First Economic Model Proposal for The OneFinity Blockchain

OneFinity Team – v0.0.3
Last Updated: Feb. 2024

Disclaimer: *Nothing in this paper or OneFinity website is an offer to sell, or the solicitation of an offer to buy, any tokens. OneFinity is publishing this paper solely to receive feedback and comments from the public. Nothing in this paper should be treated or read as a guarantee or promise of how OneFinity's business, services or the token will develop or of the utility or value of the token. This paper and OneFinity website outlines the current plans, which could change at its discretion, and the success of which will depend on many factors outside OneFinity's control, including market-based factors and factors within the data and cryptocurrency industries, among others. Any statements about future events are based solely on OneFinity's analysis of the issues described in this paper or OneFinity website. That analysis may prove to be incorrect. ONE is not a "stablecoin" and may be volatile and/or may lose value. No recommendation is made herein as to the advisability of purchasing ONE; notwithstanding, do not purchase ONE if you cannot bear the loss of the entire purchase price.*

Preface

Through OneFinity we propose a new vision, providing a new economic model and language specifically designed for the information age. The power of an EVM-compatible blockchain built on MultiversX Sovereign Shard infrastructure.

This paper offers a temporary snapshot of the economic principles governing the OneFinity Network, as they stand at the time of writing.

The OneFinity token is inseparable from the OneFinity Network, and thus intrinsic to it. Some of ONE's intended use cases include staking, delegation, payments, fees for storage rent and for smart contracts deployment, as well as rewarding the validators that contribute to the Network's performance, stability and security.

In this first phase, gaining access to a recurring value stream generated by the network is conditioned by owning the ONE token, as the native asset of the OneFinity Network.

Once OneFinity becomes a thriving global ecosystem and public utility, one might expect the token to become a robust store of value, owing to compounding programmable incentives and strong underlying network effects governing the blockchain architecture. Its quality as a store of value will be a function of the underlying economic incentives amplified via real world adoption, defined conditional transition to a true deflationary economic model, and accrued trust in the OneFinity Network.

The OneFinity Network, on the other hand, is a proof-of-stake based blockchain platform where a set of validators, who have staked ONE, produce blocks by reaching consensus. Validators are rewarded for their work and staked ONE.

Any number of ONE holders can participate in staking indirectly by delegating their ONE to existing validators, usually professional validators (staking-as-a-service providers), that choose to accept delegations. A ONE holder indicates which validator candidates they trust and puts some ONE at stake to support their delegation. If one or more of their candidates are elected as validators in an epoch, they will share with them any economic rewards, proportional to their delegated stake. Delegating ONE is a way of investing one's ONE and contributing to the security of the system. The larger the total amount of ONE staked, the higher the system security, thanks to the increasing amount of stake needed by an adversary to get any nodes elected as validators.

How to contribute and give feedback

This paper is the first public draft of the OneFinity economic model. The individuals and companies contributing to this paper operate in a dynamic environment where new ideas and risk factors emerge continually. Thus, we are constantly looking for feedback, with new assumptions that could challenge and improve parts of our model. We encourage those who want to contribute, to provide their feedback through OneFinity's available forums.

1.1 Cryptoeconomics

Definitions
Cryptoeconomics can be aptly described as the use of incentives and cryptography in designing distributed networks. It is not a subfield of economics, but rather an area of applied cryptography that takes game theory into account.

Game theory is the mathematical modeling of strategic interaction among rational (and irrational) agents. Mechanism design, on the other hand, is a subfield in game theory, often referred to as reverse game theory, because we start with a desired outcome in mind, and work backwards to design a game promoting it. A game where rational self-interested players will produce a desired outcome.

So, if game theory is about choosing the best moves in a given game, mechanism design is about creating a game which accounts for the moves you desire.

To sum up, cryptoeconomics consists of two components: **cryptography** which is the part of the mechanism that ensures the integrity of past moves, and **economics** which is the part of the mechanism that ensures all actors take the proper future moves.

The economic security guarantees of any crypto network depend in part on the strength of its assumptions, about how people react to economic incentives. However, it is worth noting that mechanism design is not a panacea, and **cryptoeconomics** cannot be applied in a vacuum. There is a limit to how much we can rely on incentives to predictably shape future behaviours.

Creating a model
Several aspects are taken into account when creating a crypto economic model:

- desired behavior of all actors
- economic incentives such as rewards and fees for the well behaving actors, but also penalties for any actor that may have misaligned incentives relative to desired behaviors
- economic rules (like rating, penalties or slashing) that discourage specific behaviours: invalid protocol messages, failure to produce, omission of protocol messages, equivocation and others.

The economic aspects of the incentives being implemented must take into consideration that, regardless of the monetary value of a particular token, there are factors that influence the wellbeing of the system, namely:

- the inflation should be small enough to not "tax" the token holders, but large enough to cover their staking costs (running nodes)

- the monetary supply being staked should be large enough so that there are enough distinct entities that collusion is unlikely, but small enough so that money velocity is not affected (MV=PQ)

As can be seen, cryptoeconomics are the rules of the game, but how does one change the rules after they have been put in motion? The answer is governance. Governance is the power to change the rules, and as the game becomes more valuable, governance becomes the metagame that can sustain or destroy that value.

1.2 Governance

Cryptographically secured distributed networks provide a neutral layer of decentralization, immutability, privacy and trust. Smart contracts can thus be used to both run provably fair electronic elections, or to buy them.

Given their significant and far-reaching implications designing an effective governance mechanism for decentralized systems is a strenuous task. Mere extrapolations of real-world governance models are proving naive, and many crypto-networks will likely die due to flawed governance once their network will reach a sufficiently high value for a range of decisive attacks to be warranted. Thus, governance requires separate, in-depth consideration. The OneFinity governance model will be outlined in a future paper, to be released at a later stage, after the official launch of the OneFinity Network. Prior to that point, OneFinity will use a robust off-chain governance approach to ensure maximal speed and efficiency.

1.3 Terms and Organizational Components

A clear definition is necessary, for the terms used to describe various actors, and actions, inside the OneFinity Economic Model.

Users or Network Participants
Any party, individual, entity, enterprise, blockchain or network that uses, develops, creates or interacts with any aspect of the OneFinity Network. Users are identified by a unique account address (derived from their master public-private key pair stored in a wallet).

Token Holders
Users that are holders of native ONE tokens, to be used on the OneFinity Network to submit signed transactions for value transfers, smart contract execution or to provide liquidity.

Application Developers
Users that develop smart contracts and/or applications that rely on smart contracts to provide services. Developers need an account to deploy smart contracts on the Network.

Consensus group
In order for a block to be proposed and committed, a specific number of nodes (*numNodesConsensus*) are randomly selected from all eligible nodes (*eligibleNodesPerShard*) assigned to a shard to form the consensus group during each round (*blockTime*). The consensus group has the responsibility of committing blocks in that shard, during each round. At the beginning of each round, a new consensus group is selected.

Nodes
Devices (computers or servers) running the software (the OneFinity client) and relaying messages received from their peers. They can be either **Validators** (actively participating in securing the network) or **Observers** (passive members of the network that can act as a read & relay interface) and can be either Full (have the entire history of the blockchain) or Light nodes (only keep 2 epochs of blockchain history). A node is on the eligible nodes list if several requirements are met: a rating above a specific threshold, won a node slot in the selection auction (when auction will be enabled).

Validators
Validators are nodes — computers on the OneFinity network that process transactions and secure the network by participating in the consensus mechanism, while earning rewards from the protocol and transaction fees. In order to become part of the OneFinity network, a validator needs to commit a collateral in the form of ONE tokens **(3000 $ONE)**, which are staked to align the incentives of the validators with the correct functioning of the network. Validators stand to lose some, or all their stake if they deviate from the protocol instructions, or otherwise collude to disrupt the network. In order for a node to be able to become a validator it needs to be on the list of eligible nodes.

Block proposer
The block proposer role is designated to the first selected (through an unbiasedly, random process) validator node in the consensus group. The block proposer is the validator who proposes the next block, which the rest of the consensus group must verify and approve.

Block rewards
The blockchain will reward the validator nodes for their staked ONE. The reward consists of two types: a part of the transaction fees, and new emission of ONE (also called minting or inflation). OneFinity holders who do not put their ONE at stake by being a validator or delegating their ONE to a validator will not receive any of the block rewards.

Protocol Sustainability
The protocol sustainability has the purpose of increasing the security and value of the Network on the short, medium and long term. The specifics of governance and management of protocol treasury will be presented in the governance paper. Until then, the protocol treasury will be under the control and supervision of the OneFinity Core Team.

Protocol Governance Body
A self-organized decentralised autonomous organization, overviewed by a non-profit foundation.

2. Validators

In order to secure the network, OneFinity will utilize a Proof of Stake model.

Unlike Proof-of-Work (PoW) systems, OneFinity does not require machines to solve any puzzles. Instead, all the computational power of the network is used for actual transactions, hence, with Proof-of-Stake, the energy saving is substantial. Additionally, OneFinity does not require any GPUs or specialized chips in order to support the network: you can contribute to and support the network using the hardware you already have (if it meets the minimum requirements:

- 4 x dedicated/physical CPUs, either Intel or AMD, with the SSE4.1 and SSE4.2 flags
- 8 GB RAM
- 200 GB SSD
- 100 Mbit/s always-on internet connection, at least 4 TB/month data plan
- Linux OS (Ubuntu 22.04 recommended) / MacOS).

In Proof-of-Work (PoW) systems where a miner takes everything (block reward + transaction fees), there is only one way to improve your chances of being successful: increase your hash power. This leads to three outcomes:

I. It becomes uneconomical for small/low power devices to participate.

II. Mass pooling of resources becomes desirable.

III. Specialisation of hardware becomes necessary.

In contrast, Proof-of-Stake (PoS) does not rely on rewards for securing the network, but rather on penalties. Validators put money (“security deposits”) at stake and are compensated for locking up their capital and incurring costs for maintaining the node. Most of the cost of acting against the rules comes from penalties that are hundreds or thousands of times larger than the rewards an attacker could get in the meantime. So, if in PoW the miners are competing with each other, in PoS validators are collaborating with each other.

In this way, a PoS network allows for a much more resource-efficient, scalable, and inclusive way of maintaining a permissionless blockchain network. Looking at the numbers gathered in early PoS networks and the two largest PoW networks (Bitcoin and Ethereum), we can see that the money spent on infrastructure is an order of magnitude smaller in PoS compared to PoW (around 10% of the rewards instead of 100%).

So, in OneFinity there is no mining. Instead, validators earn tokens for doing useful work. One of the most important aspects we had in mind when designing the OneFinity Network was achieving guarantees of fairness for all the network participants. In the case of validators, we designed OneFinity to be resistant to concentration of resources and to ensure an equal and fair distribution of rewards based on the work done by all validators, whether big or small.

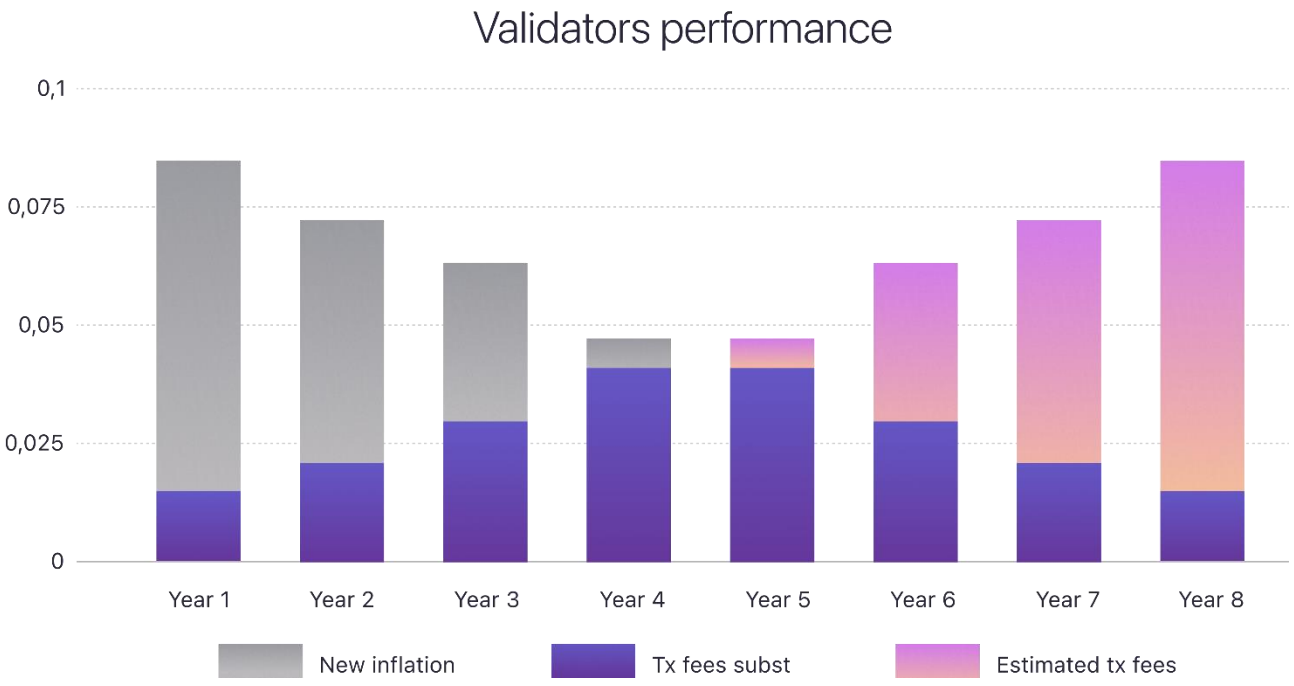
Network participants bring value to the network. The more validators, the more ONE staked, and the greater the security and decentralization of the network.

2.1 Validator ratings

As with any decentralized and permissionless network, we are expecting to see participation from many validators, from different locations, using different hardware specs, infrastructure setups, internet connections, bandwidth, etc. This will lead to different performances in terms of up-time, response time, computation time, etc. While these variations are acceptable and expected, the more decentralized the network is, the clearer it becomes that specific actions are more desirable, while others actions are not. Keep in mind that when discussing rating, we are referring in general to the up-time and hardware/setup performance (manifested as the amount of blocks successfully proposed and signed), and not behaviour and actions against the protocol, which are covered by the slashing section (double-signing, equivocation, etc.).

Through the rating mechanism, we are rewarding desired performance (such as uptime and correct proposal of a block), but we're also penalizing undesirable actions affecting the performance of the network (such as missing block proposals). The higher the rating of a node, the higher the chance to be selected as a consensus validator in a round (which implies having the opportunity to earn rewards). Conversely, the lower the rating (but above a configured value ratingThreshold), the lower the chance to be selected as a validator. The reward or penalty is performed merely through an increase or decrease of the node rating, so no slashing is involved.

Figure 1: Validator performance 8-year plan.



The rating of a node is an integer value between 0 and 100, inclusively. All ratings are stored by the Sovereign Shard, which tracks the activity of the nodes round by round, and at the end of an epoch, the Sovereign Shard adjusts the ratings accordingly. Each node joins the network with the same initial *startRating*, which is carried on and adjusted from epoch to epoch.

Table 1 quantitatively presents how the rating of a node increases or decreases its chance to be selected as a consensus validator (subject to change in time when more data is available).

Table 1: Node rating as a consensus validator.

Rating interval	Chance modifier
0-10	-100%
10-20	-20%
20-30	-15%
30-40	-10%
40-50	-5%
50-60	0%
60-70	+5%
70-80	+10%
80-90	+15%
90-100	+20%

A validator node can increase its rating in two ways:

1) Maintaining a good record of signing proposed blocks. Whenever a node is chosen to be a consensus validator, its rating will be implicitly increased by the value *validatorRatingIncrease*, given a good block signing record.

2) Proposing a valid block when selected to be the block proposer (i.e. consensus leader). A valid block will cause the rating of the block proposer to be increased by the value *proposerRatingIncrease*.

In order for a node to be eligible to receive *validatorRatingIncrease* upon its selection for consensus, it needs to have signed a minimum percentage of blocks out of the last continuous sequence of *numValidatedBlocksRange* it has been a validator for (counting into the past epoch is allowed). The percentage of signed blocks must be equal to or greater than the value of *signedBlocksThreshold*. The reason behind this approach is the fact that for a proposed block to be validated, only ⅔ +1 signatures are needed. We do expect that a node will have its signature present on at least some blocks on a specific, long enough, time frame (or number of blocks), in order to increase its rating for validating. This limit needs to be high enough, so we don't encourage free-riding nodes which don't actually sign blocks, but only propose blocks when they happen to be block proposers. On the other hand, nodes that are consistently slow and which are not able to send their signature for blocks in the required time will, at some point, stop receiving a rating increase for being selected in a consensus group, because their percentage of signed blocks will drop below *signedBlocksThreshold*. Moreover, in this situation, they might start losing rating points (to be implemented at a later stage).

The rating model is thus designed around encouraging productive nodes as much as possible, either as validators or as proposers. A primary design concern is how long does an ideal node need to reach the maximum rating possible, after joining the Network. This duration is named *HoursToMaxRatingFromStartRating*, and it is the expected number of seconds needed by a node

to gradually reach the maximum possible rating (*maxRating*) in ideal conditions, starting from the initial rating value, *startRating*. The value of *HoursToMaxRatingFromStartRating* will likely be configured to equal a few days.

Starting from *HoursToMaxRatingFromStartRating*, the model defines the functions *avgValidatorRatingPerRound*(·) and *avgProposerRatingPerRound*(·), which express the average number of rating points gained by an ideal node, per round, when selected as a validator and as a proposer, respectively. These functions depend on the aforementioned *HoursToMaxRatingFromStartRating*, consensus group configuration and on the *importanceRatingRatio*, a fixed proportion defined as:

$$importanceRatingRato = avgProposerRatingPerRound(\cdot) \,/\, \text{avg}\,ValidatorRatingPerRound(\cdot)$$

This proportion balances the amount of rating points gained by validators versus proposers, a necessity given the fact that it is far more likely to be selected as validator in a round, instead of as proposer. For block proposers, it is desired that the overall contribution to rating of the total *proposerRatingIncrease* awarded in one epoch should ideally be the same with the total of awarded *validatorRatingIncrease*. The exact definitions of *avgValidatorRatingPerRound*(·) and *avgProposerRatingPerRound*(·) are currently in development, as they depend on the model chosen for the consensus selection algorithm.

The values for *validatorRatingIncrease* and *proposerRatingIncrease* can be expressed as follows:

$$validatorRatingIncrease = (maxRating - startRating) \,/\, avgValidatorRatingPerRound(\cdot)$$

The current intention is to keep both *validatorRatingIncrease* and *proposerRatingIncrease* to constant values. To achieve this, the definitions of *avgValidatorRatingPerRound*(·) and *avgProposerRatingPerRound*(·) must be adjusted accordingly, because they alter the rating of a node, which in turn alters its probability of being selected for consensus. As explained earlier, this then affects the rating, forming a controlled feedback loop. Non-constant alternatives for *validatorRatingIncrease* and *proposerRatingIncrease* are being considered by the team as well.

Apart from having its rating increased, a validator will have its rating decreased if it fails to propose a valid block when selected as a block proposer, regardless of the reason of the failure. Every time a block proposer fails to fulfil its role, its rating will be adjusted with the following penalty:

$$proposerRatingDecrease = -4 \cdot proposerRatingIncrease$$

A validator that is offline or not able or willing to produce new blocks or sign blocks, will have its rating decrease a lot faster than the rate is increasing. This can be further accelerated if more nodes have their rating below *ratingThreshold*.

It is up to the implementation phase to avoid penalizing honest block proposers after a malicious round took place (1 block before) by delaying the block.

2.2 Transactions with MultiversX mainnet

Validators from the sovereign shard run a notifier service which sends information from the mainchain when there is a bridge transaction from the mainchain to the Sovereign Shard. The validators can connect to their own nodes for the notification events, or they can connect to the public notifier services.

Every time the leader sees a new header from the mainchain, they will try to include it in the block which they are currently building: specifically adds the MultiversX *ShardHeaderHash* to the *SovereignShardHeader*, if there are bridge transactions from the mainchain, the protocol obliges the leader and validators to process all the transactions if they want to add the *ShardHeaderHash*. Here's how destination shard processes in case of cross shard transactions. One *SovereignShardHeader* can contain a list of *ShardHeaderHashes*, and the leader will add only those *ShardHeaders,* which were final on the mainchain.

If the MultiversX *ShardChain* Block contains incoming bridge transactions, those must be added and executed in the sovereign chain. The same as destination miniblocks. The leader cannot add a new MultiversX *HeaderHash* into the list if we did not execute all the transactions from the last one. The leader in the Sovereign Shard has the job of collecting all the incoming bridge transactions and creating a *miniblock* out of it. This is the first thing in the Sovereign Shard we should process - exactly like *DestMe* transactions. Technically speaking: the validator receives block data through go-notifier - which selects and prepares the relevant data for the Sovereign Chain. The prepared information is pushed to the Sovereign Chain client. The prepared data contains a proof of execution, data availability, correctness and finality for those transfers. The name of this *miniblock* is "*INCOMMING TXS"*

The validators on the Sovereign Chain will do the same process and verify if the thing the leader created is correct or not. The validators are connected to the mainchain, will run the header verification, incoming transaction creation, proof verification and finality checks. If any of those fails, the validators do not sign the block, thus the block is not created.

The incoming bridge TX can be executed without GAS on the Sovereign Chain and there is no need for a Smart Contract on the Sovereign Chain - treat these transactions as *DestME* ESDT transfers. Thus, speeding up execution as well. This is a complete integration, complete coupling of the OneFinity Sovereign Chain to the MultiversX chain.

2.3 Staking rewards

Staking rewards, possibility of slashing, or increasing/decreasing a node rating, are a set of incentives that encourage token holders and validators to secure the OneFinity Network. In return for security, the validators can increase their relative share of token holdings in the network.

We believe that staking rewards do not exist to provide an income stream per se to the token holders. In fact, the economic rationale for staking is not to receive a reward ("yield"), but instead to clearly assert to the validators that staking increases their relative interest (through the amount of ONE owned) in the network, and also contributes to significant token appreciation.

With this in mind, it is better to look at the inflation rate as a token holder dilution rate instead. As such, staking is the best way to grow your token holdings and interest in the OneFinity network.

Here are how rewards will be paid in **OneFinity**:

> There will be a minimum guaranteed reward amount per year. The minimum guaranteed reward amount will come from fees and token inflation. The maximum inflation rate per year is based on if the fees are 0.

The guaranteed rewards per year are calculated based on the following formulas:

Table 2: Max Issuance 8-year plan.

Year	Max Total Supply	Max Issuance rate %	Max Yearly supply to be added
	25,500,000		
1	26,775,000	5.0%	1,275,000
2	27,979,875	4.5%	1,204,875
3	29,099,070	4.0%	1,119,195
4	30,117,537	3.5%	1,018,467
5	31,021,064	3.0%	903,526
6	31,796,590	2.5%	775,527
7	32,432,522	2.0%	635,932
8	**32,919,010**	1.5%	486,488

*These rewards are used as an example. As the token is deflationary the actual rewards will adjust based on the current total supply.

The guaranteed rewards will be distributed between all the active validators. Therefore, less validators means more rewards per validator.

There will be a maximum total of **7,419,010** ONE minted to provide the rewards for staking based on the total supply being **25,500,000** ONE**.**

2.4 Fee rewards calculation distribution

The fee rewards are distributed at the end-of-epoch by the following rules:

- 70% to validators
- 20% to be burned
- 10% for the OneFinity Foundation

Please refer to the fees section **3** for more information about fees.

2.5 Unstaking and unbonding

Unstaking
If a validator wishes to unstake, they will initiate a transaction that indicates that they want to unstake a number of nodes, including the BLS public key of each node. The transaction is generated by the validator and sent to the Sovereign Chain.

At the end of the epoch, when nodes are re-shuffled, those who unstaked during the just-completed epoch will be shuffled out first.

- If the node cannot be shuffled out, then the node must “stay and work.” If the node decides to go offline, then their rating decreases and at some point they will be under *ratingThreshold* making it ineligible to participate in the next selection or auction process. A node below *ratingThreshold* cannot be un-staked until the rating is above *ratingThreshold* (see *resetRating* transaction).
- If there are more unstaking nodes on a shard than the number of nodes in the waiting list the Sovereign Chain computes an order for shuffling out and just the first waiting nodes from the list are removed.

If a validator initiates unstaking, and then in the same epoch, decides not to proceed, he can send a re-stake transaction and his unstaking will be canceled.

The unstaking information is saved in the validator staking smart contract. The re-stake transaction is the same as submitting an initial stake, the only difference is that the validator does not need to send the value again.

Unbonding
The unbond period is set at **10 days**, after which the node will be able to retrieve its previously staked funds.

During the unbonding period:

- It is theoretically possible for a node’s unbonding period to never end if all the nodes of the system have left, and there are not enough nodes to run the shard. However, this cannot practically happen, as OneFinity will provide nodes for at least the shard, at the minimum reserve node price. In this way, we ensure a fail-safe mechanism where OneFinity is the node operator of last resort.

At the end of the unbond period, the validator sends a transaction requesting the unstaked money for each of the nodes that he is choosing to unstake.

The unbond request is processed by the Sovereign Chain nodes only if the unbond period has concluded for each respective node. If the unbond period has not concluded, then all gas is consumed.

2.6 Delegation

Since not everyone will be able to be a validator and run a node, those who still want to stake, can delegate their stake to other validators or Staking as Service providers, and split the rewards between them.

OneFinity as a company will run a number of nodes. Community members will be able to delegate their stake to OneFinity as a staking as a service provider during the first few months, along with community-based validators too.

The general requirements for one such contract is to distribute the rewards generated by the validators, towards the community members who staked their token through the contract. The distribution has to take care of when to give the rewards to the registered members, and how much of a service fee is taken out.

More information about delegation in general and the smart contract template for delegation, provided by OneFinity as a guidance, will be announced later in a different paper or medium post, which will detail more about the 1000 nodes distribution through the NFT sale process.

3. Fees

A sustainable value stream for the network can come from transaction fees and asset inflation. Since the success of the network is reflected by the adoption and usage which will generate transaction fees, the economic model will be able to finance the growth and maintenance of the network without the need of inflation.

The calculation and distribution of rewards and fees is done at the end-of-epoch and added to the start-of-the-epoch block by the block proposers, verified by all the validators.

30% of the fees generated by smart contract computation will be taken by the owner of the smart contract.

The remainder of the fees will be distributed as follows:
For all blocks produced in each round, 70% of all transaction fees of a block are added in a pool and are distributed to all validators at the end of the epoch, 20% of the fees will be burnt and 10% will go to the OneFinity foundation.

After reviewing initial simulations, we have decided that transaction fees will start with a minimal amount of ONE per transaction, with specific costings provided prior to Mainnet deployment.

3.1 Transaction and smart contract fees

Transaction fees are calculated as follows:

1. Value transfer transactions:

*(moveBalanceGas + storePerByteGas * len(txData field)) * GasPrice GasPrice >= minGasPrice*

2. Smart contract transactions:

*(moveBalanceGas + storePerByteGas * len(txData field)) * GasPrice + (actual smart contract processing gas) * GasPrice*

The appropriate block proposer will calculate them directly during the consensus process.

Transaction fees are calculated using a gas model. This takes into consideration: the quantity of resources used per transaction, including:

I. CPU
II. Bandwidth
III. Storage

For any given block in the OneFinity Sovereign Shard, the transaction fees included in the block are aggregated (see Staking Rewards section 2.3). Until the end of the epoch (at which point the pooled transaction fees are distributed to the appropriate agents), the transaction fees are controlled by no agent, and are stored as information in the Shard Chain block header.

Each transaction must specify the amount of gas it needs as part of the transaction data. While a block producer creates a block, it will execute each transaction with the consumed gas being deducted, and remaining gas being refunded. If a transaction specifies enough gas for the execution, but insufficient funds for the actual transfer, then the execution will consume the given

gas, but the move balance function (or smart contract call) will not cause any balance change due to insufficient balance (the account nonce will be increased and the transaction will be added to the blockchain as an invalid transaction).

For any transaction, this amount can be calculated by an overestimation (but no more than 10x) of expected gas usage, thanks to the unused amount being returned to the payer, after all transactions are completed. If a transaction doesn't attach enough gas to execute a required function, the transaction will terminate early and fail, but still charge for spent gas.

For any transaction that specifies less gas limit as stated in section 3.1, formula 1, the system will reject that transaction, thus not notarizing the transaction (not even as a failed transaction).

Future work will explore the possibility to adjust fees based on the load on the entire network, so that, for example, as long as the load is under 50% we have a min. price per gas, but when the load goes above 50%, the gas price increases. In order to avoid manipulation of the gas price by holding transactions, an expiration time will be set to each transaction.

3.1 Storage fees

Storage should be considered separately from computation or bandwidth, because each smart contract transaction that will require storage across all validators going forward, is not just subject to a one-time fee at the execution of the transaction, but also a storage cost.

OneFinity will introduce a state rent for smart contract transactions, where there will be a fixed price per byte that needs to be stored (in the future this fixed price can be adjusted via governance), a price that will be paid periodically. The state rent price is applied only to smart contracts and not to normal balance accounts. We will also introduce a mechanism to temporarily clear the state of an account (unable to pay the rent), to hibernate the account, and restore it once needed.

3.2 Developer's fees and monetization

To significantly accelerate developer adoption, we will provide developers with a built-in protocol monetization solution. Thus 30% of the fees directly associated with a dApp, will go to the developer. So, when processing a smart contract transaction, 30% of the fees from that transaction will be added to the smart contract balance.

4. ONE

The native OneFinity ONE token is opening a new growth phase for the OneFinity economy. It is a natural step toward enabling native OneFinity services such as staking and delegation, and native DeFi options.

4.1 Overview

An overview of the most important ONE premises:

a) **The ONE currency is designed for simplicity and global adoption**
Complexity is the most important obstacle for real world adoption. To reach the greatest audience and user base, we've completely rethought the OneFinity currency, capturing its essence into a universally appealing and powerful actor.

b) **The ONE currency is designed as a digital reserve standard and robust store of value**
A new economics model has been defined to position ONE as the core network token, fundamental to all OneFinity's internal usage. This token is designed to optimize parameters that lend themselves to creating a robust store of value. OneFinity, will in fact, intend to onboard many new tokens, such as stable coins and EVM-compatible project tokens, new and existing.

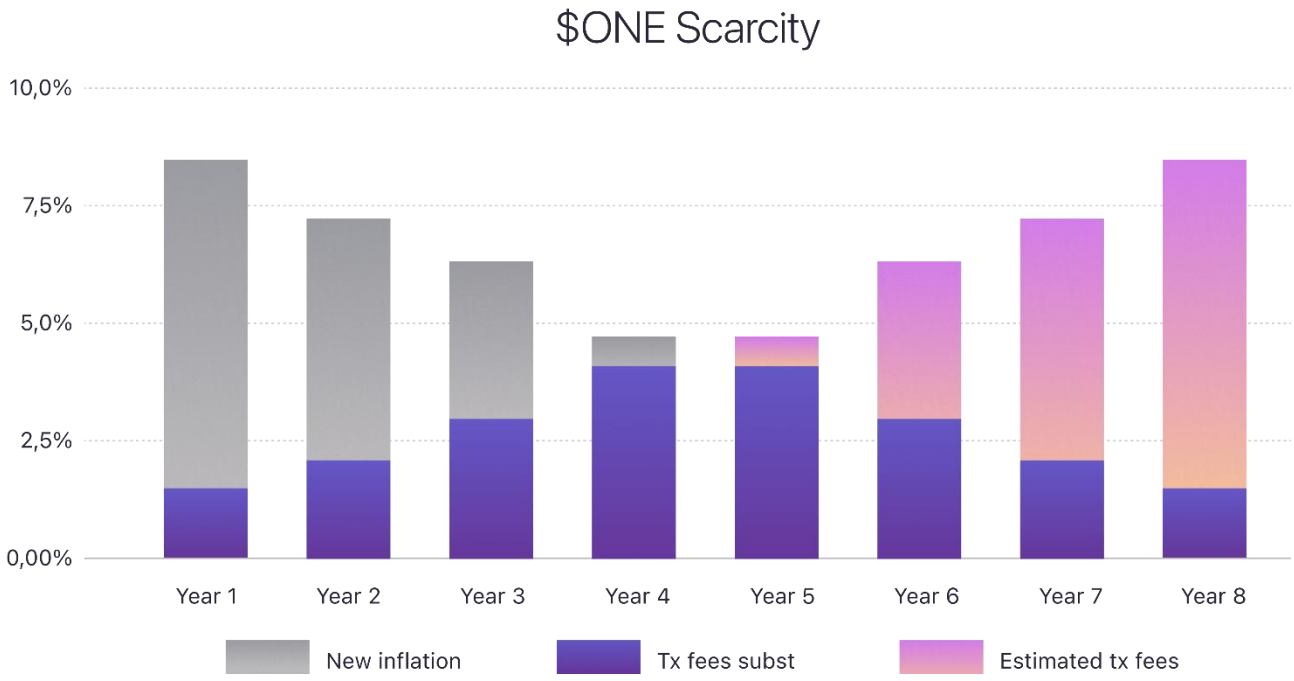
c) **Strong staking incentive for validator adoption paired with a max supply limit** There are strong staking incentives for validators to secure the OneFinity network.

d) **Adoption reduces this theoretical inflation and increases scarcity** One of the most powerful features of the OneFinity economic model is that each transaction fee paid reduces the theoretical limit by substituting inflation with fees, thus making ONE scarcer, ensuring that the max supply limit will never be reached.

e) **A sustainable adoption model growing the entire ONE economy and reinforcing deflation** OneFinity offers arguably one of the strongest adoption models in the blockchain space, thanks to the network being able to immediately transition to a fully deflationary model via any adoption scenario. Indeed, the zero-inflation threshold shows that since 10% of the network capability is needed to cross the threshold, with enough adoption OneFinity can exceed this threshold and create a significant amount of value for all the network participants.

Scarcity In OneFinity the supply starts at ~25,000,000 and exhibits a predictable temporary increase in supply to incentivize network security via staking rewards. The defined maximum supply cannot exceed 32,500,000 over a span of 8 years, with a maximum ~7,300,000 ONE being minted over a 8-year timeline. However, this theoretical cap will actually decrease with each transaction processed and fees generated. Thus, the stronger the adoption, the smaller ONE's supply will become.

Figure 2: ONE scarcity 8-year model.



Divisibility Successful currencies are divisible into smaller incremental units. In order for a single currency system to function as a medium of exchange across all types of goods and values within an economy, it must have the flexibility associated with this divisibility. The currency must be sufficiently divisible so as to accurately reflect the value of every good or service available throughout the economy. OneFinity has a much larger degree of divisibility than most fiat currencies around the world. One ONE is divisible to 18 decimal points. If OneFinity continues to increase in price over time, the large divisibility of OneFinity ensures that with tiny fractions of a single OneFinity, people can still take part in everyday transactions.

Transportability Currencies must be easily transferred between participants in an economy in order to be useful. In fiat currency terms, this means that units of currency must be transferable within a particular country's economy as well as between nations via exchange. In contrast to fiat currencies, where the process of transferring money can take days and have significant fees, as long as there is internet, ONE can be transferred anywhere in the world, in an instant, and at a 100x less cost than current available options. This will be the vision when centralized exchange listings unroll in the coming months.

Durability Durability is a major issue for fiat currencies in their physical form. Paper currency, while sturdy, can still be torn, burned, or otherwise rendered unusable.

It cannot be destroyed in the same way that paper currency can be, although it can be lost. If a user loses his or her cryptographic key, the ONE in the corresponding wallet may be effectively unusable on a permanent basis. However, the ONE itself will not be destroyed and will continue to exist in records on the blockchain.

Counterfeitability Thanks to the robust built-in security of its decentralized blockchain system, ONE is incredibly difficult to counterfeit. Doing so would essentially require confusing a non-trivial part of the network participants and would require an increasingly large and prohibitive cost. The

single way one would be able to create a counterfeit ONE, would be by executing what is known as a double spend attack.

This refers to a situation in which a user "spends" or transfers the same ONE in two or more separate settings, effectively creating a duplicate record. While this is not a problem with a fiat currency note—it is impossible to spend the same dollar bill in two or more separate transactions—it is theoretically possible with digital currencies. What makes a double spend unlikely in OneFinity, is the increasing and prohibitive cost of resources needed to perform it.

5. Protocol sustainability

The OneFinity foundation will receive 10% from the total generated rewards, in order to provide the necessary resources and funds to further develop, maintain and promote the OneFinity protocol.

Please note:

This paper is the first public draft of the OneFinity economic model. The individuals and companies contributing to this paper operate in a dynamic environment where new ideas and risk factors emerge continually. Thus, we are constantly looking for feedback, with new assumptions that could challenge and improve parts of our model. We encourage those who want to contribute, to provide their feedback on the OneFinity forum.