Compounding Decreasing LQTY Deposit and Corresponding ETH Gain

Scalability of Ethereum dApps is constrained by the gas costs of writing to storage.

Here we derive formulas for a scalable implementation of a compounding, decreasing Stability Pool deposit and its corresponding ETH gain. Thus, Liquity tracks the deposits and rewards of every depositor with only O(1) complexity for each liquidation.

In previous work, Batog[1] and Solmaz[2] solved the problem of scalable reward distribution in a staking pool, distributing rewards in proportional to a staker's share of total stakes with a "pull" based approach. Only a single running sum need be updated at each reward, sidestepping the need for an expensive O(n) storage update for *n* stakers. However, their work stopped short of solving the problem for a compounding stake.

For a pull-based implementation in Liquity's Stability Pool, we must factor out the initial deposit, and track reward terms that are independent from the *individual* deposit values. We derive formulas for a compounding LQTY deposit and corresponding ETH gain as functions of the initial deposit, and involving a product P and a sum S, which are in turn purely functions of the LQTY losses, ETH rewards and the *total* deposits.

Terms

$\mathbf{d_i}$: A given user's LQTY deposit at liquidation event *i*
$\mathbf{q_i}$: The LQTY debt absorbed by the Stability Pool from liquidation *i*
$\mathbf{e_i}$**:** The ETH sent to the Stability Pool from liquidation *i*
$\mathbf{D_i}$: Total LQTY deposits at liquidation *i*

Let $\mathbf{d_0}$ represent the user's initial Stability Pool deposit. After the first liquidation, their new deposit $\mathbf{d_1}$ is given by:

1) $\mathbf{d_1 = d_0 - LQTYLoss}$

Liquidation "earns" a LQTY loss for the deposit proportional to the prior share of total deposits:

2) $\mathbf{d_1 = d_0 - (q_1 * d_0/D_0)}$

3) $\mathbf{d_1 = d_0(1 - q_1/D_0)}$

Since the deposit compounds, the deposit value after liquidation 2 is a function of the previous deposit value:

4) $d_2 = d_1 * (1- q_2/D_1)$

and substituting equation 2):

5) $d_2 = d_0 * (1-q_2/D_1) * (1-q_1/D_0)$

Similarly:

6) $d_3 = d_2 * (1-q_2/D_1)$

7) $d_3 = d_0 * (1-q_3/D_2) * (1-q_2/D_1) * (1-q_1/D_0)$

And the general case for a compounded deposit:

8) $d_n = d_0 * PROD^n_{i=1} (1-q_i / D_{i-1})$

Deposits made at t > 0

For a deposit made between liquidations [*t, t+1*], which is withdrawn between liquidations [*n, n +1*], rewards are earned from liquidations *t+1*, *t+2*, …, *n*.

We take snapshot of the product at liquidation *t*, and calculate the compounded stake at liquidation *n* as:

9) $d_n = d_t * PROD^n_{i=1} (1-q_i / D_{i-1}) / PROD^t_{i=1} (1-q_i / D_{i-1})$

i.e. labelling the product term ‘P’:

10) $d_n = d_t * P / P_0$

Where **P** is the current product, and P_0 is the snapshot of the product at the time when the user made the deposit.

Corresponding ETH Gain

The LQTY deposit earns an ETH gain at each liquidation.

At each liquidation *i*, the user’s LQTY deposit effectively decreases. Thus we can write the user’s corresponding cumulative ETH gain, from a series of liquidations, as:

11) $E_n = d_0 * (e_1 / D_0) + d_1 * (e_2 / D_1) + d_2 * (e_3 / D_2) + ... + d_{n-1} * (e_n / D_{n-1})$

Then, using our expression for the deposit from equation 8):

12) $E_n = d_0 * (e_1 / D_0) * PROD^1_{i=1} (1-q_i / D_{i-1}) +$
$d_0 * (e_2 / D_1) * PROD^2_{i=1} (1-q_i / D_{i-1}) +$
$d_0 * (e_3 / D_2) * PROD^3_{i=1} (1-q_i / D_{i-1}) + \ldots +$
$d_0 * (e_n / D_{n-1}) * PROD^n_{i=1} (1-q_i / D_{i-1})$

And factoring out the initial deposit, d_0:

13) $E_n = d_0 * SUM^n_{k=1} [(e_k / D_{k-1}) * PROD^k_{i=1} (1-q_k / D_{k-1})]$

Thus, we have E_n as a function of the initial deposit d_0, and the rewards and total deposits at each liquidation.

Let the summation $SUM^n_{k=1} [\ldots]$ be denoted **S**.

Deposits made at t > 0

For a deposit made between liquidations [*t, t+1*], which is withdrawn between liquidations [*n, n +1*], rewards are earned from liquidations *t+1*, *t+2*, …, *n*.

To account for the deposit entering after liquidations have already begun, we correct the product terms in 13) to $P^k_{i=1} / P_0$, as per 10).

We take a snapshot of the sum at liquidation *t*, labelling it S_0**.** Then, the ETH gain earned by the deposit at liquidation *n* is:

14) $E_n = d_t * (S - S_0)/P_0$

Implementation

Making a deposit:
Record deposit: deposit[user] = $\mathbf{d_0}$
Update total deposits: $\mathbf{D = D + d_0}$
Record product snapshot: $\mathbf{P_0 = P}$
Record sum snapshot: $\mathbf{S_0 = S}$

Upon each liquidation yielding LQTY debt 'q' offset with the Stability Pool and ETH gain 'e':
Update S: $\mathbf{S = S + (e / D) * P}$ *(intuition: a deposit's marginal ETH gain is equal to the deposit * ETH per unit staked * current correction factor)*
Update P: $\mathbf{P = P * (1- (q / D))}$
Update total deposits: $\mathbf{D = D - q}$

Withdrawing the deposit and ETH gain:
Compute final compounded LQTY deposit d: $\mathbf{d = d_0 * P / P_0}$
Compute final corresponding cumulative ETH gain E: $\mathbf{E = d_0 * (S - S_0) / P_0}$
Send **d** and **E** to user
Update deposit: deposit[user] = 0
Update total deposits: $\mathbf{D = D - d}$

References

[1] B. Batog, L. Boca, N. Johnson, *"Scalable Reward Distribution on the Ethereum Blockchain",* 2018. http://batog.info/papers/scalable-reward-distribution.pdf

[2] O. Solmaz, *"Scalable Reward Distribution with Changing Stake Sizes",* 2019. *https://solmaz.io/2019/02/24/scalable-reward-changing/*