New dynamics of bonds and staking in the rebase token ecosystem

sc0t#1620 in Discord

@therealsc0t in Twitter

abenhamo@xtendplex.com

I. The Dynamics of Bonds

Traditional Finance Approach:

1. The user buys the bond at the spot price of $TOKEN (no discount at all)

2. When the user claims he gets = $e^{d*((T-t-1)/T)}$ number of TOKENs

d is the discounting rate for the whole period T

T is the maturity to have the full discount rate in hours.

t is the time to maturity, in hours.

1 is one hour

If the user buys the bond and claims immediately, he will get:

$$e^{d*((T-T-1)/T)} = e^{d*(-1/T)} < 1$$

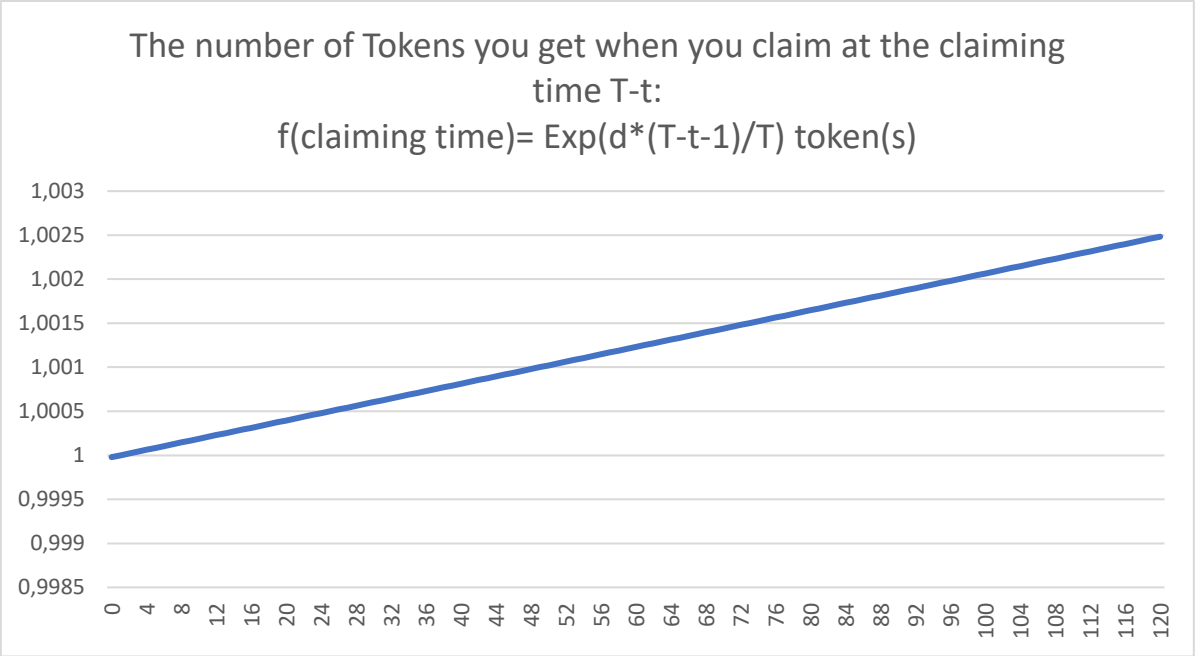
In other words, the more you leave, the bigger the exponential part gets to reach its maximum at maturity when t =0

$$e^{d*((T-0-1)/T)} = e^{d*((T-1)/T)} > 1$$

In summary, the number of claimed TOKENs: $e^{d*((T-t-1)/T)}$ is in the range [$e^{d*(-1/T)}$, $e^{d*((T-1)/T)}$].

Let's assume: d=5%, T=120 hours, then the number of claimed TOKENs will slide between:

[0,999583420126834, 1,05083315799603]



Pros: simple

Cons: it's linear, so users can start claiming right away when the claim > 1

New Innovative Approach:

Let's try now using another dynamic to incentivize users not to claim bonds. For this purpose, we will use a non-linear model, which will be steeper over time.

1. The user will buy the bond at the spot price of $TOKEN (no discount at all)

2. When the user claims he will get = $[\boldsymbol{a} * \left(\boldsymbol{d} * \frac{\boldsymbol{T-t-1}}{\boldsymbol{T}}\right)^{\boldsymbol{3}} + \boldsymbol{c}]$ number of TOKENs

d is the approximative discounting rate for the whole period T

T is the maturity to have the full discount rate in hours.

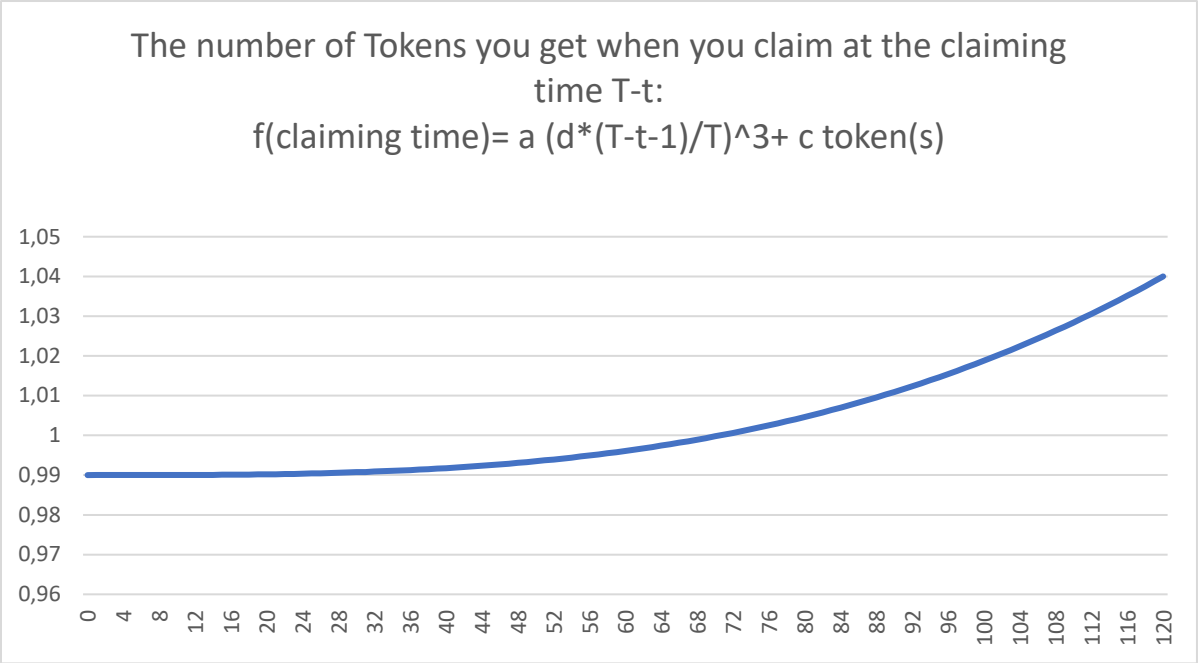
t is the time to maturity, in hours.

1 is one hour

a is a multiplier equal to $\frac{d}{[\left(\boldsymbol{d} * \frac{\boldsymbol{T-1}}{\boldsymbol{T}}\right)^{\boldsymbol{3}}]}$

c is a constant slightly < 1, in our example = 0.99

Here is the dynamic graph:



Pros: We can clearly see that if the user claims immediately, he loses money. Arb bots are out of the game if you buy and claim because that won't work. You just need to wait until the end of period T to get a juicy discount.

No need to be locked. A user will naturally buy and wait for T.

It is a natural anti-dumping model. In case of a sell-off, users will rush and claim tokens at a loss, which would trigger an increase in the treasury to buy back and burn tokens.

II. Dynamics of token staking:

Current OHM Approach: Shares dividend distribution as follows:

Assuming that the reward is distributed EXACTLY at the Rebase_Epoch ~ every 8 hours = 480 minutes.

The user stakes their tokens just before the Rebase_epoch to get a full reward, then unstakes immediately to sell or wait for the next Rebase_epoch, which could trigger a selloff after each rebase.

New Linear Approach:

Assuming that the reward is distributed EXACTLY at the Rebase_Epoch ~ every 8 hours = 480 minutes.

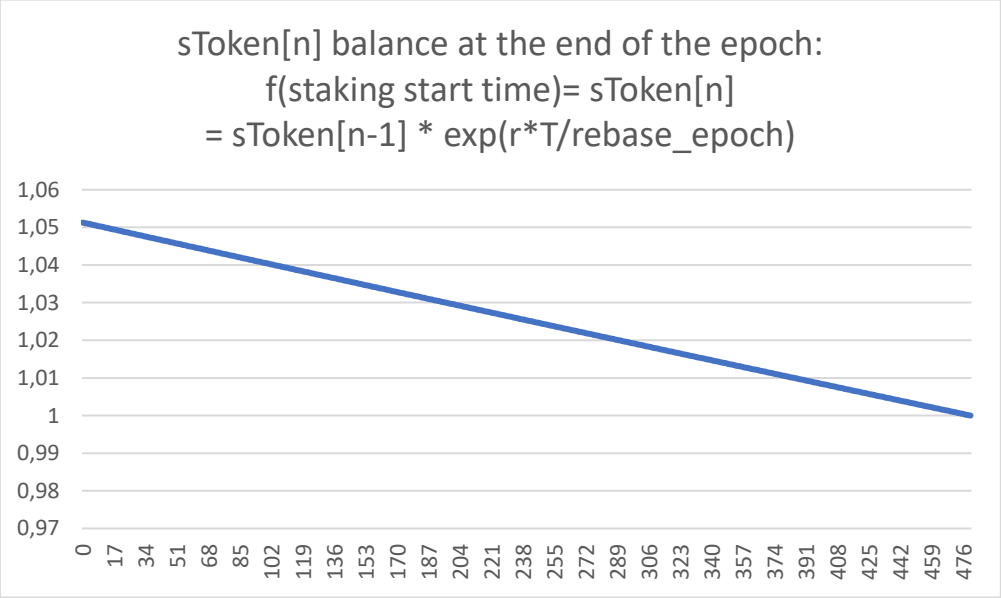
1. The users stake their TOKEN at time t to get sTOKEN

2. At the Rebase_Epoch, the user's new sTOKEN balance will be equal to $e^{r*T/\text{rebase\_epoch}}$

t: staking start time in [0,Rebase_Epoch]
T: Total staking time = (Rebase_Epoch -t) in [0,Rebase_Epoch]

r: the reward rate computed by the rebase system.

Here is the dynamic graph:



Pros: Users that usually stake just before the Rebase_epoch are out of the game.

Cons: as it is linear, users can unstake just after the Rebase_Epoch(n-1) and stake later in the middle of the next cycle to get a reward proportional to the total staking time.

New Innovative Approach:

Assuming that the reward is distributed EXACTLY at the Rebase_Epoch ~ every 8 hours = 480 minutes.

1. The user stake their TOKEN at time t to get sTOKEN

2. At the Rebase_Epoch, the user's new sTOKEN balance will be equal to

$$[\boldsymbol{a} * \left(\boldsymbol{r} * \frac{\boldsymbol{T}}{\boldsymbol{rebase\_epoch}}\right)^{\boldsymbol{3}} + \boldsymbol{c}] \text{ sTOKEN(s)}$$

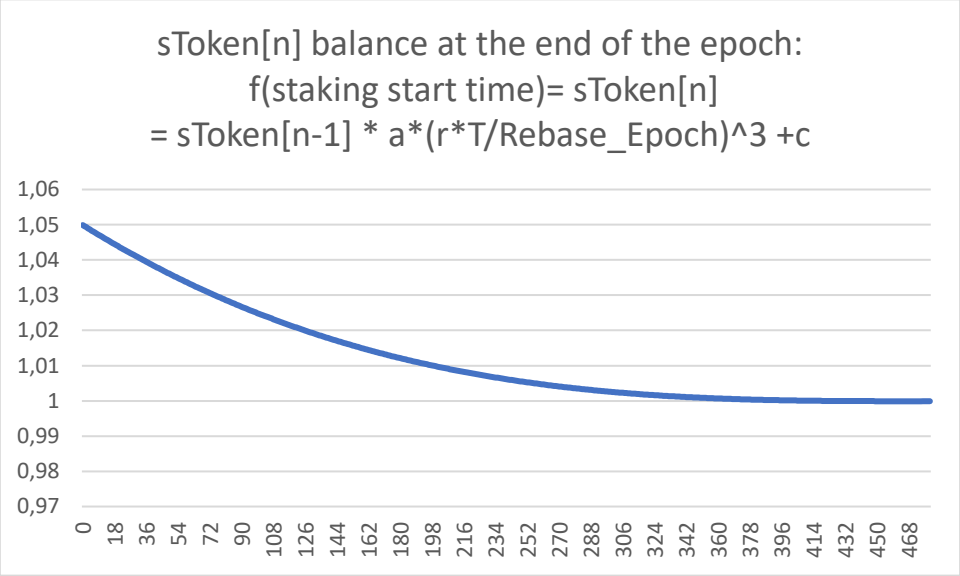
t: staking start time in [0,Rebase_Epoch]
T: Total staking time = (Rebase_Epoch -t) in [0,Rebase_Epoch]

r: the reward rate computed by the rebase system.

a is a multiplier equal to $\frac{1}{(r)^2}$

c is a constant equal to 0.9999

Here is the dynamic graph:



Pros: Users that usually stake just before the Rebase_epoch are out of the game.

It incentivizes users to stay staked all the time.

If a user unstakes just after the Rebase_epoch[n-1] to get in later, it will be too late as there is a steeper decrease in reward.