Document number: N2
Date: 2019-04-12
Project: Nebulas Mainnet
Reply-to: Xuepeng Fan
<xuepeng.fan@nebulas.io>

Arthur Proposal: The Parameters and Functions for DIP and NR

NEBULAS

1 Introduction

This proposal includes all parameters and functions for DIP(Developer Incentive Protocol) and NR(Nebulas Rank). Although the Yellow paper[1] and the Mauve paper[2] described the algorithm and properties of NR and DIP, concrete functions and parameters are left blank.

More specifically, this proposal includes

- The period time of Nebulas Rank is to be 40320 blocks, which is 7 days.
- The period time of the forwading loop is to be 128 blocks.
- The Equation 17 in the Yellow paper is to be

$$\gamma(v) = (\frac{\mathcal{G}(v)}{\mathcal{G}(v)+1})^2$$

- The Equation 21 in the Yellow is to be

$$\mathcal{C}(v) = \frac{\beta(v)}{1+(100/\beta(v))^{0.5}} \cdot \gamma(v)$$

- The Equation 7 in the Mauve paper is to be

$$f(\mathcal{C}(a_i)) = \mathcal{C}(a_i)$$

- The e_{i0} in the Equation 6 in the Mauve paper is to be 0.
- The Equation 10 in the Mauve paper is to be

$$\lambda = \min\{\frac{0.008}{1-\frac{\Gamma_p}{\Gamma_s}}, 1\}.$$

2 Motivation and Scope

For the significance of NR and DIP, you may check the Yellow paper and the Mauve paper.

The parameters and functions described in this proposal may affect all NAS token holders, as they may have different ranking scores according to NR. Also, this proposal may affect all DApp developers on Nebulas, as they may get reward according to DIP.

Finally, this proposal also affect Nebulas mainnet developers, since they need to implement the detailed functions and update the parameters.

[1] https://nebulas.io/docs/NebulasYellowpaper.pdf
[2] https://nebulas.io/docs/NebulasMauvepaper.pdf

3 Impact on the Nebulas Protocol

This proposal largely affect the Nebulas Protocol. And the old Nebulas Protocol implementation can't be compatible with the proposal described here. Specifically, each Nebulas node should upgrade in order to verify the DIP reward transactions, which needs all details specified in this proposal.

4 Design Choices

There are several parameters are not specified in the Yellow paper and Mauve paper. Here we discuss how we choose the concrete value and the tradeoffs we make.

4.1 The Period of Time for the Core Nebulas Rank

In the Section 4 of Yellow paper, we described

> Core Nebulas Rank is used to measure the contributions of a user in reference to the entire economy over a **certain period of time**.

We hope the Core Nebulas Rank is stable if users are active enough. Thus, we find that 1 week is practical. If it's too long, computation cost is too large, while it's unstable if it's too short.

4.2 The period time of the forwarding loop

Actually, we didn't have this in the Yellow paper. However, we find that finding forwarding loops could be time-consuming. Finding loops in a transaction graph which contains millions of transactions could take hours. Thus we decide to reduce the problem size. And we only find forwarding loops in a transaction graph built from 128 blocks.

The side effect is that users may construct forwarding loop across this time period, instead of inside this time period. Still, users may do this to improve the NR, yet still with upper bond.

We may adjust this by introducing some randomness.

4.3 The equations in the Yellow paper

4.4 The equations in the Mauve paper

5 Technical Specifications

The NR and DIP can be updated via NBRE (Nebulas Blockchain Runtime Environment), since the algorithms are already included in `nbre/runtime` [3]. Here are present all details about the

[3] https://github.com/nebulasio/go-nebulas

submitted code.

5.1 NR source code

```
#include "runtime/nr/impl/nr_impl.h"

neb::rt::nr::nr_ret_type entry_point_nr(
  neb::compatible_uint64_t start_block,
  neb::compatible_uint64_t end_block) {
  auto to_version_t = [](uint32_t major_version,
                         uint16_t minor_version,
                         uint16_t patch_version)
                      -> neb::rt::nr::version_t {
    return (0ULL + major_version) +
           ((0ULL + minor_version) << 32) +
           ((0ULL + patch_version) << 48);
  };

  neb::compatible_uint64_t a = 100;
  neb::compatible_uint64_t b = 2;
  neb::compatible_uint64_t c = 6;
  neb::compatible_uint64_t d = -9;
  neb::rt::nr::nr_float_t theta = 1;
  neb::rt::nr::nr_float_t mu = 1;
  neb::rt::nr::nr_float_t lambda = 2;
  return neb::rt::nr::entry_point_nr_impl(start_block,
          end_block, to_version_t(0, 0, 1), a, b, c, d,
          theta, mu, lambda);
}
```

5.2 DIP source code

```
#include "runtime/dip/dip_impl.h"

extern neb::rt::nr::nr_ret_type
entry_point_nr(neb::compatible_uint64_t start_block,
               neb::compatible_uint64_t end_block);

neb::rt::dip::dip_ret_type entry_point_dip(
  neb::compatible_uint64_t height) {
  neb::rt::dip::dip_ret_type ret;
  std::get<0>(ret) = 0;

  uint64_t block_nums_of_a_day = 5;
  uint64_t days = 1;
```

```
neb::compatible_uint64_t dip_start_block = 15;
neb::compatible_uint64_t dip_block_interval =
  days * block_nums_of_a_day;
std::string dip_reward_addr =
    std::string("n1c6y4ctkMeZk624QWBTXuywmNpCWmJZiBq");
std::string coinbase_addr =
    std::string("n1HrPpwwH5gTA2d7QCkVjMw14YbN1NNNXHc");

auto to_version_t = [](uint32_t major_version,
                       uint16_t minor_version,
                       uint16_t patch_version)
                    -> neb::rt::dip::version_t {
  return (0ULL + major_version) +
         ((0ULL + minor_version) << 32) +
         ((0ULL + patch_version) << 48);
};

if (!height) {
  std::get<1>(ret) = neb::rt::dip::dip_param_list(
      dip_start_block, dip_block_interval,
      dip_reward_addr, coinbase_addr,
      to_version_t(0, 0, 1));
  std::cout << "dip␣param␣list␣returned" << std::endl;
  return ret;
}

if (height < dip_start_block + dip_block_interval) {
  std::get<1>(ret) = std::string("{\"err\":\"invalid␣height\"}");
  return ret;
}

uint64_t interval_nums = (height - dip_start_block) /
     dip_block_interval;
neb::compatible_uint64_t start_block =
    dip_start_block + dip_block_interval * interval_nums;
neb::compatible_uint64_t end_block = start_block - 1;
start_block -= dip_block_interval;

auto nr_ret = entry_point_nr(start_block, end_block);

neb::rt::dip::dip_float_t alpha = 8e-3;
neb::rt::dip::dip_float_t beta = 1;
return neb::rt::dip::entry_point_dip_impl(start_block,
      end_block, to_version_t(0, 0, 1), height,
      nr_ret, alpha, beta);
```

```
}
```

6 Acknowledgement

Thank Yulong Zeng for the design of DIP, Zaiyang Tang for the design of NR, and Chenmin Wang for the implementation.