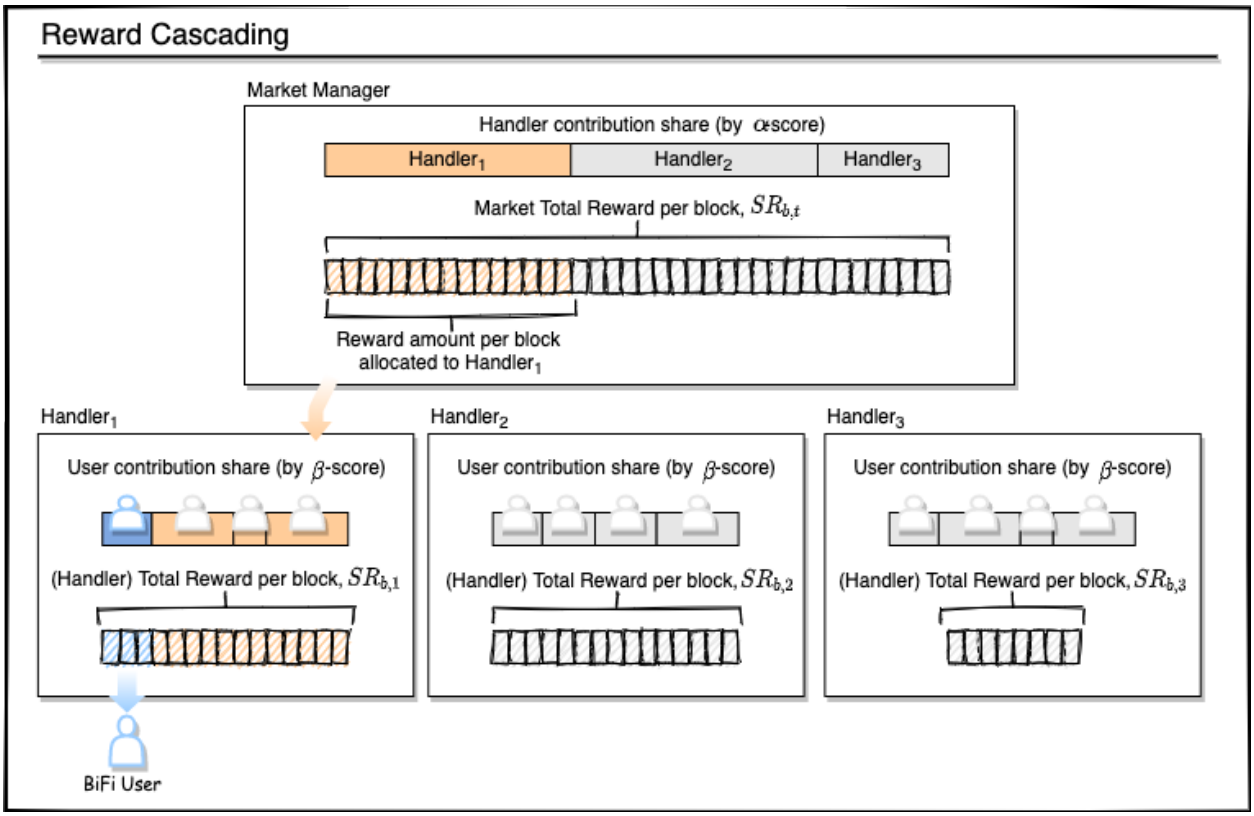
(KOR) Appendix I: Reward Method for Deposit and Borrow

1. Overview

BiFi는 사용자들의 예금량과 대출량을 인자로 계산된 실시간 이자율을 사용한다. 그 결과, 예금량이 많으면 이율이 떨어져서 대출을 장려하게 되고, 대출량이 많으면 이율이 올라가서 예금을 장려하게 된다. 따라서 사용자의 예금과 대출은 Token 유동성을 제공하는 것뿐만 아니라 Bifi 시스템을 균형있게 유지시키는 작용을 한다.



BiFi는 예치와 대출을 수행한 사용자에게 SI Token을 보상으로 지급한다. 각 Token에 대한 보상량은 Token Handler의 α-score에 의해 산출된 Reward Parameters(위 그림에서 $SR_{b,i}$)로 할당된다.
($Appendix III$: *Reward method for Reward Parameter Update* 참고)

각 Token Handler에 할당된 보상량은 다시 사용자의 예금 및 대출량으로 산출된 β-score에 따라 사용자들에게 분배된다. 즉, 매 블록마다 사용자들은 전체 β-score 대비 자신의 β-score 비율대로 보상을 지급 받는다.

Terminology

SI Token 보상량 산출 및 지급은 BiFi의 각 Token Handler가 수행하며 다음 변수들은 Token Handler마다 별도로 보유한다.

Variables for β-score Calculation

- SR_b: β-score 기반으로 지급되는 블록당 SI Token 수량
- R_u: 매 블록에서 1 β-score 당 지급되는 SI Token 수량
- R_{user}: 해당 블록에서 사용자에게 제공되는 SI Token 수량
- L: 모든 블록에 대한 R_u 값의 누적합
- β: β-score 계산에 사용되는 가중치
- S_t: 모든 user의 β-score 총량
- S_{user}: user의 β-score
- H_L: Handler에서 최근 action이 실행된 block height
- H_C: 현재 transaction의 block height

2. Distribution Method

2.1 User β-score

$$Score = \beta * (Deposit\ amount) + (1 - \beta) * (Borrow\ amount)$$

사용자의 β-score는 위 식과 같이 정의한다. 가중치(β)에 의해 예금량과 대출량이 Score에 미치는 영향을 조절할 수 있다.

2.2 Reward Distribution Method

매 블록마다 각 사용자의 SI Token 보상량은 다음과 같이 산출된다.

$$R_{user} = SR_b * \frac{S_{user}}{S_t}$$

각 Handler는 매 블록마다 총 SR_b개의 SI Token을 사용자들의 β-score 비율대로 분할하여 보상으로 제공한다.

3. Implementation

사용자 action에 의해 계산되는 R_u값은 매 블록마다 산출되어야 한다. 하지만 모든 블록에서 사용자들이 action을 실행한다는 가정은 현실적이지 않다. 게다가 action이 실행되지 않은 기간에서 보상을 계산하기 위해 모든 블록의 β-score 변화를 저장해야한다. 이것은 smart contract의 데이터 비효율을 야기한다. 따라서 "단위 β-score당 보상량"의 누적곱 (`RewardLane`)을 운영한다.

3.1 `RewardUnit`, R_u

Definition

각 블록에서 1 β-score 당 지급되는 SI Token 보상량, $R_u = \frac{SR_b}{S_t}$

Usage

- 보상량 산출식: $R_{user} = SR_b * \frac{S_{user}}{S_t} = S_{user}\frac{S_b}{S_t} = S_{user}\frac{s_b}{S_t} = S_{user} * R_u$
- 매 블록마다 사용자 β-score(S_{user})와 R_u를 곱하면 해당 블록에서의 보상량을 얻을 수 있다.

3.2 RewardLane, L

Definition

모든 지난 블록의 RewardUnit(R_u) 누적 합, $L = \sum_i R_u$

Requirement: L에 밀린 R_u 가산 시 R_u가 S_t에 따라 결정되는 값임을 고려해야 한다.

Batch Update for Inactive Periods

현재 사용자 action을 적용하기 전에 BiFi 상태는 이전 사용자 action이후 모든 블록에서 업데이트가 수행되어야 한다. 사용자 action이 없으면 R_u가 변경되지 않으므로 사용자 action

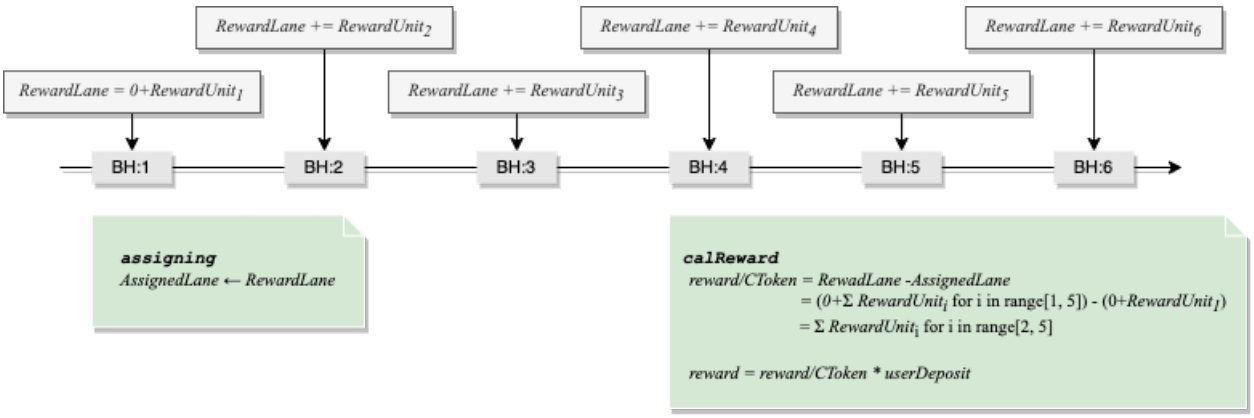
이 요청되지 않은 기간에 대한 업데이트는 간단하다. 따라서 현재 action은 현재 R_u 값과 이전 action부터 경과된 시간(block time)으로 업데이트 할 수 있다.

Update Process

1. 경과 시간(block height) 계산 : $\delta = H_C - H_L$
2. `RewardLane` 업데이트: $L = L + \delta * R_u$
3. 마지막 block height 업데이트: $H_L = H_C$

- Note: 한 블록에 여러 action이 포함된 경우 첫번째 action만 L을 업데이트하면 충분하다.

3.3 `AssignedLane` and Reward Calculation



Handler는 Bifi에 처음 접근한 사용자와 이미 보상을 지급 받은 사용자에게 현재 L 값을 할당한다. 이 AssingedLane과 L값을 활용하면, 이후 그 사용자의 action이 언제 발생하더라도 Handler는 해당 사용자의 보상량을 다음과 같이 계산하여 보상을 지급할 수 있다.

$$reward = S_{user} * (L - AssignedLane)$$

"$L - AssignedLane$" 값은 사용자의 이전 action과 현재 action 사이 모든 Block에서의 R_u의 누적 합을 의미하므로 사용자는 밀린 보상을 한번에 수령할 수 있다.