator rewards" refer to rewards that are distributed to compensate all agents that are participating in network validation, whether as operators or delegators.

As we shall see, however, operators do get privileged claims to validator rewards, in comparison to delegators. This is designed to compensate operators for the extra cost, risk and responsibility of their role.

FUNDING OF REWARDS

ParallelChain compensates validators for their work by means of a "rewards" distribution. Naturally, these funds have to come from somewhere. In ParallelChain's design, rewards are completely funded from two sources:

- Transaction fees
- Token issuance

The supply of tokens created by these two processes are determined by set rules that have been chosen to meet tokenomic objectives. We explain these rules below.

TRANSACTION FEE DESIGN

ParallelChain's transaction fee mechanism closely follows that used by Ethereum following its recent "London Fork" upgrade (which implemented changes detailed in proposal EIP 1599).

The simple argument to recommend this choice, is to claim that ParallelChain will benefit from the use of a tried and tested model. In a new and experimental space, the assurance this provides should not be discounted.

However, we did also conduct a deeper review of EIP 1559's properties, and concluded that they were attractive and worthwhile replicating. In particular, we value EIP 1559's capabilities to:

- Select transactions according to their economic efficiency.
- Generate price signals that respond to end-user demand.
- Filter erratic noise from these price signals.
- Facilitate more accurate price expectations for end users.
- Reduce instances of overpayment for services by end users.
- Link price dynamics to measures of network optimality.
- Integrate validator incentives with end-user value creation.

We review the key features of EIP 1559 that we are implementing for ParallelChain below:

Transaction requests and selection

1. Users requesting a transaction are required to submit two numeric values:
    i. The maximum price per unit of computational work that they are willing to pay for their requested transaction.
    ii. A premium (or "tip") that they are willing to pay over a protocol-calculated "base fee" that is charged on all transactions included in the next block (both the premium and base fee are also priced in the units of computational work).

2. The user's transaction fee is calculated by taking the minimum of:
    i. The base fee, plus the premium.
    ii. The maximum submitted price.

3. The process determining which transactions are included in the next block is as follows:
    i. If the transaction fee is lower than the base fee, the transaction is automatically excluded from the next block.
    ii. All other transaction requests are ranked by the magnitude of their transaction fees.
    iii. All requests with fees that have a rank greater than or equal to the number of positions available in the next block, are selected (and therefore, executed).

Base fee calculation

The transaction fee calculations described above include a base fee term that is itself calculated by the network protocol.

To determine how this base fee is adjusted, the size of the last block is compared to a target block size that is associated with optimal chain performance. This target block size has been set at half the maximum block size that the system can handle without compromising performance.

A maximum magnitude of change to this base fee (up or down) is set at 12.5%. This value is then multiplied by the relative distance of the current block size above (+) or below (-) the target block size.

This design relates the base fee to an objective measure of system capacity that trades off against the level of demand served by each block. Updates to this value are thereby linked proportionately to the network's current state as it relates to its optimal performance level.

An important outcome of this design is a smoothing out of price movements, which avoids users facing erratic price surges.

At the same time, if high demand is sustained, this does eventually transmit into the pricing process. When this occurs, the higher base fees help users avoid overpaying or being subject to block exclusions that result arbitrarily from erratic price movements (both which create a recurring poor end-user experience).

Burning

The main difference between EIP 1559 and ParallelChain's transaction fee model is not found in the way fees are calculated, but in how they are used.

Ethereum have decided that the base fee should be burnt unconditionally, for every block produced. There may have been grounds to recommend this choice specific to their case, but we do not think it offers good fit for us.

Base-fees account for large sum of value that has been generated from end-service use. Burning them redistributes this value diffusely, in an undifferentiated way among all ETH holders. Alternatively, you could use these funds to:

- Motivate better operators to join the network.
- Provide funds that help existing operators invest in better equipment upgrades.
- Fund improvements to ParallelChain itself.

The opportunity cost of foregoing these actions in favour of a diffuse value redistribution is too great for any new up-and-coming blockchain. That is why we are taking a different approach.

In ParallelChain transaction fee streams will fund two general uses:

- **Treasury funding**: 20% of transaction fee volume will be collected by treasury to fund future improvements to the network.
- **Reward funding**: the remaining 80% goes to fund part of the reward pool available to validators.

We think that it is valuable to create a burn mechanic to regulate inflationary pressure, but this operates further downstream in our design and, critically, is contingent on operator performance.

The effect of this approach is that we send a message to operators saying: "Look, we won't burn a big chunk of your fees like Ethereum does – but you still have to perform to earn the full share!". In this way, the burning mechanics of our design, function as an incentivising instrument as well as a mechanism for broad value redistribution.

The burning mechanism will be explained in greater detail later on, in the section that explains how rewards are calculated and distributed.

TOKEN ISSUANCE DESIGN

Funding positive incentives with issuance is a common design feature in layer 1 protocols. The approach often finds theoretical justification by appeal to some variation of the infant industry argument, or as a way of internalising higher network effect contributions made by early adopters.

Bitcoin set the exemplary precedent for an issuance-driven rewards system. Its deterministic issuance design creates long-term certainty about token supply, and limits this supply at a terminal upper bound.

We agree with the broad outline of this approach, but differ on the view that there ought to be a fixed terminal supply.

We think it is better to have a low, fixed rate of issuance in the long run. This helps keep transactions flowing and prevents agents leaving the system. Those departures would otherwise risk a downward spiral of value-depletion setting in.[2]

Another feature of this design worth noting is that it is agnostic to staking levels on the network. That is a given since Bitcoin is uses *proof of work* to arrive at consensus, but other DPoS layer 1 protocols have chosen linked their issuance rates to staking levels (e.g. Avalanche, Cosmos).

We have decided not to follow that approach, and keep staking levels independent and managed "by the market".

[2] If the high fees needed to run a zero-issuance network make users to leave for cheaper alternatives, these departures represent a loss of value tied to a reduced network size. The supply of transaction fees available to operators might also be reduced. This leads to operators charging higher rates to compensate which, in turn, prompts additional flight and loss of value from the network.

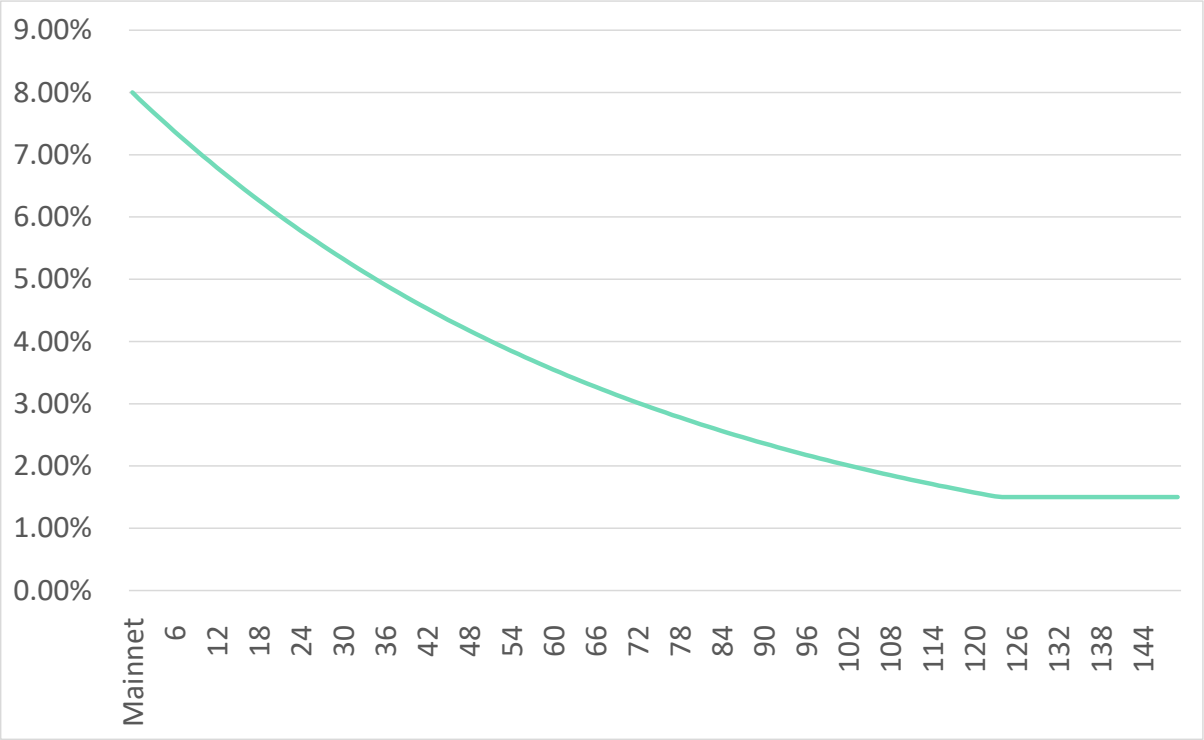
In summary, we have opted for a deterministic, declining issuance rate schedule that levels off at a low, fixed long-term rate, which is agnostic to network staking levels.

Solana has proposed an elegant parameterisation for this style of issuance design, that we have chosen to follow. It's defined by:

- An initial issuance rate of 8.0% per annum
- An annual rate decrease of 15.0% per annum
- A long-term issuance rate of 1.5% per annum

Given the parameter values shown, the issuance rate declines to reach the long-term rate of issuance is reached in just over 10 years.

Scheduled issuance rate curve



To determine the issuance minted for each block, the protocol calls a function that reads in the time elapsed since the Mainnet launch. This returns the issuance rate that applies for that block.

While the rate is deterministic, the actual volume of tokens minted cannot be determined upfront, because the token base referenced in the calculation is affected by the non-deterministic behaviours of agents on the network, which may trigger token burns (discussed in the reward distribution section below).

That said, the performance-linked terms we stipulate below for token burning, give us confidence that these non-deterministic factors will not drive large variations in burn volumes. Net issuance rates and changes to circulating supply should be relatively predictable as a result.

The basis used for the calculation of new issuance will be the *total current supply* (TCS).

This concept is defined to be the sum of all tokens that exist on the network, whether they are locked by a vesting contract, or unlocked. At Mainnet launch, therefore, TCS has a value of 250 million.

Owing to the process of issuance itself, and token burning exercised by the protocol, TCS will evolve dynamically over time. TCS for the current block, is given by the TCS value of the previous block, plus the amount of issuance generated for that block, less the amount burned for that block.

Unlike transaction fees, issuance will not be used to fund ParallelChain's treasury. While this approach has been used by other layer 1 protocols, we think it risks being regarded as overly self-serving given that issuance is created unconditionally by the network protocol.

By restricting treasury funding to transaction fees, we create a clearer, stronger link between treasury funding and the amount of value that ParallelChain creates for its end users.

It should be noted that there is no pre-minted issuance used to fund validator rewards. All rewards will be minted on the fly, block by block, and this process will only commence at the Mainnet launch date, which is expected to occur some months after TGE.

REWARD DISTRIBUTION

Motivations

The transaction fee collection and issuance processes described above, generate a supply of tokens that contribute to the funding of rewards that are paid to validators (operators and delegators) on the ParallelChain Mainnet.

The next layer of tokenomic design that we are now turning to, controls how these rewards are distributed to validators.

These matters need considerable care and attention.

We have to preserve the positive link between receipt of rewards with actual contributions being made to the network. The design also has to be free of loopholes that would allow operators to "cheat" and receive the reward without the commensurate work. In addition, we need to avoid creating adverse incentives that make it easier for operators to earn rewards by attacking the performance of other operators, or taking other actions that similarly reduce the performance and security of the network.

These foregoing concerns are of essential value. Without them, we lower the efficiency with which this system performs its work or, at worst, risk that unravelling altogether.

We also have another important (albeit contingent), concern to address with our distribution design: How do we regulate or maintain a desirable degree of decentralisation in a DPoS blockchain?

This is the so called “concentration of stake” problem.

It’s based on the observation that the reward dynamics in some DPoS blockchains enable (or even create a tendency towards) ever growing concentrations of stake among a smaller set of operators.

These outcomes erode the attack resilience of that network. If stake is concentrated in fewer hands, it makes it easier to these actors to conspire together to control ledger entries for non-consensual ends.

ParallelChain has opted to accommodate a degree of centralisation in its design to improve performance, but no more than is needed. Within our multi-class framework, we want our consensus protocol to be as distributed as possible. We have added elements to our reward distribution model that are designed to counteract excessive concentrations of stake.

Having stated our key motivations and concerns, we can now introduce the outline of our reward distribution model with six high-level steps:

1. Calculate the total reward funds available for the block.
2. Adjust this quantity based on global availability scores.
3. Calculate the reward available to each validator pool.
4. Adjust this quantity based on that pool operator’s availability score.
5. Calculate the reward share of the pool’s operator.
6. Calculate the reward share of the pool’s delegators.

We now look at the details involved for each step, following the same headings, below.

1. Calculate the total reward funds available for the block

This is given by:
(1- share of transaction fees to treasury) * block transaction fees + block issuance

2. Adjust this quantity based on global availability scores

We chose to include a global performance incentive within the reward mechanic.

The objective here is to create conditions where how much individual validator pools earn also depends on how the other pools perform. This feature creates a personal cost to operators attacking others.

At present, operator availability, is the only measure we could find which would not be adversely manipulated by operators for gain. To the extent that operators have control over their own availability, increasing it always benefits the network.

Roughly speaking, an operator is ‘highly available’ during a particular epoch if its signatures is included in a large proportion of the epoch's blocks. The converse is true if its signatures are not included in a large proportion of the epoch’s blocks.

Having selected availability as our performance measure, the simplest measure of collective performance that meets our needs is the *average availability* of all operators on the network over a protocol-defined period we term an 'epoch'. We shall refer to this as the *global availability*.

Put simply, we want lower global availability to generate a penalty in the form of reduced rewards for every validator in the system (delegators also are exposed to this markdown).

We can target and sharpen the disincentive resulting from collective poor performance, by setting an arbitrary interval which defines the value at which the disincentive first starts to apply, and the value at which rewards have been maximally reduced to zero.

Mathematically this adjustment factor is calculated as:

= 1 when global availability >= upper bound
= 0 when global availability <= lower bound
= 1- (block global availability – lower bound)/(upper bound – lower bound) otherwise

Whatever sum of tokens is not passed on to the next reward calculation step, is burned.

3. Calculate the reward available to each validator pool

Introducing the Cap and Margin Model

The simple and intuitive way to determine the share of the rewards left over at the end of step 2, would be to divide these up in proportion to the share of total stake held by that validator pool (of all the stake on the network for that block).

This approach however, fails to defend against concentration of stake occurring.[3]

To achieve this result, we have taken inspiration from Cardano's cap and margin reward distribution model.

Fortunately for us, our use of minimum operator stake requirements for each validator pool removes considerable complexity from the calculation the Cardano implements for their reward distribution.

That leaves a simple formula for calculating a given pool's share of rewards. It is just the minimum of its actual share of network stake, or a capped maximum share of rewards that the pool is not able to claim beyond, regardless of how much stake they actually have.

So, more formally:
pool reward share = rewards after step 2 * minimum of (pool's share of network stake, pool cap)

The Cardano team set the pool cap as being 1 divided by the desired number of validator pools in the network.

[3] Note, that there is no obvious reason to think that it would lead to a concentration of stake either, it is just to observe that there's nothing to stop that happening either.

What this means in practice, is that once a pool has more stake than the cap (what is termed an “oversaturated” state), it no longer earns additional rewards when more delegators stake in.

When a pool reaches this state, the only way operators can pass the same returns to delegators and preserve their profits while attracting more stake, is by reducing their costs (which is hard to do).

In these circumstances, undersaturated pools become more economically attractive to delegated stake. This induces a rebalancing of stake that tends towards the levels set by the cap parameter.

Adjusting Cap and Margin to Three Validator Classes

Our multi-class validator design creates some complexity that needs to be catered to.

If we apply the same cap structure to all validators, regardless of class, the Governing nodes will not be able to accumulate the larger voting power intended by our design.

To avoid making arbitrary adjustments to the voting power of stake based on class, we have decided to calculate different cap levels for each validator class.

Recall that one decentralising objective is defined at class-level. We want a quorum of validators in the other two classes to be able to overturn a consensus of all validators belonging to the other remaining class.

This means we want the share of stake contained in each class to be 1/3 (~33%).

To work out the cap to apply to each node in each class, all we need do is divide the 33% by the number of nodes in that class.

What this cap represents, is the share of total stake that each node would have if stake were maximally distributed within that class, when that class is itself maximally distributed with respect to the other two classes, holding 33% of the network’s stake.

So in summary, the reward share claimable by any validator pool will be the minimum of the value of their actual share of network stake, and the value of their cap that applies the class of their pool.

This model admits of a theoretical possibility where all validator pools are perfectly balanced, and therefore unable to reward new stake that wishes to participate in the validation economy.

We do not think this would ever obtain. Random underlying factors would always disturb this equilibrium (if it ever had a chance to arrive!).

Nevertheless, we have included a tolerance factor that ensures that a small, marginal increase in stake always pays, which creates a little extra space for new stake to be entered into validation service.

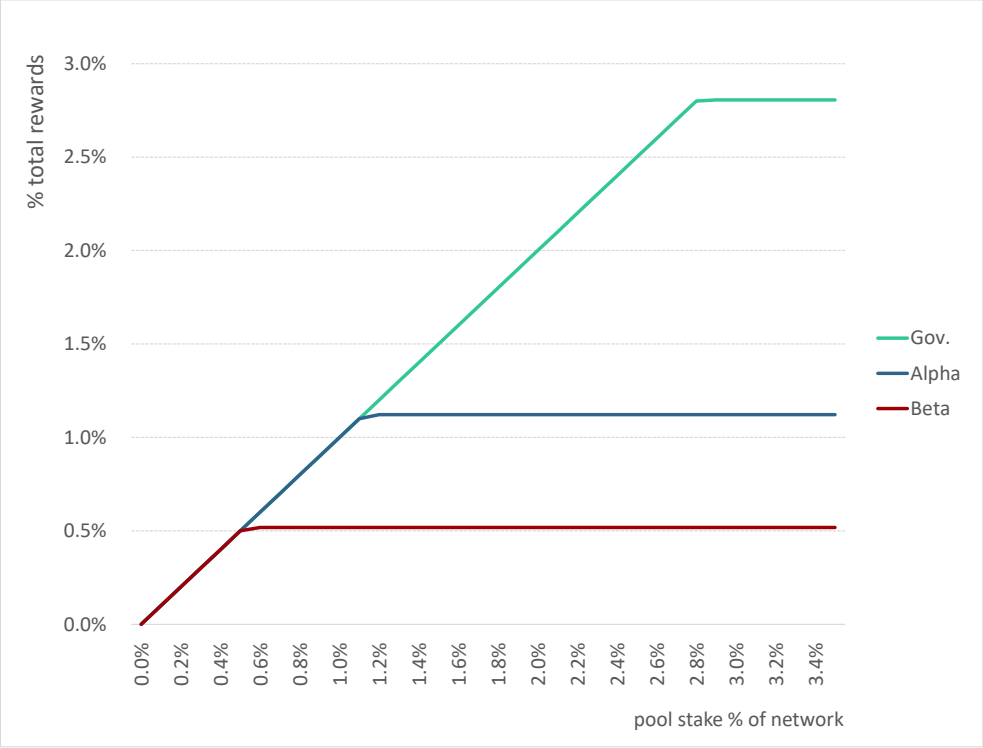
Altogether this means that the effective caps for each validator, according to their class are:

- Governing: (1 + tolerance) * 33%/number of Governing validators
- Alpha: (1 + tolerance) * 33%/number of Alpha validators
- Beta: (1 + tolerance) * 33%/ number of Beta validators

The chart below visualises these relationships under the assumption that all node token sale investors exercise their option to become node operators. This would mean that at Mainnet launch we have:

- 12 Governing nodes
- 30 Alpha nodes
- 65 Beta nodes

Share of total rewards vs. pool's share of network stake (by node class)



An Extra Incentive to Support Class-Level Balancing

The cap calculations described above are determined at the individual validator level. They reference the current level of stake in the validator pool and the target level of stake for that class, but they do not have any reference to the current share of network stake actively being held at class level.

We therefore decided to add an extra contingent incentive that helps keep the share of stake at class level close to the target 33% value.

This incentive only acts on the pools that are already oversaturated with respect to their individual caps, but only if that validator pool's class as a whole is oversaturated.

Here again, we specify some tolerance for variation above the 33% mark that we are prepared to let the system have without activating this disincentive.

As before, the effect of this incentive is concentrated over a specific numeric interval that maximally reduces rewards to zero if the class share of stake is above some designated threshold.

So, formally, the adjustment factor is:
=1 if class share <= class target + tolerance OR pool share <= pool-level cap
= 0 if class share >= upper bound
=1 – ((class share – lower threshold)/(upper threshold – lower threshold) otherwise

Whatever sum of tokens is not passed on to the next reward calculation step, is burned.

4. Adjust this quantity based on that pool operator's availability score

Calculations and reasoning from this point on are more straight forward. In this step we now apply an incentive that confers a consequence to individual performance. Note that this adjustment is additive over the earlier adjustments made for the performance of the *group*.

Availability was chosen again, due to the fact that other alternative measures of system performance are prone to manipulation, while availability is not.

The calculation here is simple: Take whatever reward is outstanding after step 3 and multiply that by that validator pool's availability. Whatever sum of tokens is not passed on to the next reward calculation step, is burned.

5. Calculate the reward share of the pool's operator

The simple way to determine the operator's share of that pool's rewards is to pro-rate it on the share of tokens the operator themselves has committed to that same pool (including the minimum operating stake).

The operator of this validator pool has assumed additional risk and responsibility to run and maintain this node, and it is fair for them to claim a fee for this.

ParallelChain's design allows operators to declare a fee (a "margin") that's charged on the entire sum of tokens that they have in their pool. Delegators see this fee before deciding where to delegate stake.

The share of rewards they claim in proportion to their own stake, is actually calculated on the sum of rewards left over *after* this margin fee has been removed (as is done for the other delegators below).

Mathematically, operator rewards are calculated by:
(margin % * rewards in 4) + (1 – margin%) * rewards in 4 * operator's share of pool's stake

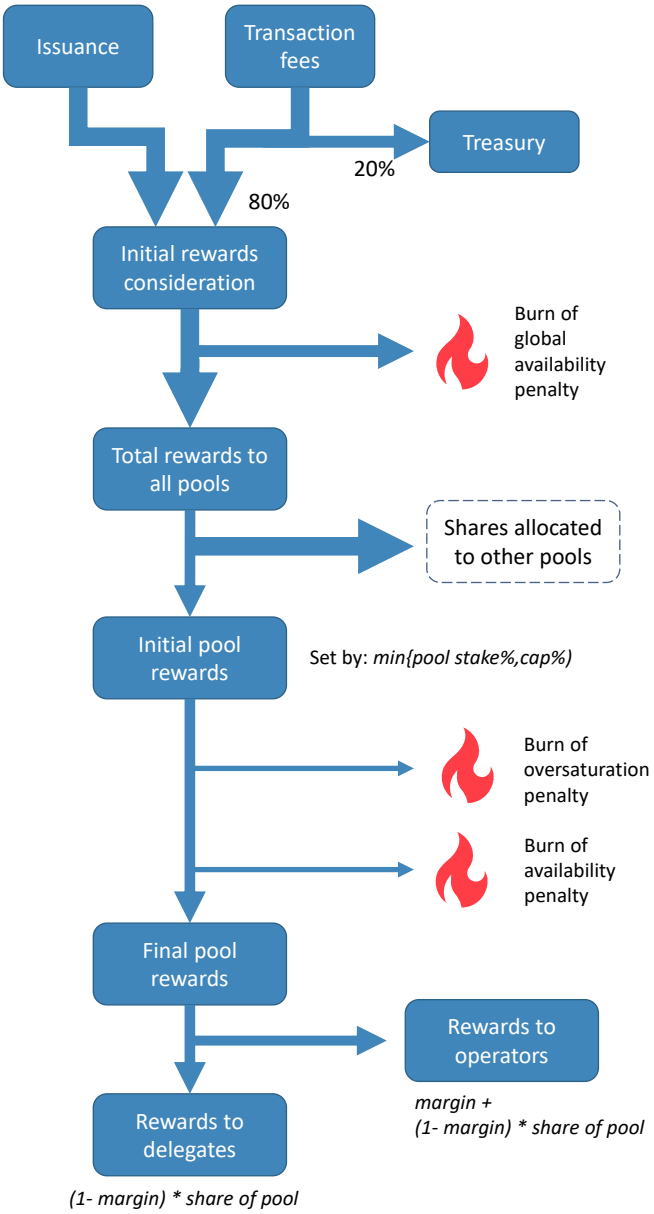
6. Calculate the reward share of the pool's delegators

The final calculation of the share of rewards received by delegators is simple and intuitive. It is simply proportional to their share of that pool's stake, on the amount of rewards remaining after step 4, and the removal of the operator's margin in step 5.

More formally:
(1 – margin%) * rewards in 4 * delegator's share of the pool's stake

A VISUAL ILLUSTRATION OF THE REWARD DISTRIBUTION MECHANISM

To help summarise the flows managed by this reward distribution process, we have composed the flow chart provided below. It begins with the issuance and transaction fee collection processes at the top, and terminates with the rewards distributed to the operator and delegators of one specific validator pool on the network.



SLASHING MODEL

In addition to the reward funding and distribution mechanics detailed above, specific penalties need to be created that punish other bad actions on the network that are not referenced by the incentives described thus far.

This is done using a slashing mechanism that removes a portion of tokens from the pool of the offending party, thereby taking those tokens out of the control of their previous owners. Crucially, this slash is also applied to delegated tokens. This creates an incentive for delegates not to delegate stake to bad actors on the network.

Slashing conditions include:

1. Signing a block that 'conflicts' with a given chain.
2. Having one's individual availability over an epoch fall below some threshold.
3. Other severe infractions that the democratic Governing mechanism deems punishable, including censoring transactions and attempting long-range attacks.

A small proportion of tokens slashed due to condition 1 is awarded to the network participant (the 'whistle-blower') who provides the evidence that an operator signed a conflicting block. The rest of the tokens are burned to make whistle-blower front-running economically unviable. Tokens slashed due to conditions 2 and 3 are burned in totality.

TOKEN SALES AND ALLOCATIONS

ParallelChain will pre-mint 250 million XPLL tokens (the *initial total token supply*) which will be used to fund fixed allocations that are set aside for pre-specified uses.

These uses fall into one of three general categories:

1. Raising startup capital
2. Seeding node operators on the network
3. Funding operations and future growth

Following the approach taken other DPoS project fundraises, ParallelChain is going to use the sale of node operator rights as a fundraising instrument – the Node Round.

More uniquely to ParallelChain's offering, the Node Round does not obligate purchasers to become node operators. What being sold instead, is the *option* of becoming a node operator, along with the required minimum number of tokens that they would need to do so.

Should the purchasers choose not to exercise their option to become operators by a set exercise date (currently set for 6 weeks before Mainnet launch), the tokens they have bought will default to a typical vesting contract, which unlocks token over time.

The terms of the Node Round actually provide purchasers with a second embedded option, which allows them to exercise the right to receive tokens according to those default vesting terms, at any time prior to the expiry of the operator option. Put simply, purchasers will not have to wait until 6 weeks before Mainnet launch before they start receiving vesting tokens, if that is what they wish to do.

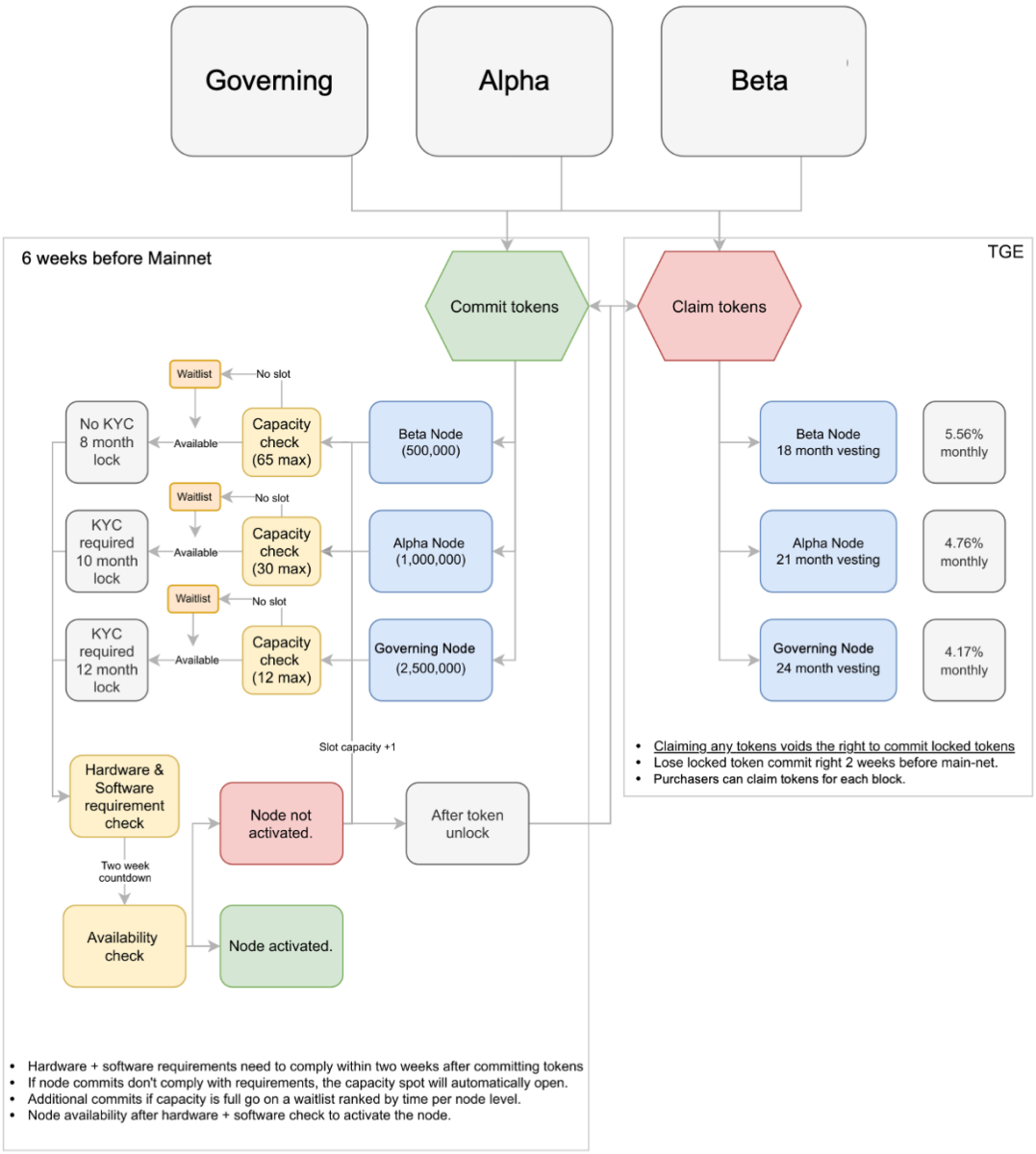
Specifically, if purchasers choose to exercise their node operator option, assuming there is a 6-month period between TGE and the launch of Mainnet, this subjects the tokens they have bought to total lockup periods of:

- 18 months for Governing nodes
- 16 months for Alpha nodes
- 14 months for Beta nodes

Conversely the default vesting terms (whether exercised early, or defaulted to after the operator option expires) are:

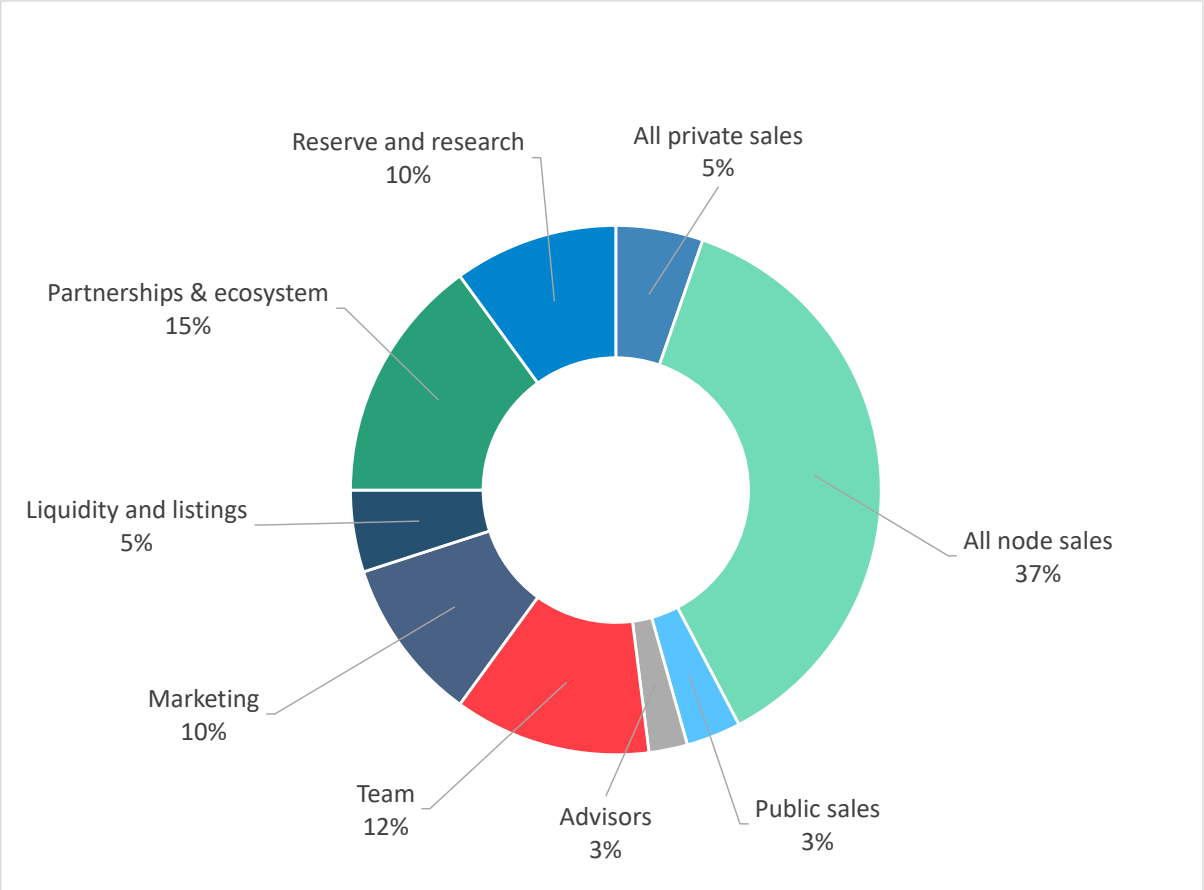
- 24 months vesting beginning day 1 after TGE, for Governing nodes
- 21 months vesting beginning day 1 after TGE, for Alpha nodes
- 18 months vesting beginning day 1 after TGE, for Beta nodes

These conditions are summarised, along with additional details, in the flow diagram below.

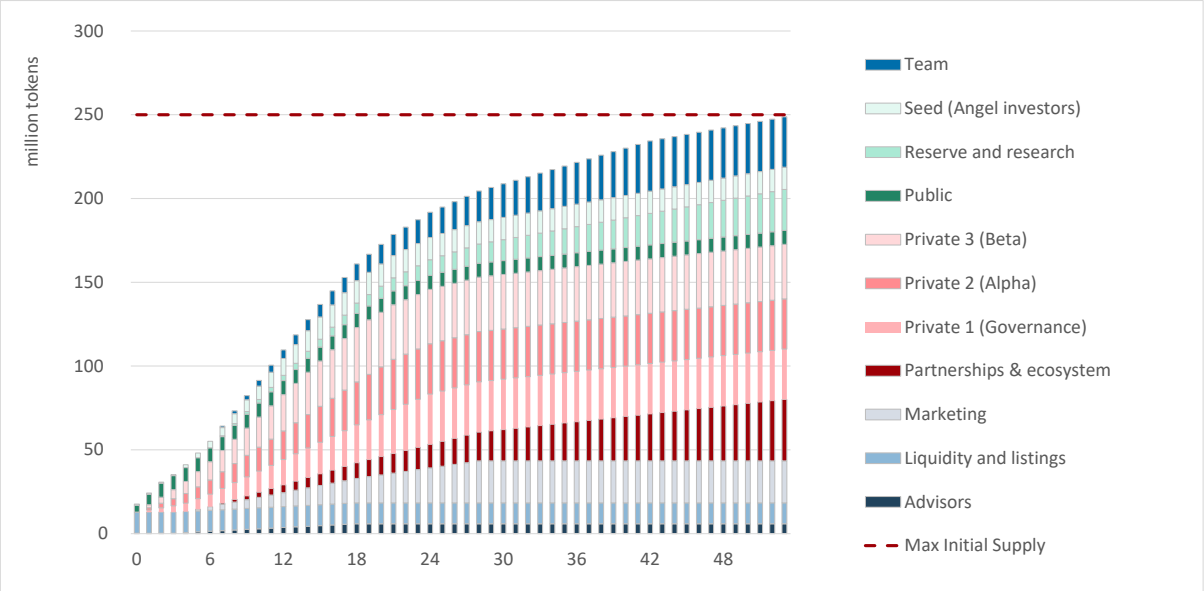


Having explained the terms of the node sales in some detail, let's now review them alongside the other token allocations.

Distribution of allocations as shares of initial total supply



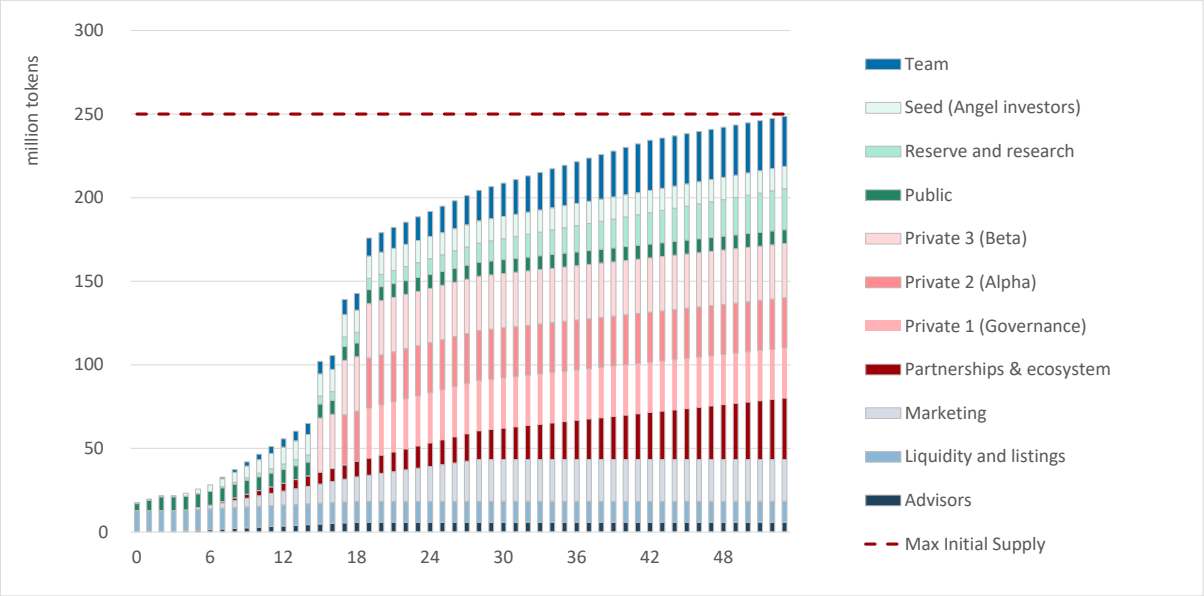
Emissions from vesting schedules governing each allocation (no nodes activated)



Notice however, that the evolution of supply shown, assumes that no Node Round purchasers choose to become operators, and so represents a limit case for token supply.

As we have seen exercising the option to become an operator subjects node sale tokens to longer lockup terms. This has the effect of backshifting the liquid supply schedule. The chart below illustrates the outcome in the opposite limit case that all nodes choose to become operators.

Emissions from vesting schedules governing each allocation (all nodes activated)



Note that the sharp cliff shown is unlikely to result in practice, since we have no reason to believe that all nodes would exit their roles at the same time, as soon their lock periods expire.

TOTAL SUPPLY DYNAMICS INCLUDING ISSUANCE AND BURNING

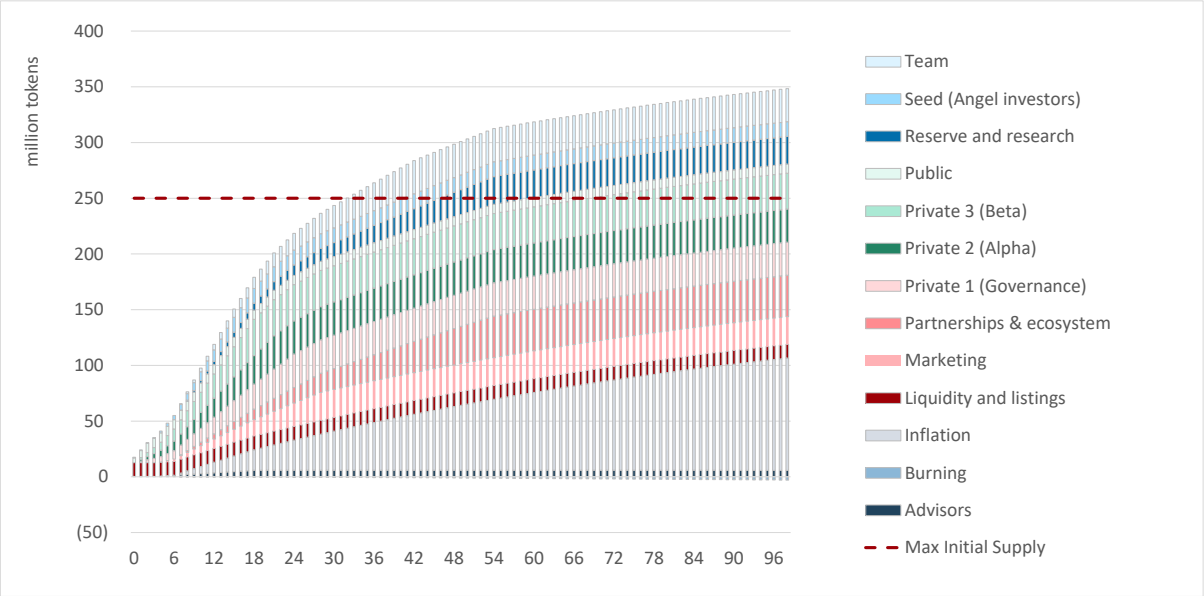
The emissions charts above only represent a partial account of the overall token supply dynamics on the network. As discussed earlier, there are also contributing issuance and burning processes that have impacts on the total token supply.

These are also non-deterministic processes, and that means that the impact of issuance and burning on circulating supply can only be generated as a provisional forecast.

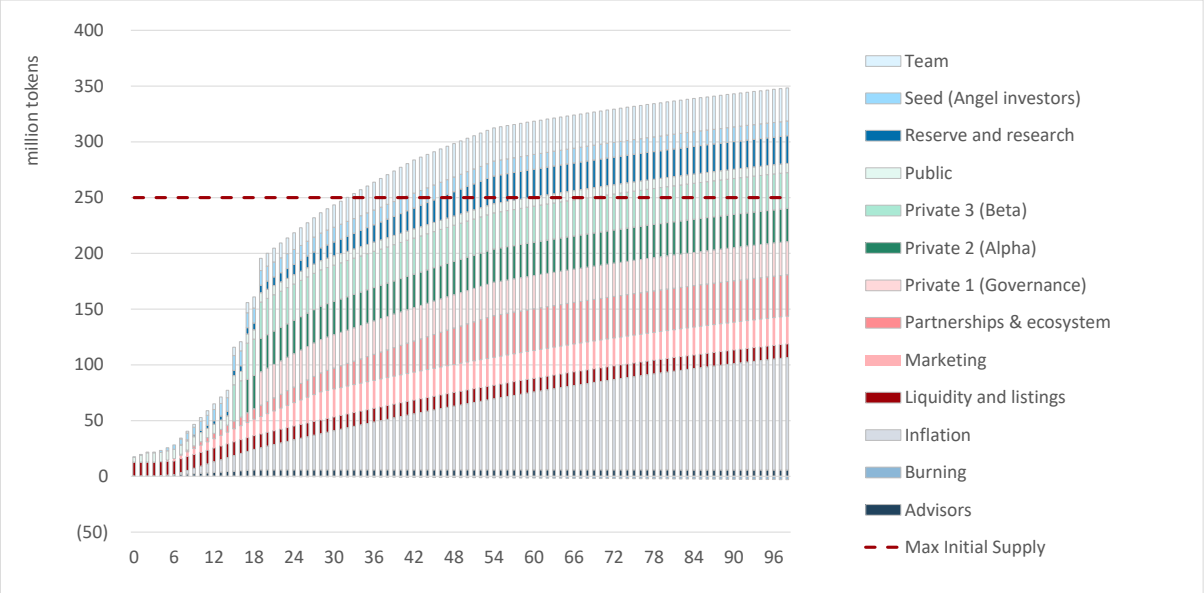
Given that we anticipate high availability performance from our operators, burn rates remain systematic but moderate. Under such circumstances, we can expect the increase to total supply to be mainly governed by the scheduled issuance rate.

This outcome of this forecast is illustrated in the charts below for the zero node and 100% node cases described earlier.

Estimated circulating supply after issuance and burning (no nodes activated)



Estimated circulating supply after issuance and burning (all nodes activated)



DELEGATOR ECONOMICS

Now that we have described the funding and distribution of validation incentives and explained some facts about the token sale, we have all these facts we need to evaluate the economics of becoming a validator on ParallelChain's network. Of course, the economic proposal differs for delegators and node operators. So, we will examine each in turn, beginning here, with the delegator case.

The net economic benefit that delegators receive is captured by a measure of yield that they receive from their delegated tokens.

What then, are the main drivers that determine this yield?

The first drivers that may come to mind, are the volumes of issuance and transaction fees collected, and how these change over time. This is intuitive to grasp, since it is the token flows generated from these processes that set the total sum of rewards available for each block.

The next important driver to delegator yield is the number of delegators on the network. The more delegators there are, the smaller the share of fixed rewards that each of them is able to receive.

The measure we chose to work with for this analysis has been termed the *delegate staking rate*. We define this as:

(sum of staked tokens – sum minimum operator stakes) / sum of unlocked tokens on the network [4]

So, in summary, the higher the delegate staking rate, the lower the expected yield received by delegators.

The third factor determining delegator rewards are the number of nodes on the network. Node operation requires a substantial minimum stake commitment and, all else equal, a greater number of nodes will require higher delegation rates in order to defend the share of stake that accrues to delegates.[5]

We illustrate the impacts of these various factors on yield, in the charts below.

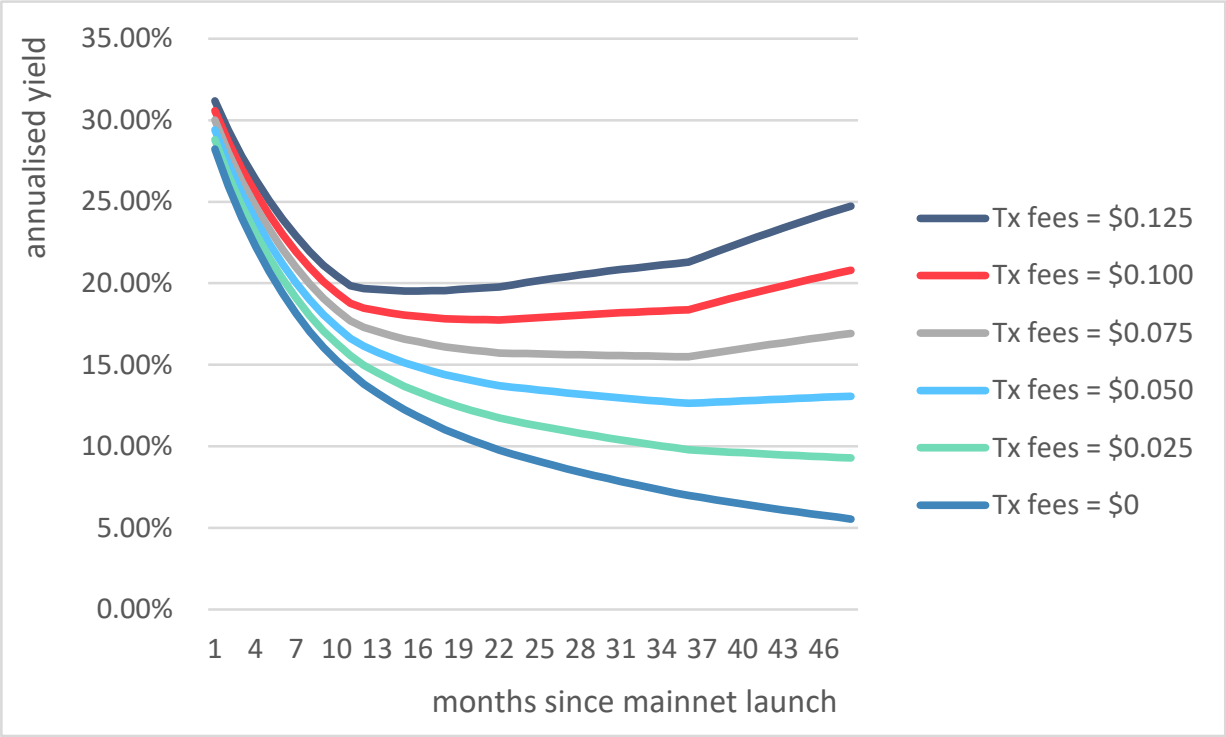
In the first chart we consider the case with a delegate staking rate of 50%, and 6 Governing nodes, 15 Alpha nodes, and 33 Beta nodes being active on the network, and a token price of US$1 per XPLL.

Holding these values constant, the chart graphs delegator yields over time, and explores how they vary with respect to different transaction fee prices.

[4] This definition has been chosen so that we exclude locked tokens that cannot be staked and exclude tokens that have to be staked by node operators to satisfy the minimum criteria needed to carry out their roles.

[5] This is essentially a different source of staking competition, but otherwise has nearly identical effects to the reward dilution that's created from a higher delegate staking rate.

Delegator yield (in tokens) over time, charted at different transaction fee levels



The zero transaction fee case is included in this chart to show what the "baseline" issuance-only yield is, and to provide a reference point for how significantly yields change at various transaction price levels.

Due to the auction mechanics used to set the final transaction fee price, we cannot be certain what these prices will be. However, this analysis indicates that if transaction costs are in the super low range achieved on Solana (~$0.00025 per transaction), then delegator yield dynamics are dominantly driven by the declining levels of issuance being generated on the network.

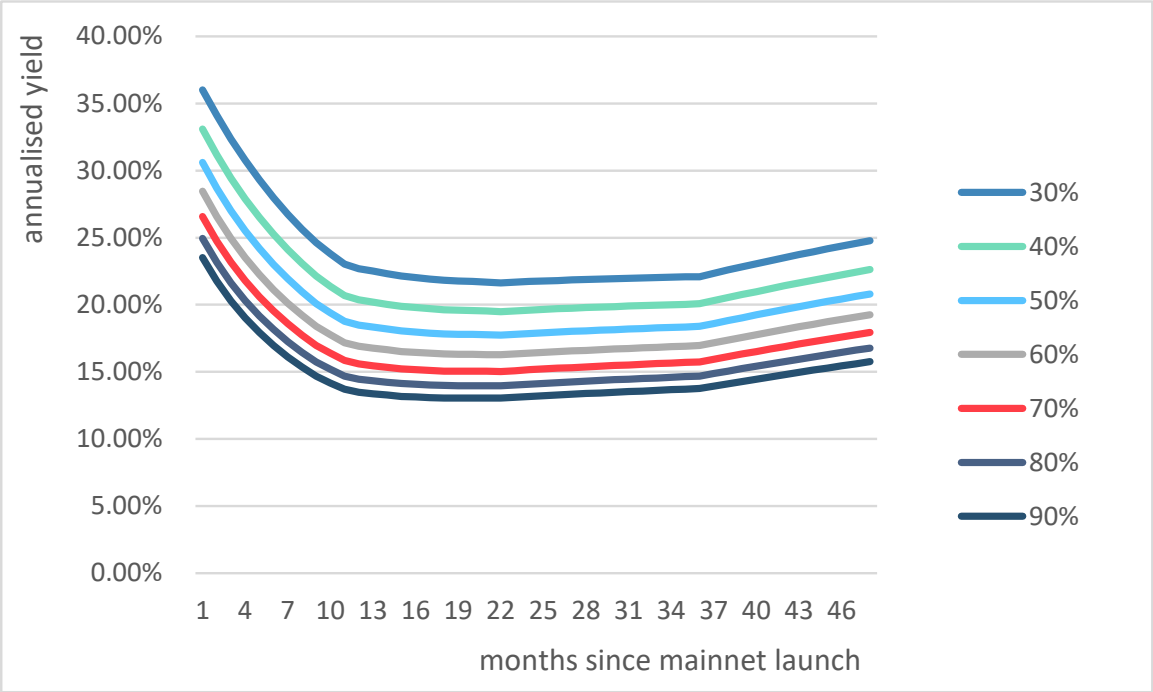
Alternatively, in cases where transaction fees increase to moderate levels (that are still very low by Ethereum's standards), growth in network transaction volume may eventually lead them to become the dominant contributor to delegator yield.[6]

This time we will examine how the delegator staking rate impacts yields.

We'll do so for a transaction fee case of $0.1, with all other assumptions remaining the same.

[6] The chart assumes that there are 50,000 transactions per day at mainnet launch, rising to 1,200,000 by year four. Sensitising yield to transaction volume is analogous to sensitising yield on transaction price, so this analysis has been excluded for the sake of brevity.

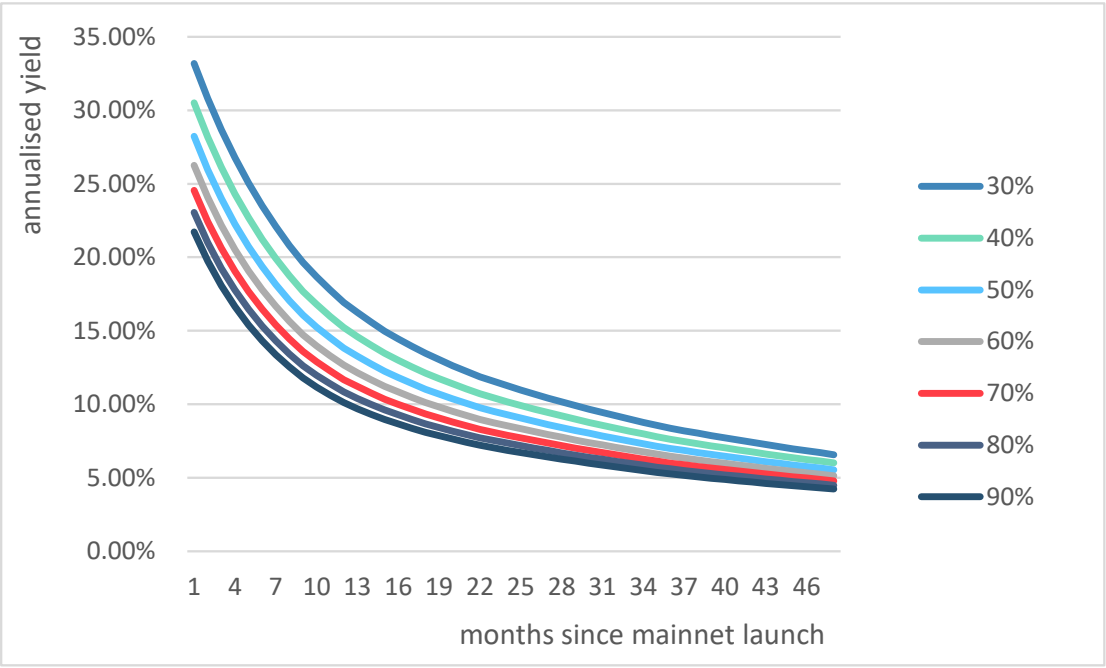
Delegator yield (in tokens) over time, charted for different delegate staking rates (tx = $0.1)



We see clear and substantial changes in yield being driven by the rate of delegate participation on the network. We do however find an interaction between the transaction prices assumed, and the sensitivity of yield to the delegate staking rate.

The chart below plots the same curves, but this time for the zero transaction fee limit case. And we see that the difference in yield that's driven by the delegate staking rate becomes much smaller, over time.

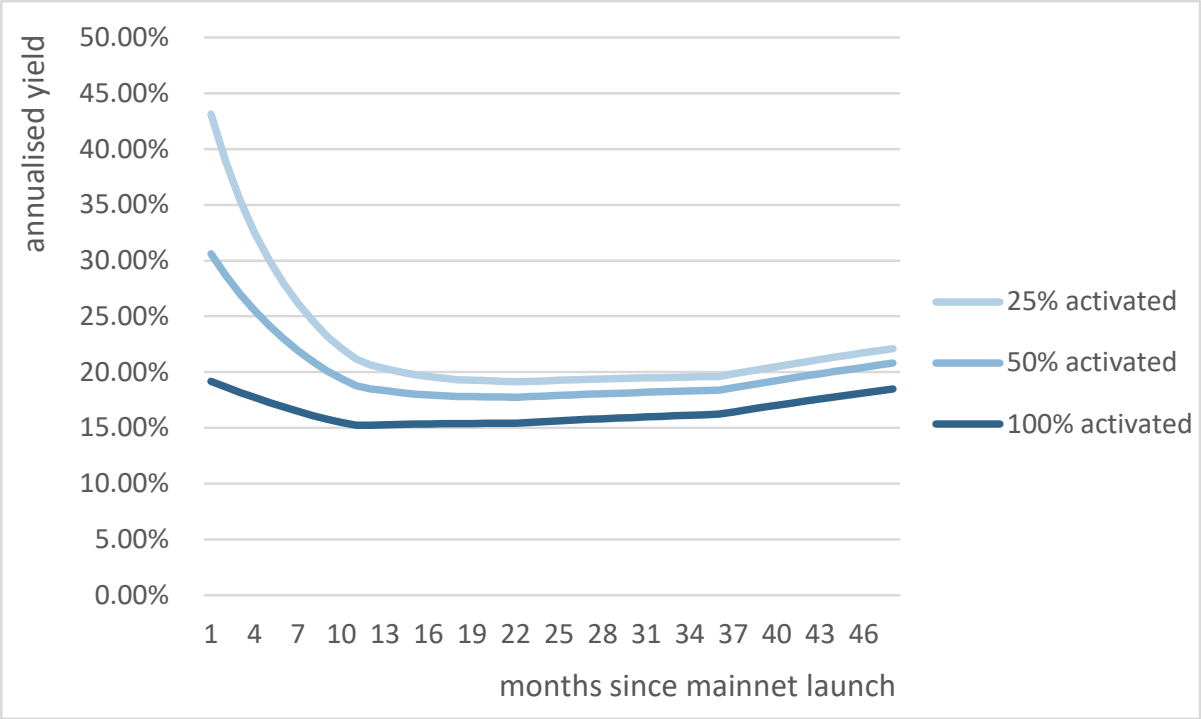
Delegator yield (in tokens) over time, charted for different delegate staking rates (tx = $0)



The final sensitivity we planned to examine how the number of nodes impacts the returns available to delegators.

To do this we now plot delegator yields under three scenarios. One where 100% of tokens sold with operator privileges are activated, one where 50% activate, and another where only 25% activate. We keep all our previous assumptions the same, set the delegate staking rate to 50%, and transaction fees to $0.1/txn.

Delegator yield (in tokens) over time, charted for differing numbers of operators



We observe that differences start off very large, but quickly narrow. The explanation behind this is due to the gradual unlocking of tokens released by vesting contracts. Since this liquid quantity of tokens starts off small, the operating stakes of additional operators have a larger impact on the share of rewards claimed by delegators.

As liquid supply increases however, operators minimum stake forms an ever smaller share of liquid supply, and so the number of node operators has a smaller impact on the share of stake claimable by delegators.

NODE OPERATOR ECONOMICS

Node operator economics share many of the same fundamentals that drive outcomes for delegators. This is simply down to the fact that they both share the same funding sources.

Nevertheless, operator economics do exhibit distinctive features.

The most important difference is the substantial minimum stake required to enact the operator role. This naturally shifts the analysis of economic benefit to be framed in terms of Node value.

Node values were calculated for a 3-year horizon, assuming costs that are set by the token prices provided in the Node Round.

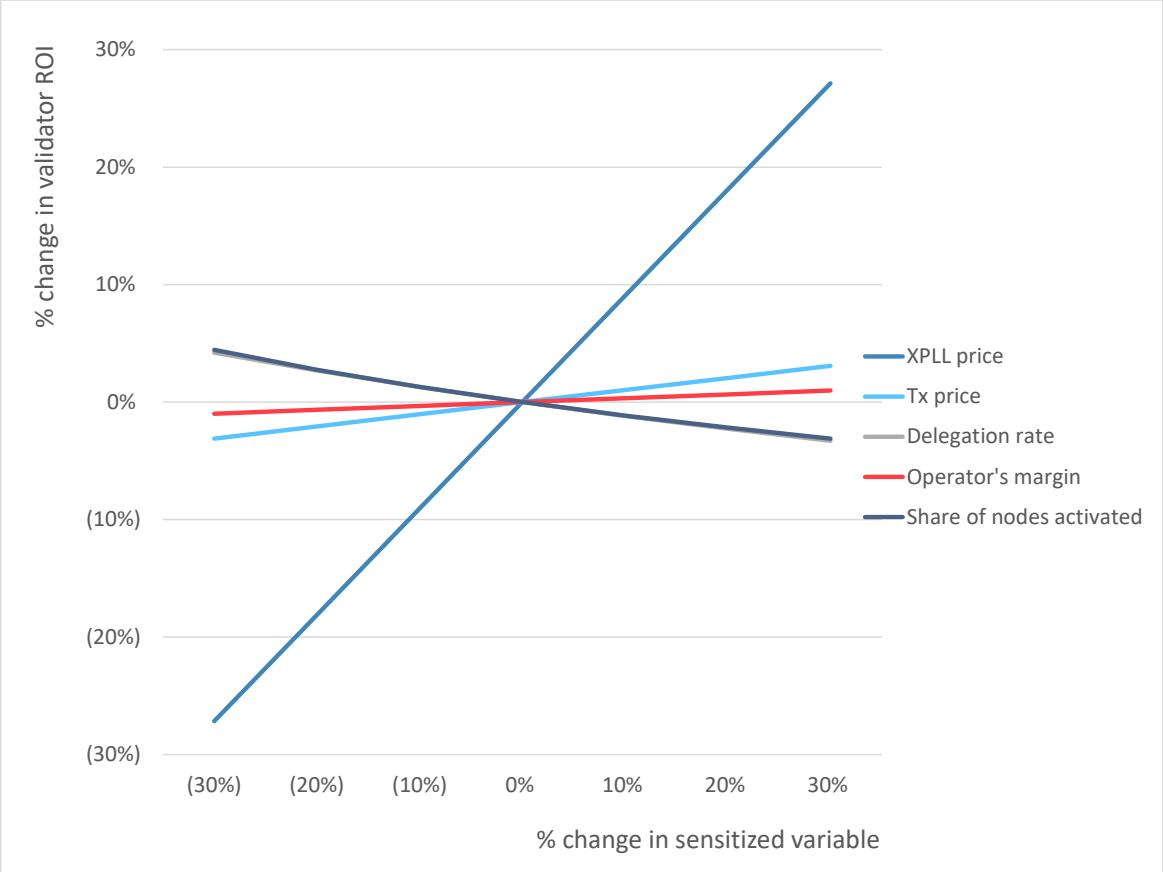
We then examined how this value is affected by changes to other key drivers, made one at a time, independently of each other. Luckily, this limited approach still generates us with a rich account of operator economics.

Some preliminary analysis showed us that a comprehensive economic assessment can be created by examining the impact of variations of just six key factors:

1. Node type
2. Price of XPLL
3. The number of nodes on the network
4. Average transaction price[7]
5. Delegate staking rate
6. Operator margin

The spider diagram below charts the relative impact to Node value made by relative changes to the value of each of these selected factors (around a chosen base case value).

Sensitivity of operator ROI vs. relative changes of key economic factors



[7] The transaction volume also matters. But the proportional impact of changes to transaction volume is identical to what happens when the same proportional change made to the average transaction price. So we can leave out transaction volume for this exercise.

Node type: Governing Node
Base case values sensitized:

- *XPLL price = $1.000/XPLL*
- *6 G-nodes, 15 A-nodes, 33 B-nodes*
- *Average transaction price = $0.1/txn*
- *Average delegate staking rate = 50%*
- *Operator margin = 5%*

The conclusions that we can draw from this analysis are:

- The price of the token is the most important factor. It charts the steepest curve, and increases positively impact the node value (unsurprisingly).

- Transaction collections are the next most important factor, although in this chart it looks far less important. Broader analysis showed that this sensitivity did vary considerably over a range of plausible transaction price values. The line would become much steeper if we had assumed a higher price for our base case.

- Operator margins will not create a large impact. This conclusion is robust over a large range of base case values.

- Higher delegate staking rates reduce node value. There are some auxiliary assumptions behind this result. We are assuming that a higher number of delegated tokens dilutes the share the operator holds within their own pool (all else equal). This means that the operator receives a smaller share of the rewards allocated to their pool. We expect delegation rates on ParallelChain network to start off fairly high, perhaps around 70%, and then fall down to around 30% once additional token utilities are created by DApps developed for the chain.

- The share of operator options activated has a similar relative impact to that of delegate staking rates. This makes some sense, although the closeness of the slope of the two curves may be accidental given all that factors at work.

Note that the chart and its analysis come from the data modelled for a Governing node. The inferences offered however, do not change materially by node-type, since all the analysis is done on a relative basis.

There is something important to add concerning the economic implications of node class, that could otherwise go unnoticed.

Node classes are not just distinguished by the level of return they offer, but by their ability to generate those returns on larger capital bases.

This follows from the reward cap structure that ParallelChain employs to avoid excessive concentrations of stake. Let's illustrate this with the following example.

Consider a situation where a Governing node and a Beta node both happen to both be generating 5% annual token yields on the minimum operating stakes they hold in their pools, and all other tokens in their pools are contributed by delegates. Suppose, additionally, that both pools are at the optimal size targeted by ParallelChain's incentive design.

Even though both pools are yielding 5%, the Beta pool makes less in *absolute terms*, because it has a smaller operating stake. Furthermore, in this case the Beta node operator cannot accumulate more tokens to grow the base upon which they earn that 5% yield. This is because the pool is already at its target size, and the protocol will not allocate any more rewards to it if it grows any larger.

This is an important economic difference between nodes that might otherwise be obscured if you limit your analysis to the operator %node-value metric (or other similar *rate* of value measures).

Further explorations beyond what we have presented here, indicate that attractive operator returns can be sustained for all three node classes, even under some rather modest assumptions on the token and transaction prices, and the delegation rate.