Proof of Stake and Activity: Rewarding On-Chain Activity through Consensus

Karen Terjanian, Aram Jivanyan, Dmitry Sidorov

June 7, 2023

Abstract

We propose a new consensus protocol for blockchain, that builds upon the Ethereum consensus protocol by combining its Proof of Stake component with a special Proof of Activity type of system. Our Proof of Stake and Activity (PoSA) protocol offers an interesting economic model for both simulating decentralization and also rewarding validators based on both their initial stake (wealth) and also on the business value they generate on the chain (creativity). The protocol will be deployed on a full-fledged blockchain platform designed for iGaming and other markets.

1 Introduction

In this note, we discuss a novel consensus method that motivates active network participants and builders to become chain validators and directly rewards their activity. This method will allow the creative dApp creators to get rewarded for the business value they bring to the chain. The presented consensus method referred to as *Proof of Consensus and Activity* still requires the validators to stake a fixed amount of assets to enable slashing mechanisms in the consensus processes. This means every staker can become a validator, but the validator rewards can be increased based on his on-chain activity.

2 Design Rational

In this section we briefly describe the main design points behind the PoSA consensus mechanism.

2.1 Consensus Mechanism

The presented PoSA consensus algorithm replicates Ethereum's Proof of Stake concepts with an interesting modification. The validators are selected not only based on their effective stakes but based on their overall **Validator Power**, which is calculated based on the Validator's staked amount and its generated activity as will be discussed in the following sections. Here are a few important points highlighting the overall design and similarities with Ethereum.

- Like Ethereum, we will maintain both the blockchain execution layer and the consensus layer.

- Certain amounts of tokens will be burned in the execution layer per block and new tokens will be minted on the consensus layer. The details of burn and issuance mechanisms will be detailed in the sections below.

- A single proposer is chosen to propose the next block and a committee is selected to validate the proposer's proposal. Both the proposer and the committee members will be rewarded for their honest and on-time activity The rewarding mechanism will be detailed in the section below.

- The Proposers and sync committee members will get penalties like in Ethereum for misconducting their actions. The penalty mechanisms are detailed in Section 2.8.5 in [1]

- The maximum number of validators in our economy is now limited by 20480. This limitation can be cancelled in the future. The minimum target is starting from 4096 validators.

- The stake amount per validator is fixed to be 8192 FTN. Assuming all 20480 validators are joined to the network, this will result in $8192 \cdot 20480 = 167772160$ staked FTN tokens, which is the 16.77 % of maximum supply.

- The block time is 12 seconds like in Ethereum. The number of blocks per year will be $2628000 = 5 \cdot 60 \cdot 24 \cdot 365$.

- I - Is the default yearly minted amount according to issuance formula $I = F \cdot \frac{S}{\sqrt{S}}$ where S is the sum of the effective balances of validators - $S = \sum_i EB_i$. F is a constant chosen by the development team to hit certain issuance rates under concrete conditions. At this moment $F = 156$ in order to ensure at least 7 % return in the setup with 4096 validators. This constant can be adjusted to control the minimum return of interest for each validator.

2.2 Staking

In order to become a validator, the network participant must stake a certain amount s_{max} of native tokens. This staked amount is locked up as a security deposit on the network and is used as collateral which compels the validator nodes to behave properly and keep the network secure. The stake size is fixed to be 8192 which is referred to as the validator's actual balance. The validator's effective balance EB_i is the actual balance deducted by the penalties happened due to slashing.

In our consensus algorithm, we replicate the slashing and penalty mechanisms designed and used by the Ethereum blockchain with minor modifications [2]. Note that slashing occurs when validators break very specific protocol rules which could be part of an attack on the chain. Validators will receive penalties which can be of three different types depending on the validator's violation. We preserve the logic of correlated penalties where a light punishment is happening for isolated incidents, but a severe punishment occurs when many validators are slashed in a short time period. Like in Ethereum, the block proposers will receive rewards for reporting evidence of slashable offenses. The details of slashing mechanism in Ethereum are detailed in [2] but for the sake of paper integrity

we will discuss the most relevant aspects here.

The Initial Penalty: Slashing is triggered by the evidence of the offense being included in a beacon chain block. Once the evidence is confirmed by the network, the offending validator (or validators) is slashed. The offender immediately has $\frac{1}{32}$ of its actual balance deducted from its effective balance. This is a maximum of 256 FTN due to the cap on effective balance. The effective balance defines the probability of a certain validator being selected as a block proposer or as a committee member. Along with the initial penalty, the validator is queued for exit, and has its withdrawability epoch set to around 36 days in the future.

The Correlated Penalty: 18 days after being slashed which is the halfway point of its withdrawability period, the slashed validator is due to receive a second penalty which is more a correlated penalty and can be zero when there will not be many other slashings during that 18 days of period. This second penalty is based on the total amount of stake slashed during the 18 days before and after our validator was slashed. The idea is to scale the punishment so that a one-off event posing little threat to the chain is only lightly punished, while a mass slashing event that might be the result of an attempt to finalize conflicting blocks is punished to the maximum extent possible. To be able to calculate this, the beacon chain maintains a record of the effective balances of all validators that were slashed during the most recent 8192 epochs (about 36 days). The correlated penalty calculation is detailed in [2]

The effective balance which is the validator staked balance minus all penalties up to that certain period will play a vital role in calculating the validator's power which in turn will define the success probability of the certain validator being selected in the consensus protocol as a block proposer or a sync committee member.

2.3 Validator Activity Score Calculation

The purpose of PoSa consensus protocol is to have a decentralized network where the decision-making power is given to entities not only affording to hold a stake on chain, but more importantly to ones generating certain on-chain activity and thus maximizing the network value in general. In order to do that, we allow each validator to link an active smart contract for his activity consideration and next provide an on-chain activity evaluation method which can be uniformly applied to any smart contract. The quantification approach we will discuss below will eventually allow assigning a deterministic and publicly verifiable activity score to each validator for the given epoch. Note that we still require each validator to lock-up the collateral (stake) and hence this activity score parameter should share the same unit of measurement as the validator's stake, which is the ledger's native currency. In our consensus model, a validator may participate in the consensus with only a staked amount, in which case his activity score will be simply zero. For all other validators associated with active smart contracts and generating on-chain activity, their on-chain activity is properly calculated and added to the staked amount defining the validator's final power in the decision-making processes.

We chose a window size to be equal to 1575 epochs and each epoch to be comprised of 32 blocks. Each validator activity score and hence validator power is calculated and updated over epochs. The validator activity score is calculated based on the gas amount of all transactions associated with the validator's contract. Following the Ethereum's transaction fee calculation formula, each transaction fee on our chain will be computed as

$$\mathrm{tx_{fee}} = \mathrm{gas\_price} \cdot (21000 + \mathrm{tx_{gas}})$$

where gas_price is the current gas price defined by the market dynamics, the 21000 is a fixed constant gas amount, and $\mathrm{tx_{gas}}$ is assigned in a deterministic way based on the smart contract code instructions. For asset transfer transactions not requiring any smart contract code execution, the transaction fee will be simply $\mathrm{tx_{fee}} = \mathrm{gas\_price} \cdot 21000$

Let's define the activity score assigned to a certain transaction as $\mathrm{a_{tx}}$ and the activity score assigned to the smart contract for the given epoch e as $\mathrm{A^e_{contract}}$. For computing the smart contract transaction activity score, we omit the constant 21000 gas and assign

$$\mathrm{a_{tx}} = \mathrm{tx_{gas}}$$

. The smart contract activity score for the given epoch is the aggregation of all its transaction activities which has happened during that period of time.

$$\mathrm{A^e_{contract}} = \sum_{\mathrm{tx} \in \mathrm{e}} \mathrm{a}_{tx} = \sum_{\mathrm{tx} \in \mathrm{e}} \mathrm{tx_{gas}}$$

. The smart contract activity score is updated at each epoch through Algorithm 1. According to this logic, the activity score is updated at each epoch $\mathrm{A^{score_i}_{contract}}$ gradually by adjusting to the score change dynamics spanned over the window period.

2.4 Validator Power

The validator V's power is the final property that defines V's participation in the block validation lotteries. We assume there are n validators $V_1, V_2, \ldots, V_n$ for the given epoch and denote the effective balance of the i-th validator for the given epoch by EB_i. Next, the overall activity happening on-chain over the given epoch can be summarized as the sum of three independent components defined as follows:

- **Validators Activity A**: A is the aggregation of all smart contracts activity scores which are associated with the validators. By defining the i-th validator's activity score as A_i, we will have $A = \sum_{i \in 1,\ldots,n} \mathrm{A}_i$. Hereafter we assume each validator can have only a single smart contract linked to his stake for tracing his on-chain activity. A single entity controlling multiple smart contracts can stake multiple instances each linked with a separate smart contract. The value A can be zero if no smart contract owner becomes a validator.

- **Other Smart Contract Activity B**: B is the aggregation of all smart contract activity scores which are deployed on the chain but are not linked

Algorithm 1 Smart Contract Activity Score Calculation

window_size = 1575(**window** = window_size × epoch)
epoch = 32 × block
for each epoch **do**

$\mathrm{A}_{\mathrm{epoch}} = 0$

Let $\mathrm{TX}_{\mathrm{epoch}}^{\mathrm{contract}} = \{\mathrm{tx}_1, \mathrm{tx}_2, \ldots, \mathrm{tx}_{k_i}\}$ be the given contract's transactions happened during that epoch

for $\mathrm{tx}_j \in \mathrm{TX}_{\mathrm{epoch}}^{\mathrm{contract}}$ **do**
$A_{\mathrm{epoch}} = A_{\mathrm{epoch}} + a_{\mathrm{tx}_j}$
end for

if $i >$ window_size **then**
$\mathrm{A}_{\mathrm{contract}}^{\mathrm{score}_i} = \mathrm{A}_{\mathrm{contract}}^{\mathrm{score}_{i-1}} - \frac{\mathrm{A}_{\mathrm{contract}}^{\mathrm{score}_{i-1}}}{\mathrm{window\_size}} + A_{\mathrm{epoch}}$
else
$\mathrm{A}_{\mathrm{contract}}^{\mathrm{score}_i} = \sum_{j=1}^{i} A_{\mathrm{epoch}_i}$
end if

Output $\mathrm{A}_{\mathrm{contract}}^{\mathrm{score}_i}$
end for

with any validator. We assume most active smart contract owners will be motivated to become a validator although there definitely will be some on-chain activity not linked with any validator. B can be zero only when all smart contracts deployed on-chain are also associated with separate validators or there is no on-chain activity at all.

- **Asset Transfer Transactions Scores T**: T is the aggregation of all smart contract and asset transfer transaction activity scores calculated based on the constant 21000 gas usage $\mathrm{T} = \sum_{\mathrm{tx} \in \mathrm{epoch}} 21000$. T will be zero if and only if there will zero transactions on-chain during that given epoch.

For the given epoch e we define the V_i's power as

$$\mathrm{P}_i^{\max} = \mathrm{T_e} \cdot \frac{\mathrm{EB}_i}{S} + A_i$$

and we define the effective power as

$$\mathrm{P}_i = \mathrm{P}_i^{\max} \cdot \frac{\mathrm{EB}_i}{s_{\max}}$$

where

- $A_{\mathrm{V}_i}^{\mathrm{e}}$ is the activity score assigned to the validator V_i for the epoch e. We omit the index parameters e and V for brevity and simply use the notation A_i.

- T is the sum of all transaction constant gas usage components: $T_{\mathrm{e}} = 21000 * N$ where N is the number of all transactions which have been executed during the epoch e. Again we have removed the index component e for bravity.
- $s_{\max}$ is the staked amount of the V_i-th validator. The initial stake amounts of all validators are equal.
- EB_i is the effective balance of the V_i-th validator. The effective balance is the validator's initial staked amount $s_{\max}$ minus all penalties which it has got before the subject epoch.
- S is the sum of all effective balances: $S = \sum_{i \in 1,\ldots,n} \mathrm{EB}_i$.

In Ethereum PoS consensus algorithm, the validator's power is basically defined by his effective balance EB_i.

2.5 Validator Selection Process

The selection process in Ethereum PoS is executed through a special *shuffle and select* algorithm where the list of validators is first shuffled randomly, and then a fresh randomness *rand* is chosen between the range 0 and MaxRand to check if the equation

$$\frac{\mathrm{EB}_i}{S} \geq \frac{rand}{\mathrm{MaxRand}}$$

holds for the next validator in the shuffled list. If the condition holds, the next validator is selected otherwise the selection algorithm moves on to check the next validator in the shuffled list.

In our protocol, we take the following selection process. For the concrete block

- list all N validators with $P_1, P_2, \ldots, P_N$ powers respectively and compute the overall effective power as $P' = \sum_{i=1}^{N} P'_i$.
- Next randomly shuffle the list of validators and consider the list

  $$\frac{P_0}{P}, \frac{P_1}{P}, \frac{P_2}{P}, \cdots, \frac{P_N}{P}$$

  where $P_0 = 0$ is a void validator added to the list for the sake of integrity
- Chose a random value $x \leftarrow_R [0, 1]$ from the range
- If $x = 0$ then select the first validator with the power P_1, otherwise select the k-th validator from the shuffled sorted list such that

  $$\sum_{i=0}^{k-1} \frac{P_{k-1}}{P} < x \leq \sum_{i=0}^{k} \frac{P_k}{P}$$

It is important to note that in Ethereum, the top 100 smart contracts among the millions deployed on-chain generate almost 25 percent of the overall chain activity. Enabling these few sharks to be constantly selected as proposers would put the decentralization of decision-making processes at a huge risk. So in future versions of the protocol, we may also consider the validator selection process to be executed through an alternative method using the mathematical Sigmoid function with tuned parameters as follows:

$$\{2 \cdot \frac{1}{1 + e^{-1.5 \cdot \frac{P_i}{MP}}} - 1\} \cdot \frac{s_i}{s} \geq \frac{rand}{MaxRand} \cdot 0.62$$

where

1. MP is the maximum power of all validators: MP $= \max\{P_1, P_2, \ldots, P_n\}$
2. s is the actual staking amount to be equal 8192
3. P_i is the power of the i-th validator computed as above.
4. s_i is the effective balance of the i-th validator.
5. The sigmoid function is tuned through certain parameters to enable a smooth empowerment of small validators over the super powerful validators.

2.6 Reward Distribution

In our blockchain certain amounts of tokens are constantly burned and minted in parallel to ensure concrete economic dynamics. New tokens are minted through two separate and logically independent processes. The first process creates new tokens out of fresh air no matter of the existing on-chain activity volume. The second process is linked to the validators activity and will mint new tokens depending on the gas fees spent by the active validators on their activity.

2.6.1 Minting New Tokens Out of Fresh Air

This process is designed to ensure a minimum yearly return of interest for validators depicted as is listed below.

- Number of validators 4096: ROI = 7 percent
- Number of validators 8192: ROI = 4.95 percent
- Number of validators 12888: ROI = 3.9 percent
- Number of validators 16384: ROI = 3.496 percent
- Number of validators 20480: ROI = 3.127 percent

This minting process depends only on the number of staked tokens and is proportional to the square root of the number of validators. The number of overall minted tokens per year will be controlled by a distribution formula yielding the expected minimum number of tokens as follows.

$$M = c \cdot \sqrt{S}$$

The constant c is derived through the following reasoning. New tokens will be minted per each epoch and there are $N_e = 82181.25$ epochs per year like in Ethereum. Assuming the effective balance of each validator to be $EB_i = 8192$ tokens and the initial number of validators is $N_V = 4096$ and the target ROI is 7%, we can assume the expected number of newly issued tokens in gwei is to be

$$M_e = 4096 \cdot 8192 \cdot 10^9 \cdot 0.07$$

Hence we should have

$$\frac{c \cdot N_v \cdot 8192 \cdot 10^9}{\sqrt{N_v \cdot 8192 \cdot 10^9}} \cdot N_e = M_e$$

This argument yields to the constant value $c = 156$. This issuance process is similar to the Ethereum issuance logic with the single deviation that we start from a fixed expected ROI target for the initial numbers of starting validators.

2.6.2 Minting Based on the Validators Activty

This process incorporates the main logic of PoSA consensus algorithm and provides the main logic behind the activity rewarding process.
First, let's sum up all gas fees spent on all transactions on-chain and interpret the result as a sum of four independent arguments as

$$F = A + B + T + t$$

Here A is the gas usage of smart contracts linked with active validators (minus the constant part linked with each tx), B is the gas usage of smart contracts not associated with any validator, T is the aggregation of the constant gas usage argument equal to 21000, and t is the sum of tips of all considered transactions.

We will give tips t to the proposer in the Geth. The amount $A + B + T$ is totally burned in Geth (execution layer). In Prysm (consensus layer), the reward should be shared among the proposer and the committee. Step by step description is given below.

1. We will overall mint $M = m + P_m$ new tokens where $m = K \cdot \sqrt{D}$ and P_m should be equal to $A + T$. K is a specially selected constant which yields to the expecting ROI as discussed below.

2. In execution layer (Geth), we will burn $A + T + B$

3. In consensus layer (Prysm) we will meant the above mentioned $A + T$.

4. This means B amount will be burned totally, m will be created from scratch, which implies that supply equilibrium will be settled when B reaches to m.

5. Next, we can calculate what should be the base fee and the load of blocks resulting in equilibrium.

6. We can calculate what should be the base fee at 50 percent load resulting in equilibrium. Next, taking this number as a guide, we can figure out which base fee amount to start from, so the base fee change formula will lead to equilibrium at 50 percent load

7. Given an initial number of validators to be 4096, and the staked amount to be 4096 * 8192, to provide 7 percent, we can fix $m = \frac{4096 \cdot 8192 \cdot 7}{100}$ which in turn means we should mint $\frac{m}{2628000} = 0.8937$ FTN per block.

8. Remember that $m = c \times 64\sqrt{D}$ so we can calculate $c = \frac{4096 \cdot 8192 \cdot 0.07}{64 \cdot \sqrt{4096 \cdot 8192}} = 6.34$

9. Fixing the constant $c = 6.33$, this will

10. Note that the coefficient c in turn will depend on a few other factors and will encounter penalty strategies. This will impact the overall minting amount.

11. Base Fee Calculation Methodology: The goal is figuring out a formula, which depending on the block load will result in **meaningful** rewards. For half-load traffic (15M gas per block) we want transaction fees to be equal to 1/8 of the minted amount of m. $fee = 15M * base_fee$

In Ethereum, the burning rate is dynamically controlled based on the bandwidth. Each block capacity is 30M gas usage. If the block is half filled, the base gas fee is increased up to 12 percent compared to the price in the previous block. If the capacity is low, so the blocks are mostly empty, the base gas fee is decreased up to 12 percent. In our PoSA implementation we will follow to the same logic to maintain the base fee dynamics.

Apart of minting new token

Algorithm 2 Base Fee reference calculation

window_size = 1575(**window** = window_size × epoch)
epoch = 32 × block
for each epoch **do**

$\mathrm{f}_{\mathrm{epoch}} = 0$

Let $\mathrm{TX}^{\mathrm{contract}}_{\mathrm{epoch}} = \{\mathrm{tx}_1, \mathrm{tx}_2, \ldots, \mathrm{tx}_{k_i}\}$ be the given contract's transactions happened during that epoch

for $\mathrm{tx}_j \in \mathrm{TX}^{\mathrm{contract}}_{\mathrm{epoch}}$ **do**
    $f_{\mathrm{epoch}} = f_{\mathrm{epoch}} + f_{\mathrm{tx}_j}$
end for

if $i >$ window_size **then**
    $\mathrm{f}_{\mathrm{epoch}} = \mathrm{f}_{\mathrm{epoch}i-1} -$ window _size+ $\mathrm{f}_{\mathrm{epoch}}$
else
    $\mathrm{A}^{\mathrm{score}_i}_{\mathrm{contract}} = \sum_{j=1}^{i} A_{\mathrm{epoch}_i}$
end if

Output $\mathrm{A}^{\mathrm{score}_i}_{\mathrm{contract}}$
end for

Remember the proposer gets (A + T) and the committee of 128-4096 people will get 7/8 of the reward.

3 The Incentive Layer

3.1 Block Validation

3.2 Slashing and Validator Penalties

The system parameters in the consensus algorithm are

- **D** - The overall deposited amount
- **S** - The circulating supply
- **L** - Maximum number of validators
- **s** - Stake quote
- **B** - Burned amount
- **I** - Number of new issued tokens. In Ethereum, the issuance amount is calculated as $I = cF\sqrt{D}$.
- **y** - The yield. $y = \frac{I}{D}$
- **b** - Burn rate. Note that $b = \frac{B}{S}$
- **d** - Deposit rate: $d = \frac{D}{S}$

3.3 Rewards

The overall reward $R = M_1 + M_2$ is distributed among the proposer and sync committee according to the following formulas. First we define $M_t = (A + T)\frac{\sum P'_i}{\sum P_i}$ where the P_i and P'_i are the i-th validators power and effective power respectively. The proposer rewards is next calculated as

$$M_2 = M_t \left(\frac{2}{56} \cdot \frac{\sum SR_i}{\sum SR_m} + \frac{54}{56} \cdot \frac{\sum AR_i}{\sum AR_m}) \right)$$

TODO

- Proposer Rewards
- Sync Committee Rewards
- Attester Rewards
- Discuss the proposer rewards logic and how it gets less with a low quality attesters
- Burn in Geth and Mint in Prysm.
-

Base rewards

Validator base reward

Active balance is

$$B = \sum_{i=0}^{n} eb_i, \tag{1}$$

where B - active balance, n - number of validators and eb_i - effective balance of i'th validator.

Base reward per increment is

$$BR_{inc} = \frac{inc * f}{\sqrt{B}}, \tag{2}$$

where $inc = 1 * 1^9$ - effective balance increment in gwei and $f = 156$ - base reward factor.

Base reward of i'th validator is

$$BR_i = \frac{eb_i}{inc} * BR_{inc} = \frac{eb_i * f}{\sqrt{B}}, \tag{3}$$

where eb_i - effective balance of i'th validator.

Total base reward per epoch is

$$BR_{total} = \sum_{i=0}^{n} BR_i = \frac{B * f}{\sqrt{B}} \tag{4}$$

Proposer base reward

Total activity is

$$A = \sum_{i=0}^{n} ea_i \tag{5}$$

where ea_i - effective activity of i'th validator.

Base proposer reward is

$$BPR = \frac{(A + T) * bf}{W * n * gwei}, \tag{6}$$

where T - transactions gas, bf - average base fee during activity period (in wei), $W = 1575$ - activity period and $gwei = 1 * 1^9$.

Sync proposer reward

Sync committee reward weight depending on i'th sync committee member vote is

$$W_s(i) = \begin{cases} 2, & \text{if i'th validator voted} \\ 0, & \text{if i'th validator didn't vote} \end{cases} \tag{7}$$

We can calculate proposer reward per sync committee vote in two ways:

1. First method:

   Proposer reward per sync committee vote in block is

$$SPR_{vote}(i) = \frac{BPR * W_s(i)}{W_\Sigma * s * cs}, \tag{8}$$

   where $W_\Sigma = 56$ - reward weight denominator, $s = 32$ - slots per epoch and $cs = 512$ - sync committee size.

2. Second method:

   Proposer reward per sync committee vote in block is

$$SPR_{vote}(i) = \frac{BPR * W_s(i)}{W_\Sigma * cs}, \tag{9}$$

   where $W_\Sigma = 56$ - reward weight denominator and $cs = 512$ - sync committee size.

   In this case sync proposer reward is 32 times more then in first case.

Therefore, sync proposer reward per block is

$$SPR_{block}(j) = \sum_{i=0}^{cs} SPR_{vote}(i) \tag{10}$$

1. First method:

$$SPR_{block}(j) = \sum_{i=0}^{cs} \frac{BPR * W_s(i)}{W_\Sigma * s * cs}, \tag{11}$$

   where $W_\Sigma = 56$ - reward weight denominator, $s = 32$ - slots per epoch and $cs = 512$ - sync committee size.

2. Second method:

$$SPR_{block}(j) = \sum_{i=0}^{cs} \frac{BPR * W_s(i)}{W_\Sigma * cs}, \tag{12}$$

   where $W_\Sigma = 56$ - reward weight denominator and $cs = 512$ - sync committee size.

Eventually, sync proposer reward per epoch is

$$SPR_{epoch} = \sum_{j=0}^{s} SPR_{block}(j) \tag{13}$$

1. First method:

$$SPR_{epoch} = \sum_{j=0}^{s} \sum_{i=0}^{cs} \frac{BPR * W_s(i)}{W_\Sigma * s * cs}, \tag{14}$$

   where $W_\Sigma = 56$ - reward weight denominator, $s = 32$ - slots per epoch and $cs = 512$ - sync committee size.

2. Second method:

$$SPR_{epoch} = \sum_{i=0}^{s} \sum_{i=0}^{cs} \frac{BPR * W_s(i)}{W_\Sigma * cs}, \tag{15}$$

where $W_\Sigma = 56$ - reward weight denominator and $cs = 512$ - sync committee size.

So, maximum sync proposer reward per epoch is

1. First method:

$$MAX\_SPR_{epoch} = \frac{VW_s}{W_\Sigma} * BPR, \tag{16}$$

where $VW_s = 2$ - voted sync reward weight, $W_\Sigma = 56$ - reward weight denominator,

2. Second method:

$$MAX\_SPR_{epoch} = \frac{VW_s}{W_\Sigma} * BPR * s, \tag{17}$$

where $VW_s = 2$ - voted sync reward weight, $W_\Sigma = 56$ - reward weight denominator and $cs = 512$ - sync committee size.

Attestation proposer reward

Timely head reward weight for i'th validator is

$$W_{th}(i) = \begin{cases} 14, & \text{if i'th validator voted} \\ 0, & \text{if i'th validator didn't vote} \end{cases} \tag{18}$$

Timely source reward weight for i'th validator is

$$W_{ts}(i) = \begin{cases} 14, & \text{if i'th validator voted} \\ 0, & \text{if i'th validator didn't vote} \end{cases} \tag{19}$$

Timely target reward weight for i'th validator is

$$W_{tt}(i) = \begin{cases} 26, & \text{if i'th validator voted} \\ 0, & \text{if i'th validator didn't vote} \end{cases} \tag{20}$$

We have two methods:

1. First method:

   Attestation reward of i'th validator is

$$AR_i = BR_i * \frac{W_{th}(i) + W_{ts}(i) + W_{tt}(i)}{W_\Sigma}, \tag{21}$$

   where $W_\Sigma = 56$ - reward weight denominator.

   Number of attesters per block is

$$acs = \frac{n}{s} \tag{22}$$

where n - number of active validators and $s = 32$ - slots per epoch.

Therefore, total actual attestation reward in block is

$$AR_{block}(j) = \sum_{i=0}^{acs} AR_i = \sum_{i=0}^{acs} BR_i * \frac{W_{th}(i) + W_{ts}(i) + W_{tt}(i)}{W_\Sigma}, \quad (23)$$

and maximum attestation reward in block is

$$MAX\_AR_{block}(j) = \sum_{i=0}^{acs} BR_i * \frac{VW_{th} + VW_{ts} + VW_{tt}}{W_\Sigma}, \quad (24)$$

where $VW_{th} = 14$ - voted timely head reward weight, $VW_{ts} = 14$ - voted timely source reward weight, $VW_{tt} = 26$ - voted timely target reward weight and $W_\Sigma = 56$ - reward weight denominator.

Total epoch attestation reward in this case is

$$AR_{epoch} = \sum_{j=0}^{s} AR_{block}(j) = \sum_{j=0}^{s} \sum_{i=0}^{acs} BR_i * \frac{W_{th}(i) + W_{ts}(i) + W_{tt}(i)}{W_\Sigma}, \quad (25)$$

where $s = 32$ - slots per epoch.

So, attestation proposer reward in block is

$$APR_{block}(j) = \frac{AR_{block}(j) * (VW_{th} + VW_{ts} + VW_{tt})}{MAX\_AR_{block}(j) * W_\Sigma} * BPR, \quad (26)$$

and attestation proposer reward in epoch is

$$APR_{epoch} = \sum_{j=0}^{s} APR_{block}(j) \quad (27)$$

In case if all validators perform their duties, attestation proposer reward is

$$MAX\_APR_{epoch} = \sum_{j=0}^{s} \frac{(VW_{th} + VW_{ts} + VW_{tt})}{W_\Sigma} * BPR, \quad (28)$$

because $AP_{block}(j)$ is equal to $MAX\_AR_{block}(j)$.

2. Second method:

   In second method we don't take validators' base reward into account, so we can calculate only their performed duties

   Performed duties of i'th validator

$$PD(i) = \frac{W_{th}(i) + W_{ts}(i) + W_{tt}(i)}{W_\Sigma}, \quad (29)$$

   where $W_\Sigma = 56$ - reward weight denominator.

   Max performed duties by validator is

$$MAX\_PD = \frac{VW_{th} + VW_{ts} + VW_{tt}}{W_\Sigma}, \quad (30)$$

where $VW_{th} = 14$ - voted timely head reward weight, $VW_{ts} = 14$ - voted timely source reward weight, $VW_{tt} = 26$ - voted timely target reward weight and $W_{\Sigma} = 56$ - reward weight denominator.

Number of attesters per block is

$$acs = \frac{n}{s} \tag{31}$$

where n - number of active validators and $s = 32$ - slots per epoch.

Performed duties in block is

$$PD_{block}(j) = \sum_{i=0}^{acs} PD(i) = \sum_{i=0}^{acs} \frac{W_{th}(i) + W_{ts}(i) + W_{tt}(i)}{W_{\Sigma}}, \tag{32}$$

and maximum performed duties in block is

$$MAX\_PD_{block} = acs * MAX\_PD \tag{33}$$

Therefore, performed duties in epoch is

$$PD_{epoch} = \sum_{j=0}^{s} PD_{block}(j) = \sum_{i=0}^{acs} \frac{W_{th}(i) + W_{ts}(i) + W_{tt}(i)}{W_{\Sigma}}, \tag{34}$$

and maximum performed duties in epoch is

$$MAX\_PD_{epoch} = s*MAX\_PD_{block} = s*acs*MAX\_PD = n*MAX\_PD, \tag{35}$$

where n - number of active validators and $s = 32$ - slots per epoch.

So, attestation proposer reward in block in this case is

$$APR_{block}(j) = \frac{PD_{block}(j) * (VW_{th} + VW_{ts} + VW_{tt})}{MAX\_PD_{block} * W_{\Sigma}} * BPR, \tag{36}$$

and attestation proposer reward in epoch is

$$APR_{epoch} = \sum_{j=0}^{s} APR_{block}(j) \tag{37}$$

In case if all validators perform their duties, attestation proposer reward is

$$MAX\_APR_{epoch} = \frac{(VW_{th} + VW_{ts} + VW_{tt})}{W_{\Sigma}} * BPR * s, \tag{38}$$

because $PD_{block}(j)$ is equal to MAX_PD_{block}.

Minting

If all validators' duties are performed proposer reward per block is

$$MAX\_PR_{block} = MAX\_SPR_{block} + MAX\_APR_{block} \tag{39}$$

$$MAX\_PR_{block} = \frac{(VW_{th} + VW_{ts} + VW_{tt} + VW_s)}{W_\Sigma} * BPR = BPR \quad (40)$$

Maximum proposer reward per epoch is

$$MAX\_PR_{epoch} = MAX\_SPR_{epoch} + MAX\_APR_{epoch} \quad (41)$$

$$MAX\_PR_{epoch} = \frac{(VW_{th} + VW_{ts} + VW_{tt} + VW_s)}{W_\Sigma} * BPR * s = BPR * s \quad (42)$$

And total minted amount during epoch is

$$MINTED_{epoch} = MAX\_PR_{epoch} + BR_{total} = BPR * s + \frac{B * f}{\sqrt{B}} \quad (43)$$

References

[1] Ethereum Consensus Layer Documentation. https://eth2book.info/bellatrix/part2/incentives/penalties/

[2] Ethereum Technical Overview. The Incentive Layer. Slashing: https://eth2book.info/bellatrix/part2/incentives/slashing/