analysis

January 26, 2022

This notebook compares the new and old NFT Worlds' staking formulas. There is 1.75 billion $WRLD reserved for staking rewards, to be distributed over a five year period. This defines a constant rate of rewards R. Each world owner will receive a fraction of these rewards based on the number of worlds they own as well as their rarity.

The rate of rewards as a function of world rarity is defined below. The new formula is a weighted average among all worlds that are staked at any one time. This means that the number of worlds staked **and** the rarity of all staked worlds will influence the amount world owners will receive. The new formula ensures that at the end of five years, all staking rewards will have been distributed. The old formula (if it's correct) would have depleted the $WRLD staking reserves in 3.8 years if every world was staked.

$$Rewards_{r,old} = (200 - ((r - 1) * 0.015)$$

$$Rewards_{r,new} = \frac{w(r)}{\sum_{r' \in S} w(r')} \cdot R = \frac{40,003 - 3 * r}{\sum_{r' \in S} 40,003 - 3 * r'} \cdot R$$

Where r is the rank of the world, and S is the set of all staked worlds.

```python
import numpy as np
import pandas as pd
```

```python
def calc_old_reward(rank):
    return (200 - ((rank - 1) * 0.015))


r = sum(calc_old_reward(np.arange(1, 10001))) # wrld per day distributed if all
 ↪are staked
years = 1.75e9 / r / 365.0
print("$WRLD staking reserves run out in {:.1f} years using the old formula if
 ↪every world was staked".format(years))
```

```
$WRLD staking reserves run out in 3.8 years using the old formula if every world
was staked
```

With the new formula, the exact reward rate depends on not only how many worlds are staked, but the rarity of the staked worlds. While this seems overly complicated, it ensures that all staking rewards are distrubted in five years.

We'll calculate the new reward rate for different rarities, the fraction of worlds that are staked, and the rarity of the non-staked worlds. For example, if 50% of worlds are staked and the other 50% that are not staked are the 50% most rare worlds (rank 1 - 5,000) this will increase the staking rewards compared to if the 50% least rare worlds were not staked (rank 5,001-10,000). Each world

that is not staked leaves more rewards in the staking pool and since rarer worlds have more weight, when rarer worlds are not staked, rewards increase even more.

```python
R = 1.75e9 / 5.0 / 365.0 # reward rate ($WRLD per day)
def get_staked_ranks(frac_staked : float, listed_region : str):
    """
    Return array of ranks [1, 10001] of staked worlds based on the fraction of␣
    ↪worlds
    that are staked, and three scenarios specified by 'listed_region':
        least_rare: least rare worlds are not staked (listed on OS)
        mid_rare: the middle rare worlds are not staked
        most_rare: the most rare worlds are not staked
    """
    assert frac_staked >= 0 and frac_staked <= 1, "frac_staked must be in [0,␣
    ↪1]"

    ranks = np.arange(1, 10001)
    n = len(ranks)
    n_staked = int(n*frac_staked)

    if listed_region == "least_rare":
        return ranks[:n_staked]
    elif listed_region == "mid_rare":
        return np.concatenate((ranks[: int(n_staked/2)], ranks[ int(n-n_staked/
    ↪2):]))
    elif listed_region == "most_rare":
        return ranks[n-n_staked:]
    else:
        raise ValueError("listed region must be 'least_rare', 'mid_rare', or␣
    ↪'most_rare'")


def new_weights(rank):
    return 40003 - 3 * rank


def calc_new_reward(rank2calc, frac_staked, listed_region):
    staked_ranks = get_staked_ranks(frac_staked, listed_region)
    return new_weights(rank2calc) / np.sum(new_weights(staked_ranks)) * R
```

```python
ranks2calc = np.array([1, 5000, 10000]) # most rare, middle rare, least rare
old_rewards = calc_old_reward(ranks2calc)
fracs_staked = [.5, .75, 1]
listed_regions = ["least_rare", "mid_rare", "most_rare"]
data = np.zeros((len(ranks2calc)*len(listed_regions)*len(fracs_staked), 4),␣
 ↪dtype=object)
count = 0
for frac_staked in fracs_staked:
```

```
    for region in listed_regions:
        new_rewards = calc_new_reward(ranks2calc, frac_staked, region)
        b_idx = count*len(ranks2calc)
        e_idx = (count+1)*len(ranks2calc)
        data[b_idx:e_idx, 0] = ranks2calc
        data[b_idx:e_idx, 1] = frac_staked
        data[b_idx:e_idx, 2] = region
        data[b_idx:e_idx, 3] = new_rewards
        count += 1

df = pd.DataFrame(data=data, columns=["Rank", "Fraction Staked", "Listed␣
 ↪Worlds", "New Daily Rewards"])
df["Old Daily Rewards"] = calc_old_reward(df["Rank"])
```

The table below shows the new daily rewards as a function of the world's rank, the fraction of overall worlds that are staked, and rarity of the non-staked worlds (listed worlds). The old rewards are listed for comparison, but keep in mind that they would have run out in 3.8 years if all worlds were staked.

We believe the most likely scenario will consist of 75% of worlds staked with the remaining 25% non-staked worlds consiting of the "mid-rares" (in this scenario there is no correlation between a world's rank and its likelihood to be staked, which may not be true). This scenario corresponds to rows 12-14 below; one can see that the new rewards are slightly higher than the old rewards **and** they will last for five years.

```
[ ]: display(df)
```

```
      Rank  Fraction Staked  Listed Worlds  New Daily Rewards  Old Daily Rewards
0        1             0.5     least_rare         236.027041              200.0
1     5000             0.5     least_rare         147.534603             125.015
2    10000             0.5     least_rare          59.024462              50.015
3        1             0.5       mid_rare         306.830905              200.0
4     5000             0.5       mid_rare         191.792328             125.015
5    10000             0.5       mid_rare          76.730739              50.015
6        1             0.5      most_rare         438.318594              200.0
7     5000             0.5      most_rare         273.981995             125.015
8    10000             0.5      most_rare         109.612522              50.015
9        1            0.75     least_rare         177.874381              200.0
10    5000            0.75     least_rare         111.184828             125.015
11   10000            0.75     least_rare          44.481936              50.015
12       1            0.75       mid_rare         204.553937              200.0
13    5000            0.75       mid_rare         127.861552             125.015
14   10000            0.75       mid_rare          51.153826              50.015
15       1            0.75      most_rare         240.649142              200.0
16    5000            0.75      most_rare         150.423763             125.015
17   10000            0.75      most_rare          60.180334              50.015
18       1               1     least_rare         153.415453              200.0
19    5000               1     least_rare          95.896164             125.015
20   10000               1     least_rare          38.365369              50.015
```

21	1	1	mid_rare	153.415453	200.0
22	5000	1	mid_rare	95.896164	125.015
23	10000	1	mid_rare	38.365369	50.015
24	1	1	most_rare	153.415453	200.0
25	5000	1	most_rare	95.896164	125.015
26	10000	1	most_rare	38.365369	50.015