Determining Rewards for a Post

After a message is published, users (including curators, beneficiaries and author of this post) may be interested in voting process for this message, as well as predicted rewards for it. Users are given the opportunity to get up-to-date information about the voting process and predicted payments for the message on Golos application website.

The `golos.publication` smart contract implements logic to calculate all the necessary payments for the post and put the actual calculation results on the website. This logic is based on processing of actual data taken from messages that are received from Event Engine. The calculation results are modified after each received message from Event Engine in real time and depend mainly on such parameters as a size of the reward pool, a number of current publications, as well as «weight» of each voting user.

This section provides methods that can be applied in applications to determine the (predicted) amounts of rewards in real time mode, while is voting for a post. The list of determined rewards is as follows:

- total amount of rewards for the post;
- total amount of fees for curators;
- amount of fee for each of curators;
- total amount of rewards for beneficiaries;
- amount of reward for each of beneficiaries;
- amount of reward for author of the post.

The data used in the methods of calculating rewards for a post

Voting for a post occurs after its creation and before it is closed. Amount of reward for the created post mainly depends on results of voting for this post and amount of funds in a rewards pool. The rewards pool for the post is selected with taking the time this post was created.

Each vote is registered in the `golos.publication` smart contract as an event. Information about this event is sent to Event Engine. Then this information is sent to Golos application.

Each application can apply its own method of calculating the rewards for a post and display all results in real time mode. However, it is recommended that the following events received from Event Engine were applied in their methods. These are: `rewardweight`, `poststate`, `poolstate` and `votestate`.

---

rewardweight

The `rewardweight` event structure contains data about a paid share of reward for a post.

```
struct reward_weight_event {
    mssgid message_id;
    uint16_t rewardweight;
};
```

Parameters:

- `message_id` — identifier of the post.
- `rewardweight` — the paid share (in percent) of reward for the post taking into account the imposed fine.

The activity of each author is limited, that is, the author can publish only a certain number of posts within a specified time interval. If the author's activity does not exceed the limit, this author is not charged a penalty and the parameter `rewardweight` takes the value of 100 %. Otherwise, for each extra publication, the author will be dynamically charged a fine in the amount of from 0 to 100 %. Accordingly, the value of `rewardweight` for each extra publication will be reduced by the amount of the fine.

The `rewardweight` event is dispatched only in case of a penalty, at 100 % reward it is not dispatched.

---

poststate

The `poststate` event structure contains data about current state of a post.

```
struct post_event {
    mssgid message_id;
    base_t netshares;
    base_t voteshares;
    base_t sumcuratorsw;
    base_t sharesfn;
};
```

Parameters:

- `message_id` — identifier of the message.
- `netshares` — amount of funds, which will be taken from the rewards pool as a total fee for the post (this parameter is calculated as a sum of positive `voteshares` values of all votes).
- `voteshares` — a share of reward for the post is related to a separate vote. The separate vote value is calculated as multiplying a number of vesting by a weight of voting account. This parameter can take both positive and negative values. The sign «-» is assigned to «downvote» vote while the sign «+» is assigned to «upvote» one.
- `sumcuratorsw` — total weight of all curators votes at the current time.
- `sharesfn` — value calculated by the reward function (`mainfunc`) applied to the `netshares` message.

---

poolstate

The `poolstate` event structure contains data about current rewards pool.

```
struct pool_event {
    uint64_t created;
    counter_t msgs;
    eosio::asset funds;
    wide_t rshares;
    wide_t rsharesfn;
};
```

Parameters:

- `created` — the pool creation time (a pool selected by this time).

- `msgs` — a number of messages awaiting rewards from the pool `created`.
- `funds` — a number of tokens in the rewards pool.
- `rshares` — total `rshares` value of all posts in the pool.
- `rsharesfn` — total `sharesfn` value of all posts in the pool.

---

votestate

The `votestate` event structure contains data about current post voting results.

```
struct vote_event {
    name voter;
    mssgid message_id;
    int16_t weight;
    base_t curatorsw;
    base_t rshares;
};
```

Parameters:

- `voter` — account name (a user) who votes for the message.
- `message_id` — identifier of the message.
- `weight` — a vote weight (in percent) of the account `voter`.
- `curatorsw` — a «weight» of the curator.
- `rshares` — parameter used in calculating rewards for a post. The parameter is calculated as multiplication of voting account's vesting by a vote weight in percentages (`rshares = eff_vesting * weight / 10000`). This parameter can take both positive and negative values.The vote of «downvote» corresponds to a negative value.

Calculating a total reward for a post

To calculate the predicted amount of reward for a post in real time, it is recommended to use the formula

```
payout = reward_weight × funds × (sharesfn / rsharesfn)      (1)
```

Components of the formula:

- `payout` — resulting total amount of reward for the post at the time of receiving data from Event Engine.
- `reward_weight = rewardweight::reward_weight` — a weight of reward for a post.
- `funds = poolstate::state.funds` — total number of tokens in the reward pool.
- `sharesfn = poststate::sharesfn` — a share of tokens that allocated in the rewards pool to be used for reward for the post (this parameter depends on a weight of the post).
- `rsharesfn = poolstate::state::rsharesfn` — number of tokens that allocated in the rewards pool to be be spent on reward for all posts (parameter depends on total weight of all posts).

Component `reward_weight` shows the share of author's fee for the publication, taking into account possible penalty imposed on this publication. By default, the number of publications per day should not exceed four. For exceeding this amount the author is charged a penalty. By default, the `reward_weight` value is calculated using the formula

```
reward_weight = min{100 %, (400 %)² / postbw_charge²}      (1.1)
```

The `postbw_charge` component is the battery charge. Each use of the battery changes the charge by 100 %. The charge is restored completely, if within twenty-four hours from the previous posting no new post is created.

To calculate `payout` with using the formula (1), it is necessary to use data obtained from Event Engine only at the same time point.

Calculating the amount of fees to curators for a post

Total amount of fee to curators is a part of the funds as a percentage of the total reward for a post. An author can specify the percentage value `curators_prcnt` independently during the post creation. If the author does not specify this parameter, the `min_curators_prcnt` value is

taken from the `pstngparams` table as this parameter. This value means minimum possible share allocated to curators from total rewards for the post.

To determine the amount of fee to curators it is recommended to use the formula

```
curation_payout = curators_prcnt × payout        (2)
```

Components of the formula:

- `curation_payout` — resulting total amount of fee to curators for the post at the time of receiving data from Event Engine,
- `curators_prcnt` — share (in percent), deducted to curators, of the total reward for the post;
- `payout` — total amount of reward for the post, calculated by the formula (1).

Calculated by the formula (2) the total fee amount is distributed between all curators in accordance with the rules accepted in application. Below is recommended method of determining a fee amount for each of the curators.

To determine the $curator\_reward_j$ fee allocated to individual curator `j` it is recommended to use this formula

$$curator\_reward_j = curation\_payout \times (curatorsw_j / weights\_sum) \quad (3)$$

Components of the formula:

- `curation_payout` — total amount of fee to curators for the post, calculated by the formula (2).
- $curatorsw_j$ `= votestate::curatorsw`$_j$ — a weight of a positive vote `j` of the «upvote» type.
- `weights_sum = poststate::weights_sum` — total weight of all positive votes of the «upvote» type.
- $(curatorsw_j$ `/ weights_sum)` — a share allocated to `j-th` curator of the total fee of all curators.

The calculation of the fee amount via the formula (3) should be performed for each curator. Needed data about the curator are contained in the `vote_event` structure..

After calculating all the fees amount for all the curators may form a residual (unclaimed) amount. This amount may be formed due to fines during the voting (for example, for early voting). Such fine is not taken into account in the sum value of `sumcuratorsw`, but it does affect the value of `curatorsw` of a separate vote. The residual amount in the form of «surrender» will be returned back to the rewards pool for posts. This residual amount `unclaimed_rewards` is calculated by the formula

$$unclaimed\_rewards = curation\_payout - \sum(curator\_reward_j) \qquad (4)$$

Components of the formula:

- `curation_payout` — total amount of fee to curators for the post, calculated by the formula (2).
- $\sum(curator\_reward_j)$ — total fee amount of all curators, calculated by the formula (3).

Calculating the rewards to beneficiaries for a post

The part of funds allocated to author in the form of reward for a post is distributed between the beneficiaries and the author. The share of total reward allocated to beneficiaries, as well as the number of beneficiaries, are determined by the author at the time of posting. The rules according to which the reward is distributed among the beneficiaries are set and accepted for each application individually. Below is recommended method of determining the reward amount for each of the beneficiaries.

To determine the amount of reward to `j-th` beneficiary, it is recommended to apply the formula

$$ben\_reward_j = (payout - curation\_payout) \times weight_j \qquad (5)$$

Components of the formula:

- ben_reward_j — amount of reward to $j\text{-}th$ beneficiary.
- $payout$ — total amount of reward for the post, calculated by the formula (1).
- $curation\_payout$ — total amount of fee to curators for the post, calculated by the formula (2).
- $(payout - curation\_payout)$ — total amount of rewards allocated to beneficiaries and author.
- $weight_j$ — a weight of reward allocated to $j\text{-}th$ beneficiary. This percentage value is set by the author at the time of posting.

The award amount should be determined by the formula (5) for each beneficiary. Needed data about beneficiaries can be taken from the field `beneficiaries` of the table `message`. There is an array containing names and percentage deductions.

The total amount of payments `ben_payout_sum` allocated to beneficiaries is calculated by the formula

$$ben\_payout\_sum = \sum(ben\_reward_j) \qquad (6)$$

Calculating the reward to author for a post

To determine the amount of reward to author of the post it is recommended to use the formula

$$author\_reward = payout - curation\_payout - ben\_payout\_sum$$

Components of the formula:

- $author\_reward$ — amount of reward to author of the post.
- $payout$ — total amount of reward for the post, calculated by the formula (1).
- $curation\_payout$ — total amount of fee to curators for the post, calculated by the formula (2).
- ben_payout_sum — total amount of payments to beneficiaries, calculated by the formula (6).

Calculating total amount of rewards for a post in tokens and vestings

One part of reward for a post is paid in vesting, and another one is in tokens. Their percentage ratio is set at the time of posting. After calculating the amount of reward for a post by the formula (1), it is possible to determine the number of liquid tokens, as well as the vestings that make up this reward.

To calculate the number of liquid tokens `token_payout` in real time mode, it is recommended to use the formula

```
token_payout = payout × tokenprop
```

Components of the formula:

- `payout` — total amount of reward for the post, calculated by the formula (1).
- `tokenprop = post::tokenprop` — percentage of tokens in rewards.

Number of vesting in the `vesting_payout` reward is

```
vesting_payout = payout - token_payout
```

Conclusion

The presented methods for calculating rewards for a post in real time mode are intended for their using in the Golos application. Although these methods are recommendatory, they can also be applied in other applications.