Fully Automatic Dynamic Reward Allocation Formula (FADRA15)

RodionKB

December 2, 2024

Fully Automatic Dynamic Reward Allocation Formula (FADRA15) with Minimum reward, set to $0.15 \cdot T_{\text{reward}}$ to ensure fair distribution for smaller holders.

$$R_i = \max\left(0.15 \cdot T_{\text{reward}}, \min\left(T_{\text{reward}} \cdot 0.999,\ T_{\text{reward}} \cdot \frac{T_i \cdot (1 + \beta_i - \alpha_i) \cdot (1 + H_{\text{holding}}) \cdot S_{\text{activity}}}{\sum_j T_j \cdot (1 + \beta_j - \alpha_j) \cdot (1 + H_{\text{holding}}) \cdot S_{\text{activity}}}\right)\right)$$

Overview

This document provides the implementation instructions for the Fully Automatic Dynamic Reward Allocation Formula(FADRA15) in a Solana-based token ecosystem. It covers features such as dynamic reward distribution, transaction limits, and fallback mechanisms.

Components of the Formula

- T_{reward}: Total Reward Pool, dynamically updated based on transaction fees.
- R_{min}: Minimum reward, set to $0.15 \cdot T_{\text{reward}}$ to ensure fair distribution for smaller holders.
- T_i: Tokens held by participant i.
- β: Progressive bonus, incentivizing smaller holders:

$$\beta_i = \beta_{\text{min}} + (\beta_{\text{max}} - \beta_{\text{min}}) \cdot \left(1 - \frac{D_i}{D_{\text{max}}}\right)$$

- α: Regressive penalty, balancing larger holders:

$$\alpha_i = \alpha_{\text{min}} + (\alpha_{\text{max}} - \alpha_{\text{min}}) \cdot \frac{D_i}{D_{\text{max}}}$$

  exemple:

  function calculateDynamicParameters(uint256 rewardPool, uint256 totalActivity) public

  betaMin = baseBetaMin * rewardPool / targetRewardPool;

  betaMax = baseBetaMax * rewardPool / targetRewardPool;

  alphaMin = baseAlphaMin * targetActivity / totalActivity;

  alphaMax = baseAlphaMax * targetActivity / totalActivity;

- **To ensure fair redistribution** in case of shortfalls, the formula for adjusting the regressive penalty (α_i) dynamically incorporates the total share of large holders and the shortfall:

$$\alpha_i = \alpha_i + \frac{\text{TotalLargeHolderShare}}{\text{Shortfall}}$$

  Explanation of Parameters

- α_i: Current regressive penalty for large holder i.
- TotalLargeHolderShare: Total proportional share of tokens held by all large holders.
- Shortfall: The deficit in the Reward Pool required to meet minimum rewards for smaller holders.

Purpose

This adjustment ensures:

- Large holders contribute proportionally to address any shortfall in the Reward Pool.
- Smaller holders are guaranteed their minimum rewards without depleting the Reward Pool entirely.
- H_{holding}: Holding multiplier, rewarding long-term retention:

$$H_{\text{holding}} = \min\left(\frac{t}{t_{\max}}, 1\right)$$

  uint256 public tMax = 365 days;
  uint256 holdingTime = block.timestamp - participantLastTransactionTimestamp;
  uint256 dynamicTMax = baseTMax * activeParticipants / targetParticipants;
- S_{activity}: Activity multiplier, incentivizing frequent engagement:

$$S_{\text{activity}} = \frac{\text{UserTransactions}}{\text{AverageTransactions}}$$

  $\sum_j$: Sum of all participants' weighted tokens.

  With Possibilites to makes **3 groups** of holders(**small,medium,large**). Depends if everything goes well during the tests.

Core Requirements

- **Dynamic Reward Pool Management:**
  - Reward Pool is funded with 2% from each transaction and redistributed proportionally,maximum send to all holders 99.9%. Rest (0.1) Goes to Market wallet after 3 months.
  - Rewards are distributed twice a day at 12 PM (US) and 12 PM (Europe), or immediately if necessary(check this during tests).
- **Fee Structure:**
  - 2% for the Liquidity Pool (LP).
  - 2% for the Reward Pool.
  - 0.85% for the Marketing Wallet.

Reward Pool Management

- **Reward Pool Update:** Automatically grows with every transaction:

$$T_{\text{reward}}^{\text{new}} = T_{\text{reward}}^{\text{old}} + 0.02 \cdot \text{Transaction Volume}$$

- **Fallback Condition:** If less than 1000 transactions occur within 3 months, all LP and Reward Pool are transferred to the marketing wallet.

Fallback Mechanisms

- If $T_{\text{reward}} < \text{MinRewardPool}$, the system reduces bonuses and penalties:

$$\beta = \beta \cdot \frac{T_{\text{reward}}}{\text{MinRewardPool}}$$

- If activity drops below a critical threshold, payouts occur less frequently.
- Linked wallets are aggregated:

$$\text{AggregateShare} = \sum_{k \in \text{LinkedWallets}} \text{Share}_k$$

Key Features

- **Dynamic Bonuses and Penalties:** Automatically adjusted based on participant size and proportional share.
- **Minimum Rewards:** Ensures smaller holders receive a fair share of rewards.
- **Sustainability:** Caps rewards at $T_{\text{reward}} \cdot 0.999$ to avoid exceeding the available pool.
- **Incentives for Activity:** Rewards participants for frequent engagement through S_{activity}.
- **Long-Term Holding Incentives:** Encourages retention with H_{holding}.
- If $T_{\text{reward}} < \text{MinRewardPool}$, the system reduces bonuses and penalties:

$$\beta = \beta \cdot \frac{T_{\text{reward}}}{\text{MinRewardPool}}$$

- If activity drops below a critical threshold, payouts occur less frequently.
- Linked wallets are aggregated:

$$\text{AggregateShare} = \sum_{k \in \text{LinkedWallets}} \text{Share}_k$$

Instructions for Developer

- Implement the reward distribution logic based on the FADRA15 formula.
- Ensure all parameters are dynamically calculated based on current ecosystem conditions.
- Enforce transaction limits and locking mechanisms. Test with low reward pool and high reward pool.
- (?)Configure fallback mechanisms for Liquidity Pool (LP) and Reward Pool management.

TESTS STARTS:

The system automatically adjusts the following parameters based on real-time ecosystem conditions:

- **Reward Pool (T_{reward}):** Dynamically grows with transaction fees.
- **Activity Metrics (S_{activity}):** Determined from user transactions relative to the ecosystem average.
- **Dynamic Bonuses and Penalties:**

$$\beta_i = \beta_{\min} + (\beta_{\max} - \beta_{\min}) \cdot \left(1 - \frac{D_i}{D_{\max}}\right)$$

$$\alpha_i = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min}) \cdot \frac{D_i}{D_{\max}}$$

where $\beta_{\min}$, $\beta_{\max}$, $\alpha_{\min}$, $\alpha_{\max}$ are dynamically computed.

Tests to Conduct

1. Reward Pool Extremes

- **Low Reward Pool:**
- Verify minimum rewards are distributed to small holders.
- Ensure large holders' penalties (α) proportionally address any shortfalls.
- **High Reward Pool:**
- Validate that small holders benefit proportionally through dynamic bonuses (β).
- Ensure total rewards do not exceed $T_{\text{reward}} \cdot 0.999$.

2. Activity Scenarios

- **High Activity:**
- Verify frequent participants are rewarded appropriately through S_{activity}.
- Validate the scaling of H_{holding} for long-term holders.
- **Low Activity:**
- Ensure rewards adapt to reduced activity levels.
- Check that payout frequency decreases if activity falls below the threshold.
- Validate that daily selling limits (30% of stack) are applied correctly.

4. Linked Wallets

- Test if the system aggregates shares for linked wallets:

$$\text{AggregateShare} = \sum_{k \in \text{LinkedWallets}} \text{Share}_k$$

- Ensure rewards are distributed proportionally.

Fallback Mechanisms

- Verify that when T_{reward} falls below the threshold, bonuses and penalties scale down:

$$\beta = \beta \cdot \frac{T_{\text{reward}}}{\text{MinRewardPool}}$$

- Test the transfer of Liquidity Pool and Reward Pool to the marketing wallet if fewer than 1000 transactions occur in 3 months.

Test Recommendations

- Conduct edge-case simulations with both low and high Reward Pool levels.
- Include scenarios with high and low user activity to ensure parameter adaptability.
- Perform stress tests with varying transaction volumes to validate dynamic scaling.
- Validate fallback mechanisms under failure scenarios.

  - Test with low Reward Pool to ensure minimum rewards are guaranteed for smaller holders.
  - Test with high Reward Pool to validate that total rewards do not exceed $T_{\text{reward}} \cdot 0.999$.

* **Activity Scenarios:**
  - Validate high activity scenarios where frequent participants receive proportional bonuses.
  - Validate low activity scenarios where payout frequency is reduced.

* **Linked Wallet Aggregation:** Test aggregation of rewards for linked wallets:

$$\text{AggregateShare} = \sum_{k \in \text{LinkedWallets}} \text{Share}_k$$

* **Stress Testing:** Simulate high transaction volumes to verify the stability of dynamic scaling.
* **Edge Cases:**
  - Test extreme small/large holder distributions.
  - Validate fallback mechanisms when activity thresholds are unmet.

Dynamic Parameter Adjustments

* **Dynamic Calculation of β and α:**

$$\beta_{\text{min}} = \text{BaseBetaMin} \cdot \frac{T_{\text{reward}}}{\text{TargetRewardPool}}$$

$$\beta_{\text{max}} = \text{BaseBetaMax} \cdot \frac{T_{\text{reward}}}{\text{TargetRewardPool}}$$

$$\alpha_{\text{min}} = \text{BaseAlphaMin} \cdot \frac{\text{TargetActivity}}{\text{TotalActivity}}$$

$$\alpha_{\text{max}} = \text{BaseAlphaMax} \cdot \frac{\text{TargetActivity}}{\text{TotalActivity}}$$

  exemples: Base Parameters:
  - $\text{BaseBetaMin} = 0.02$: Default progressive bonus lower bound.
  - $\text{BaseBetaMax} = 0.15$: Default progressive bonus upper bound.
  - $\text{BaseAlphaMin} = 0.01$: Default regressive penalty lower bound.
  - $\text{BaseAlphaMax} = 0.1$: Default regressive penalty upper bound.
* **Dynamic Holding Thresholds:**

$$H_{\text{holding}} = \min\left(\frac{t}{t_{\text{max}}}, 1\right)$$

  where t_{max} dynamically scales with the number of active participants.

Conclusion

This enhanced formula introduces robust dynamic adaptability to ensure fairness and sustainability across varying ecosystem conditions. Testing the recommended scenarios will validate the system's integrity and efficiency.

Copyright Notice