NEBULAS

Mauve Paper: Developer Incentive Protocol

Nebulas Research

October 2018
Version:1.0.0

Contents

1 Introduction

Generally, developers develop applications on some application platforms (like Windows[1], Linux[2], macOS[3], iOS[4], Android[5] etc.) and benefit from their applications in traditional software development industry. The way to get the benefits varies for different developers, including but not limited to salaries paid by software enterprises, revenue by selling the application licenses or displaying advertisements in their applications.

However, the enterprises who build the application platforms also benefit from the applications, while the benefits are not shared with the developers. Let's take the operating system as an example here: a UI/UX designer wants to use Sketch, as we know that the application only works on macOS device, so besides paying for the application itself, the designer needs to pay Apple[6] for the device to use the application. Apparently, Apple benefits from such user while Apple does not share the benefits with the Sketch developers. Another similar example is that users have to pay Apple or Microsoft[7] to use AutoCAD[8]. In such cases, the key factor that users choose a platform is whether the platform supports required applications for users. In other words, high-quality applications are critical to the development of an application platform. Based on the above considerations, application platforms ignore the interests of developers, to a certain extent, infringing the interests of developers.

In the blockchain industry, the interests of DApp (Decentralized Application) developers are ignored by platforms as well. In 2014, Ethereum community proposed "Smart Contract", which extended blockchains' ability from peer-to-peer cryptocurrency networks to decentralized application platforms. However, in comparison with traditional centralized development industry, the ways of obtaining revenue for developers have no significant difference — decentralized application developers still can not benefit from the increment of the blockchain system's value.

Generally speaking, new-block rewards represent incremental value of the blockchain system and the distribution of such rewards determines the incentive direction of the

[1]https://www.microsoft.com/en-us/windows
[2]https://en.wikipedia.org/wiki/Linux
[3]https://en.wikipedia.org/wiki/MacOS
[4]https://en.wikipedia.org/wiki/IOS
[5]https://en.wikipedia.org/wiki/Android
[6]https://en.wikipedia.org/wiki/Apple_Inc.
[7]https://en.wikipedia.org/wiki/Microsoft
[8]https://en.wikipedia.org/wiki/AutoCAD

decentralized system. In our opinion, a blockchain system's incremental value essentially comes from the implicit values from users' data, which should be distributed to all contributed parties, including DApp developers. However, what we see in practice is that, in most PoW blockchain systems, represented by Bitcoin, new-block rewards are distributed to miner nodes; In PoS (proof of stake) based blockchain systems, the new-block rewards are assigned to stake holders. Along with it, the interests of DApp developers are somewhat infringed.

Conceptually, a DApp is a set of smart contracts with a series of specific functionalities, while a smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties[9].

From the technical architecture's point of view, most DApp usually uses smart contracts as the back-end, while using common front-end technologies and its interactions. DApps' forms can be either a traditional PC client, mobile applications or web applications.

We believe that the relationship among decentralized application platforms, DApp developers and DApp users is mutually reinforcing and symbiotic. Firstly, the emergence of decentralized application platforms enlarges the group of blockchain developers. More and more developers try to develop DApps that meet different requirements and benefit from the development of DApps. Secondly, DApp developers provide a rich variety of DApps, expanding the application scenarios of the blockchain, and bringing more incremental users to the blockchain. Finally, DApp users drive the continuous optimization and upgrade of decentralized application platforms, increasing the mobility of tokens on the decentralized application platform, making the whole blockchain system develop.

It should be noted that the developers described here only refer to developers on the decentralized application platform, not specifically the Nebulas developers, nor the developers of the blockchain system itself. Notice that we mean DApp developers instead of Nebulas DApp developers or blockchain system developers. Thus, we shall use developers short for DApp developers without ambiguity. Also, a DApp developer may be a stake holder. He may benefit from being a stake holder and that benefit is not considered as sharing the increase value of blockchains. The interests of being a developer can still be ignored or infringed.

It's inappropriate to directly distribute a platform's increased value to corresponding application developers. On one hand, the revenue is owned by centralized organizations, like an enterprise, and application developers have no chance to know

[9]https://en.wikipedia.org/wiki/Smart_contract

the details or participate in the sharing of revenues. Second, it is difficult to quantify each developer's contribution to the growth of application platforms so the fairness of the reward mechanism is hard to be guaranteed. Fortunately, this situation can be changed in blockchain industry since each invocation to smart contracts by each user is publicly recorded on blockchain. Thus, it is possible to *reward or incentive each DApp developer by quantifying each DApp's contribution*.

An ideal incentive mechanism should satisfy some basic properties:

- Fairness: the protocol should maintain objectivity when rewarding developers, that is, every DApps should be equally treated and their usages are evaluated veritably, even there are some potential manipulations.

- Effectiveness: the reward should reflect user preference, that is, the DApps with high reward are ones that frequented by active users while the DApps with low or no reward are unwelcome.

In this paper, we propose Developer Incentive Protocol (DIP), which aims at rewarding and incentivizing developers, enabling the developers to benefit from the development of the decentralized application platform. Naturally, an ideal developer incentive protocol does not exist since the users' evaluations to DApps are subjective and multi-dimensioned. So the DIP introduced in the paper still has space for improvement. However, the balance in this mauve paper is innovative, that is, under the premise of guaranteeing the interests of DApp developers, in terms of resistance to manipulation, we make the greatest efforts.

DIP is designed based on the existing Nebulas Rank (NR) [1] and it benefits from some good features of NR. Intuitively, DApp evaluation is reduced to a voting process in DIP.

An invocation from a particular user is treated as a vote and user's voting capacity is a function of his/her NR. The developers will get the rewards from the system eventually, according to the voting results.

Besides giving the theoretical model of Developer Incentive Protocol, we also analyze the properties against manipulations and illustrate the implementation of DIP, such as, how to adjust and update DIP, which specify the direction of the actual landing of the DIP.

Special Hint: As the mauve paper focus on discussing Developer Incentive Protocol, it greatly upgrades and expands the relevant chapters of the Nebulas Technology White Paper [2](version 1.02 released in April 2018). Compared with the conceptual demonstration one year ago, after a year of in-depth thinking and practical verification, we are confident and able to design more rigorous algorithms and provide clear solutions or directions for more practical details of the Nebulas Incentive Protocol.

2 Background

The DIP given by this mauve paper referred lots of related works and also extended our previous results. Here we introduce the related works, which play a significant role for the reference and guidance of this mauve paper.

2.1 DApp's Developer Incentive

As far as we know, currently, no decentralized platform on blockchain offers a long-term effective incentive mechanism for DApp developers. As a representation of blockchain 2.0, Ethereum makes a breakthrough to involve Turing-complete smart contracts. A number of DApps emerge on Ethereum, including game, gambling, crowd sourcing, credit and many other types. In particular, the CryptoKitties in the late 2017 and Fomo3D in 2018 attract most attentions, which once cause network congestion.

Actually, like the two famous DApps, most DApps gain utilities only by charging fees to users, unable to benefit from the increase of Ethereum's value or the new-block rewards.

With the lack of the incentive of developer, the application scenarios of DApps has also been affected to a certain extent. For example, implicitly, free DApps may be absence due to the difficulty for getting a return. As a result, the quantity, quality and diversity of DApps are affected. In contrast, a fair and effective mechanism for incentivizing developers brings developers' concentration to the development of DApps, which further promotes the prosperity and sustainable development of the whole blockchain ecosystem.

To a certain extent, many emerging blockchain systems realize the necessity of incentive mechanism for building blockchain ecosystems. For example, in Nebulas Incentive Program, more than 6781 DApps have been generated and a large number of excellent development teams can go to the front desk and obtain high investments.

Along with it, other public blockchains also launched short-term incentive programs based on centralized management. Such incentive programs mainly aim to publicize to the community with official evaluations taking a major role, without long-term sustainability.

2.2 Nebulas Rank

Nebulas Rank (NR) [1] quantifies each account's contribution to the total economic output and has nice properties against manipulations. In particular, Nebulas Rank introduces the Wilbur function, which has the following properties:

Property 1. *For any two positive variables x_1,x_2, the sum of their functions is less than the function of their sum.*

$$f(x_1 + x_2) > f(x_1) + f(x_2) \quad x_1 > 0, x_2 > 0 \tag{1}$$

Property 2. *For any two positive variables x_1,x_2, when they tend to infinity, the sum of their functions tends to the function of their sum.*

$$\lim_{x_1 \to \infty, x_2 \to \infty} f(x_1 + x_2) = f(x_1) + f(x_2) \quad x_1 > 0, x_2 > 0 \tag{2}$$

As the basis of NR, the two properties also offer nice properties for DIP against manipulations.

2.3 Voting Mechanism

As mentioned earlier, in DIP, the process that users use DApps can be regarded as a process that voters vote for DApps. The latter's incentive mechanism is similar to ranking algorithms.

Regarding the voting mechanism and ranking algorithm, there are lots of related work in various fields, which we have referred and show examples as follows.

One of the most famous results is the Arrow's Theorem, which shows that no ranking algorithm can simultaneously satisfy Pareto Efficiency, i.e. the ranking results satisfies the majority's interest, non-dictatorship and independent of irrelevant alternatives, i.e. the relative ranking of two candidates is not affected by any third candidate.

The result implies that no ranking algorithm can cover everything. So the DIP in this mauve paper will focus on the important and well-known attributes.

In the real life, there are a lot of scenarios requiring ranking algorithms. A typical example is the buyers' impressions to sellers on Amazon and Taobao. Sellers with higher reputation will be given better display slots, and thus obtain more attentions and higher click-through rates (CTRs). In particular, there are similar problems on such e-commerce platforms, like Sybil attack, i.e., creating fake transactions or buy over buyers to obtain 5-star evaluations.

For now, these centralized platforms mostly rely on machine learning to distinguish normal and fake users [3, 4, 5]. However, practical results show that such methods are not ideal. [6] points out that even artificial identifications can not effectively distinguish such accounts. [7] gives an algorithm that eliminates the incentive of such manipulations, based on mechanism design. Although its model is different form us, it can be used as a significant reference.

[8] introduces the ranking algorithm for postings in a social network community, which combines the users' votes and time declining. [9] introduces the ranking algorithm for postings in Reddit, which involves the situation where users can vote in the negative. [10] introduces Reddit's ranking algorithm for the comments, taking the confidence interval into account. IMDB [11] introduces the idea of Bayesian Model Averaging to film rankings, which can narrow the gap among different films due to the number of voters.

Because of properties against manipulation in NR, the DIP proposed in this mauve paper can distinguish normal users and fake users more clearly. Therefore, the emphasis of this mauve paper is to transfer users' NR values to DApps' ranking scores through interactive behaviors.

3 Developer Incentive Model

The Developer Incentive Protocol, DIP, including two processes: ranking the DApps and distributing the rewards.

Firstly, constructing a good ranking system can provide third-party developers with a convenient and effective platform to promote applications and provide users with a reliable recommendation system. Like the App Store platform, good Apps have top positions on the ranking list and thus receive more attention from users.

Secondly, users have better experience when they obtain high-quality Apps directly from the ranking list. Furthermore, Apps' ranks can be used in keyword search. Just like the search systems in search engines and e-commerce platform, listing keyword-related Apps in the search results according to their ranks is satisfactory to users.

On the other hand, as we shown in Section 2, the purpose of DIP is to provide rewards for good DApps' developers, thus further increases developers' incentives to design good DApps and promotes the ecosystem development. So the second process of DIP is to design a fair rewarding mechanism, according to DApps' ranks.

3.1 Model Representation

In this section we introduce necessary notations and symbols in DIP model.

- $\mathcal{A} = \{a_1, a_2, \ldots, a_m\}$ represents the set of users that participate the ranking system during a time period. We use voters to denote such users. Note that a user is defined to be a voter only if he invokes some DApp's EOA(External Owned Account) in a time period. Define

  $$\mathcal{A}^* = \{a_1, a_2, \ldots, a_m, a_{m+1}, \ldots, a_{m^*}\}$$

  to be the set of all users in the community during the same time period, i.e., $m^* - m$ users do not invoke any DApp.

- $\mathcal{D} = \{d_1, \ldots, d_n\}$ represents the set of DApps during a time period.

- $e_{ij}, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n$ represent the times that voter a_i invokes DApp d_j. Due to the publicity and decentralization of blockchain, the DIP model is different from the ranking system in traditional centralized application markets. Generally speaking, DIP ranks the DApps according to users' invocation behaviors, in a decentralized environment. The details are shown in the next section.

- $\Gamma_i, i = 1, 2, \ldots, m$ represent voter a_i's voting capacity during a time period. [1] proves that a user's NR value is an effective measure of his account's value. So in DIP, NR is also used as a significant criterion to decide users' voting capacities.

- $\Gamma_{ij}, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n$ represent the contributory value of voter a_i to DApp d_j, which can be regarded as the number of votes where a_i is willing to vote for d_j.

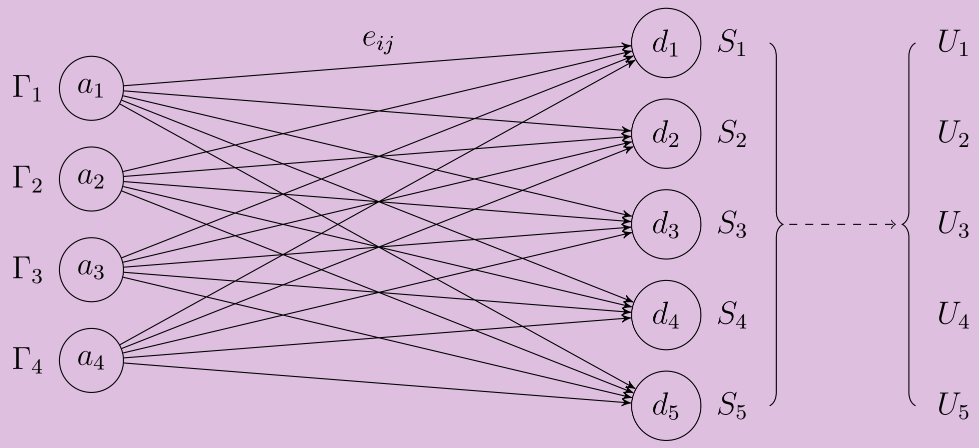

Figure 1: Interactions between voters and DApps

- $S_j, j = 1, 2, \ldots, n$ represent DApp d_j's ranking score, which is determined by its total contributory values from all voters. Intuitively, the ranking scores determine DApps' positions on the ranking list.

- M represents the total amount of the reward pool for developers, comes from the new-block rewards. The actual reward would be properly reduced according to the participation rate of the whole community during the time period.

- $U_j, j = 1, 2, \ldots, n$ represent the final reward of DApp d_j, which is determine by actual total reward and all DApps' ranking scores.

To sum up, the interactions between voters and DApps can be represented by the bipartite graph in Figure 1.

3.2 Voting Action

In a centralized App Store[10], the system can record specific information such as times that a App has been downloaded, which is a key factor for determining its rank. However, in situations of Blockchain, the way that users use DApps is to invoke smart contracts' addresses, e.g., represented as the times that user a_i invokes DApp d_i, denoted by e_{ij}. Compared to traditional information about downloads, DIP, taking invocation information as the date sources, has the following advantages:

- The number of invocations are recored on the blockchain, which is hard to be tempered and is more open and transparent, compare to centralized methods such as recording the downloads.

[10]`https://en.wikipedia.org/wiki/App_store`

- The number of invocations is more fine-grained compare to the downloads data, since downloads data only record users' one-shot behaviors. But a good DApp should have user stickiness. Therefore the number of invocations are more reasonable to reflect users' real behaviors.

Actually, there are other available information when users invoke DApps. For example, the amount of gases that has been expended and possible token transfers involved in an invocation. However, DIP takes neither of them into consideration.

Firstly, the amount of expended gases depends on the executed instructions within the smart contracts each time a user invokes a DApp, where the later is not related to the DApp's own quality. Moreover, in the current Nebulas system, the average amount of expended gases during each invocation is at the order of 10^{-8} NAS, which is negligible.

The reason why we do not consider token transfers is the lack of an effective method against manipulation. Intuitively, the wilinesses of users to pay additional tokens when invoking the DApp improve the DApp's recognition. However, in practice, when a user pays tokens to a DApp, the tokens' final destination could belong to the following three cases:

1. The tokens finally belongs to the developer of the DApp. In this case, it is believed that the user voluntarily pays to the DApp. Since the DApp's developer has benefited from the user, it is not that meaningful to further increase his rank and reward.

2. The DApp itself requires token transfer, such as gambling DApps, which leads a huge amount of token transfer between the user and the DApp. It's a normal phenomenon. However, the DApp's rank should not be increased accordingly, due to the fact that the user's purpose for paying tokens is to gather profits, which does not reflect the DApp's quality.

3. The DApp's developer commits that all tokens paid to the DApp will be returned to the user. It is actually a kind of manipulation, which would be aggravated if we increases the DApp's rank and reward accordingly.

In practice, without analyzing the source code of smart contracts, we are not able to determine which case token transfers between users and DApps belongs to and in either case we have explained reasons for not involving DApps' ranks. So the algorithm in DIP will be independent of token transfers.

In the DIP model, a user $a_i \in \mathcal{A}$ is essentially an account address. As referred in [1], a user can actually control multiple account addresses. Since there is no cost for creating new accounts, the user can forge several addresses which he controls to vote — the Sybil attack. Similarly, a developer can divide his DApp into several addresses, that is, split his DApp into several low-quality DApps, and obtain all rewards from split DApps. In the meanwhile, a developer can buy over one or more users to vote for his DApp.

We analyzed all manipulations above when designing DIP and gave corresponding solutions. The details about DIP against manipulation are in Section 5.

3.3 Sample Time Interval

In Section 3.1, we illustrated that the NR is a significant criterion to determine voters' voting capacities. However, according to the definition in [1], the sampling period of DIP data is much longer than the sampling period of NR data, which means that during the process for recording invocation behaviors, users' NRs may vary, even fluctuate wildly.

A naive solution is to synchronize the sampling period of NR data and DIP data. However in practice, in a short time period (say, one day), invocation times are very small for most users. So it is of small significance to rank the DApps when users behaviors are sparse, and the properties discretized in Section 5 are not guaranteed to be satisfied.

So our strategy is to properly extend the sampling period to gather enough invocation behaviors and bound the variation of users' NRs simultaneously. The variation of an address's NR is shown in Figure 2. Here we divide a whole ranking process of DIP into several time periods. According to the data about the variations of NRs, pick integer t such that the variance of NR within t days less than a threshold τ holds for most users. We take t days as a sampling period, by gathering the average NR values of voters and invocation data during the time period, to compute DApp's ranking score and developers' final reward. Then we take the average data over all time periods during a ranking process as the final results.

4 Developer Incentive Protocol

Based on the model in previous section, we introduce the Developer Incentive Protocol in this section. DIP includes two steps: ranking the DApps and rewarding the developers. Specifically, from voters' invocation behaviors to developers' final reward,

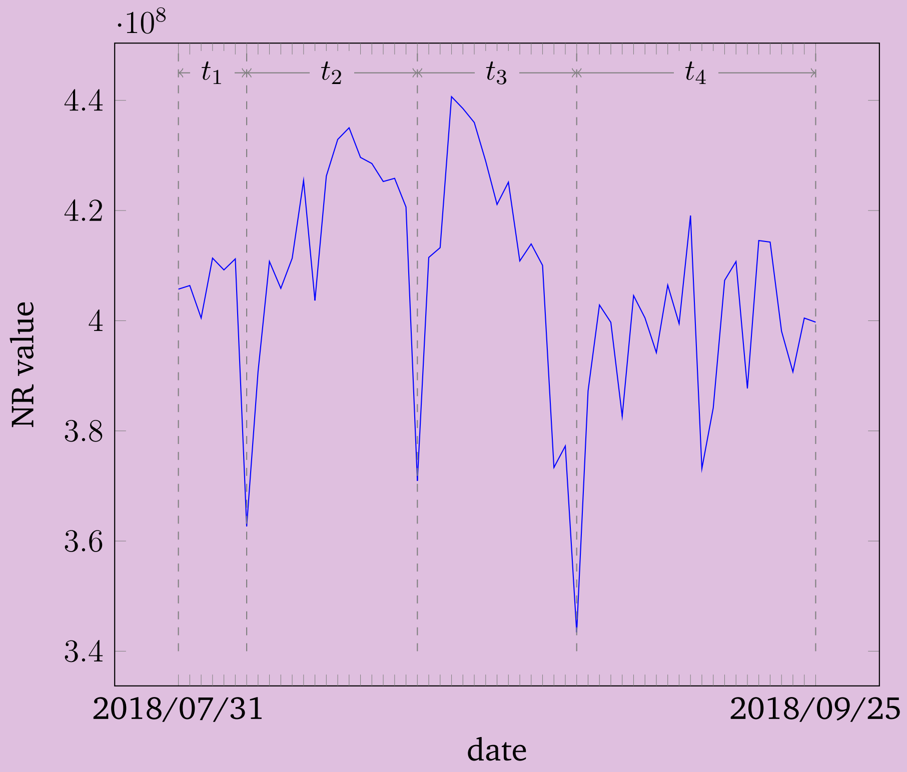

Figure 2: NR variation graph of an address on Nebulas mainnet. The mainnet address is `n1Ugq21nif8BQ8uw81SwXHK6DHqeTEmPRhj`.

it includes four transformations: invocation times$\xrightarrow{1}$ voting capacity$\xrightarrow{2}$ contributory value$\xrightarrow{3}$ ranking score$\xrightarrow{4}$ final reward.

4.1 Voting Capacity and Contributory Value

For any voter a_i, we use Γ_i to denote his voting capacity, which can be regarded as the total number of votes voter a_i has. [1] proves that voter's NR value is an effective measurement of his account's value. So in DIP, the NR is also used as a significant criterion to decide voters' voting capacities. For voter a_i, his voting capacity can be represented as a function of his NR value:

$$\Gamma_i = f(\mathcal{C}(a_i)) \tag{3}$$

where $\mathcal{C}(a_i)$ represents voter a_i's NR value.

Normally, we wish f to be an increasing function, i.e., voters with higher NR values have larger voting capacities. Here we give a satisfactory function:

$$f(\mathcal{C}(a_i)) = \mathcal{C}^2(a_i). \tag{4}$$

In other words,

$$\Gamma_i = \mathcal{C}^2(a_i). \tag{5}$$

It is analyzed that this function has nice properties, e.g., against Sybil attack. See Section 5.

Then we discuss the mechanism for distributing voting capacities. According to Section 5, Γ_{ij} represents the contributory value of voter a_i to DApp d_j. We define it as follows:

$$\Gamma_{ij} = \frac{e_{ij}}{e_{i0} + \sum_{j=1}^{n} e_{ij}} \Gamma_i. \tag{6}$$

Formula 6 can be understood as the ratio of the invocation times to d_j to the total invocation times. Here e_{i0} represents the invocation times which do not belong to any DApp. A voter can arbitrarily adjust the values of e_{i0} and e_{ij}'s.

With the introduction of e_{i0}, it holds that

$$\sum_{j=1}^{n} \Gamma_{ij} \leq \Gamma_i.$$

It is notable that formula 6's purpose is to let voters arbitrarily distribute their NRs for voting (by arbitrarily choose contributory values). In practice, it is possible that some DApps forcibly increase the number of invocations (by only receiving doubled invocations), but since all NRs and the number of invocations are public, a voter can still achieve the distribution of contributory values he want by adjusting his invocation times.

The reason for introducing e_{i0} is that, for remaining voters' individual rationals (IR), i.e., voters' interests will not be damaged, we do not force voters to cast all their votes. Voters can selectively exercise part of his voting rights or totally abstain, by properly increasing the value of e_{i0}.[11]

[11] e_{i0} can be implemented by setting an empty smart contract officially, which does not contain any actual effectiveness. Voters can invoke the empty smart contract any times.

4.2 Ranking Score

Given all voters' contributory values to DApps, we are able to compute the ranking scores of DApps: Given all contributory values $\Gamma_{ij}, i = 1, 2, \ldots, m, j = 1, 2, \ldots, n$, we define Dapp d_j's ranking score to be a multivariate function over contributory values from all voters:

$$S_j = g(\Gamma_{1j}, \Gamma_{2j}, \ldots, \Gamma_{mj}) \tag{7}$$

Similarly, here we give a satisfactory function:

$$g(\Gamma_{1j}, \Gamma_{2j}, \ldots, \Gamma_{mj}) = \sum_{i=1}^{m} \sqrt{\Gamma_{ij}} \tag{8}$$

That is, a DApp's ranking store equals to the sum of square roots of its contributory values from all voters. It is not hard to see that, for a user a_i, when he only votes for one DApp (assume that he does not discard any vote), his total contributory value is $\sqrt{\Gamma_i}$. When he casts his votes to several different DApps, his total contributory value increases due to the property $\sqrt{a+b} < \sqrt{a} + \sqrt{b}$ of the square root function. In other words, the voter gets in touch with more DApps, which is encouraged by our system. In Section 5, we will prove detailed properties for our construction. Similar method are referred in [12].

Given ranking scores $S_j, j = 1, 2, \ldots, m$, the ranks of DApps are determined accordingly. For example, in Nebulas nano client[12], high ranked DApp are listed on prominent positions and will attract more attentions.

4.3 Final Reward

DIP offers voters a reliable ranking list of DApps.[13] For developers, we need to distribute the rewards to them according to their ranking scores.

Given all DApps' ranking scores, define the reward of DApp d_j's developer to be:

$$U_j = \frac{S_j^2}{\sum_{k=1}^{n} S_j^2} \cdot \lambda M \tag{9}$$

where M is the total amount of the reward pool for developers, comes from the

[12]`https://nano.nebulas.io/`

[13]We assume that voters only care about the ranks of DApps he like, while the developers' rewards are not concerned by voters.

new-block rewards. λ is define to be the participation factor, that is, we want the total amount of rewards increases with the total NR values of voters. The specific definition is as follows:

$$\lambda = \min\{\frac{\Gamma_p}{\alpha\Gamma_s} \cdot \min\{\frac{\beta\Gamma_p^2}{\sigma^2(\Gamma_p)}, 1\}, 1\}, \tag{10}$$

where

$$\Gamma_p = \sum_{i=1}^{m}(\Gamma_i - \Gamma_{i0}), \quad \Gamma_{i0} = \frac{e_{i0}\Gamma_i}{e_{i0} + \sum_{j=1}^{n} e_{ij}},$$

i.e., the total effective voting capacities of all voters,

$$\Gamma_s = \sum_{i=1}^{m^*}\Gamma_i,$$

the total voting capacity (the square of NR value) of all users in the community, σ is the standard deviation (square root of variance) of all voters' effective voting capacities:

$$\sigma^2(\Gamma_p) = \sum_{i=1}^{m}(\Gamma_i - \frac{1}{m}\Gamma_p)^2$$

of which its maximum value is $\frac{(m-1)^2}{m^2}\Gamma_p^2$, and $\alpha, \beta < 1$ are adjustable parameters.

The purpose of introducing the participation factor λ is to expect the total voting capacities of voters more than a threshold (the total voting capacities of community users times α) and to bound the variance of voters voting capacities, preventing the case with several high-NR voters along with lots of low-NR fake accounts. The two terms can complement each other, i.e., when the participation rate is high, we can ignore the effect of the variance.

5 Property Analysis

Have introduced the Developer Incentive Protocol, in this section, we analyze manipulations that could occur in practice and the properties against manipulation of DIP. From the voters and developers' points of view respectively, manipulations including buying over, maliciously splitting DApp, Sybil attack and so on.

5.1 Buy Over Voters

The so-called buy over means that a developer lets voters' cast all their votes to the developer's DApp, by means of bribing or other, which is very common in the real life. Here we suppose that all voters are self-interest. We assume that normal voters only care about the ranks of the DApps they like, rather than the final rewards of developers. In other words, a normal voter wants to maximize the total weighted ranking scores of all DApps he likes. Our quadratic ranking algorithm guarantees the following property:

Property 3. *In the DIP model, for a self-interested normal voter, generally, he will case his votes to multiple DApps.*

We illustrate the property by the following model: suppose the weights that voter a_i values all DApps are $b_{i1}, b_{i2}, \ldots, b_{in}$ respectively (can be regarded as the true preference of the voter to all DApps). Taking the form of 8, the voter's contributory values satisfy

$$\frac{b_{i1}}{\sqrt{\Gamma_{i1}}} = \frac{b_{i2}}{\sqrt{\Gamma_{i2}}} = \cdots = \frac{b_{in}}{\sqrt{\Gamma_{in}}}.$$

In other words, voter a_i's contributory values matches his true preference to DApps. The detailed proof is in Section A.1.

Traditional voting models usually compute the ranking score linearly, i.e.,

$$g(\Gamma_{1j}, \Gamma_{2j}, \ldots, \Gamma_{mj}) = \sum_{i=1}^{m} \Gamma_{ij}.$$

In this model, a rational voter only cast his votes to the DApp he likes the most. In comparison, formula 8 can promote the interactions between voters and DApps, due to the property of the square root function. In other words, voters voting for multiple DApps maximizes the utilization of his voting capacity. Similar analysis can be found in [12]. To sum up, a voter would vote for multiple DApps and keep the priority of DApps he likes the most simultaneously, i.e.the ratio equation above.

In practice, sometimes traditional linear voting model will limit the maximum votes for a voter to a DApp, to forcibly let voters disperse their votes, while our algorithm achieve the same goal by the means of essentially incentive, with a more elegant and simple mathematical expression.

Corollary 1. *The total contributory values of a voter who is bought over is far less than the total contributory values of a normal voter.*

For a voter a_i who is bought over, he can at most offer the DApp with contributory value with amount $\sqrt{\Gamma_i}$. For a normal voter who is not bought over, assume that he plans to vote for K DApps,[14] when his value weights to these DApps are uniformly distributed, the total increment of ranking scores over all DApps caused by the voter is at the over of $O(\sqrt{K\Gamma_i})$, that is, the efficiency of a normal voter is K times the efficiency of a voter who is bought over. Therefore the cost of the manipulation about buying over is increased.

5.2 Malicious Splitting

For developers, another manipulation is to maliciously split their DApps, in order to obtain higher total rewards of all split DApps. Intuitively, splitting can increase the number of DApps that participant the reward mechanism and thus increase the total final rewards. However our model guarantees that it is not the case. Here we assume that all developers concern their final rewards (total bonus) as well as the potential utility caused by the improvement of ranks.

Specifically, as the algorithm of final rewards, the convexity of formula 9 guarantees the following property

Property 4. *If all voters are normal, splitting DApps will not increase the reward of the developer.*

It is assumed that a normal voter belongs to the following cases: i). voter simply distributes his votes that are supposed for the original DApp to split DApps. Such case occurs when an application has different smart contract address for invocation. ii). Suppose that the voter values the original DApp with weight a, and with weights b and c for two split DApps respectively, then $c > a + b$. Such case can be illustrated that after splitting, the split DApps' qualities are greatly decreased due to the lack of linkages. So the total qualities of split DApps' is lower than the original DApp.

In both cases, the final reward of the developer does not increase. Detailed proof is in Section A.2.

Furthermore, a developer can buyer over voter and split his DApp simultaneously: he first splits his DApp to K DApps. Then, he lets his bought-over voters distribute their votes uniformly to split DApps, thus maximizing the utilization of bought-over voters. We have the following corollary against this case:

[14] K reflects the number of DApps of which the contributory values from the voter can discriminate with other DApps, which is usually larger than 1, as long as the voter's value weights of the K DApps are not extremely distributed, i.e., the voter only votes for a particular DApp and the votes for other DApps tent to 0

Corollary 2. *Even with the introducing of bought-over voters, the developer can not increase his final rewards by splitting his DApps.*

Detailed proof is in Section A.3.

It is notable that, the rank of DApps will be decreased if developers split DApps, thus the potential utility is also decreased. In summary, our algorithm essentially prevents malicious splitting.

Without doubt that for a developer that develops multiple different DApps, since there is no mirrored of split relations among the DApps, the utility of the developer is not affected.

5.3 Sybil Attack

By generalized Sybil Attack we mean an attack subverts the reputation system by creating a large number of pseudonymous identities, using them to gain a disproportionately large influence [13]. In Nebulas Yellow Paper [1], the properties of NR against manipulations that increase NR by creating a large number of new accounts have been proved. So in the DIP ranking algorithm, voters are also not able to increase NR by creating new accounts, i.e.,

$$\mathcal{C}(c) > \mathcal{C}(a) + \mathcal{C}(b)$$

where c is the original account, a, b are the split sub-accounts. According to formula 6 their voting capacities satisfy the following constraint:

$$\sqrt{\Gamma_{a+b}} > \sqrt{\Gamma_a} + \sqrt{\Gamma_b} \tag{11}$$

Suppose that the propose of a voter to execute Sybil attack is to increase the ranking score of a specific and the final rewards of its developer. According to the constraint above we have the following property:

Property 5. *For any voter, executing Sybil attack will not increase the ranking score of the DApp he votes and the final reward of the DApp's developer.*

So the property against Sybil attack is guaranteed.

6 Implementation of DIP

The complete implementation of DIP is out of scope of this paper. So here we only discuss the key issues should be handled when implementing DIP.

6.1 How to Distribute Rewards

For distributing rewards, a special account D should be established. In the meanwhile, a part of the new-block rewards is transferred to D according to a fixed ratio.

Developers' deserved rewards will be sent regularly.[15]. In order to send rewards on the blockchain, the private key of the account that sends rewards is required for signing the transaction of sending. Therefore, for the purpose of security, the account D for sending rewards needs a special treatment.

Firstly, a special kind of transaction is added to the system, denoted by `dip` transaction, which contains the information about the amount of deserved reward of a developer and the height of the blockchain. Secondly, the system refuses all transactions other than `dip` initiated by D, to ensure that no account can extract tokens from D. Finally, verification nodes on the blockchain need to verify the `dip` transactions. Particularly, verification nodes need to run DIP locally and verify whether the data in `dip` transactions coincident with local results.

Through the methods mentioned above, not only the rewards for developers are distributed normally, but also the security of account D for sending rewards is ensured.

6.2 Updating of DIP

As we know, the DIP is closely related to the whole ecosystem. As the variance of the ecosystem, DIP ought to be updated, particularly, the parameters in the DIP model. So how to effectively update the DIP turns out to be a key issue.

For this, we use Nebulas Force to iteratively update DIP.

We update the blocks' structures to contain algorithms and parameters in DIP (by the form of LLVM IR). Nebulas Virtual Machine (NVM), as the engine of the algorithm, gets algorithm and parameters from blocks and run algorithms, to obtain the amount of tokens that an account deserves.

When algorithms and parameters need to be updated, Nebulas group will work together with the community, letting new blocks contain updated algorithms and

[15]The time interval of sending rewards equals to the sampling interval in Section 3.3

parameters, to ensure the timeliness and smoothness of the whole updating process and avoid any possible forks.

7 Future Work

7.1 Muli-dimension Voting

In Section 3.2, we take the number of invocations as the criterion to determine DApps' ranking scores and show reasons why token transfers involved in invocations are not taken into consideration. In the future, we may introduce token transfers as another criterion for ranking DApps when we can distinguish the causes of token transfer in more details by analyzing the invocations to smart contract.

7.2 Invocations Among DApps

Currently, DApps' ranks are determined by users' voting capacities (NR values), which come from invocations of DApps. However, more complicated invocation behaviors, such as invocations among DApps, can further transmit users' voting capacities. So in our future work, we may compute each DApp's final ranking score by giving each DApp an initial ranking score and running the Page Rank algorithm [14].

References

[1] "Nebulas yellowpaper." `https://nebulas.io/docs/NebulasYellowpaperZh.pdf`.

[2] "Nebulas whitepaper." `https://nebulas.io/docs/NebulasTechnicalWhitepaperZh.pdf`.

[3] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh, "Spotting opinion spammers using behavioral footprints," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 632–640, ACM, 2013.

[4] N. Jindal and B. Liu, "Opinion spam and analysis," in *Proceedings of the 2008 international conference on web search and data mining*, pp. 219–230, ACM, 2008.

[5] K. Yoo and U. Gretzel, "Comparison of deceptive and truthful travel reviews," *Information and communication technologies in tourism 2009*, pp. 37–47, 2009.

[6] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock, "Finding deceptive opinion spam by any stretch of the imagination," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pp. 309–319, Association for Computational Linguistics, 2011.

[7] Q. Cai, A. Filos-Ratsikas, C. Liu, and P. Tang, "Mechanism design for personalized recommender systems," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 159–166, ACM, 2016.

[8] A. Salihefendic, "How hacker news ranking algorithm works," 2010.

[9] A. Salihefendic, "How reddit ranking algorithms work," *Hacking and Gonzo*, vol. 23, 2010.

[10] E. Miller, "How not to sort by average rating," 2009.

[11] "IMDB." `https://www.imdb.com/chart/top`.

[12] V. Buterin, Z. Hitzig, and E. G. Weyl, "Liberal radicalism: Formal rules for a society neutral among communities," *arXiv preprint arXiv:1809.06421*, 2018.

[13] D. Quercia and S. Hailes, "Sybil attacks against mobile users: friends and foes to the rescue," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–5, IEEE, 2010.

[14] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

Appendix A Proof

A.1 Proof of Property 3

Proof. Without loss of generality, we assume that the weights that voter a_i values all DApps are $b_{i1}, b_{i2}, \ldots, b_{in}$ respectively, which are fixed. We assume that voter a_i's contributory values to all DApps are $\Gamma_{i1}, \ldots, \Gamma_{in}$ respectively, which are adjustable by voter a_i.

The optimization objective of voter i is the weighted sum of ranking scores that he offers, defined by

$$w_i = \sum_{j=1}^{n} b_{ij}\sqrt{\Gamma_{ij}}$$

According to Cauthy's inequality it holds that

$$w_i = \sum_{j=1}^{n} b_{ij}\sqrt{\Gamma_{ij}} \leq (\sum_{j=1}^{n} b_{ij}^2)(\sum_{j=1}^{n} \Gamma_{ij}) \leq (\sum_{j=1}^{n} b_{ij}^2)\Gamma_i$$

The most right side of the formula above is a fixed value. The equality holds if and only if

$$\frac{b_{i1}^2}{\Gamma_{i1}} = \frac{b_{i2}^2}{\Gamma_{i2}} = \cdots = \frac{b_{in}^2}{\Gamma_{in}}$$

So the property is proved. □

A.2 Proof of Property 4

Proof. Without loss of generality, we assume that d_1's developer splits d_1 into two DApps. For any normal voter who belongs to the second case in section 5.2, that is, assume the weights that he values all DApps before splitting is $b_{i1}, b_{i2}, \ldots, b_{in}$ and the weights that he values the two split DApps are b'_{i1}, b'_{i2}, it holds that $b_{i1} \geq b'_{i1} + b'_{i2}$ according to our assumption.

Then we compute the contributory values of voter a_i before splitting. Define $H_i = \sum_{j=2}^{n} b_{ij}^2$, according to the conclusion in Property 3 and Partition ratio theorem we have

$$\frac{\Gamma_{i1}}{b_{i1}^2} = \frac{\sum_{j=1}^{n} \Gamma_{ij}}{\sum_{j=1}^{n} b_{ij}^2} = \frac{\Gamma_i}{b_{i1}^2 + H_i}$$

Similarly, the contributory value of voter a_i to the t-th split DApp (denote by $\Gamma'_{it}, t = 1, 2$) is

$$\Gamma'_{it} = \frac{b'^2_{it}\Gamma_i}{b'^2_{i1} + b'^2_{i2} + H_i}$$

Note that $b_{i1}^2 \geq (b_{i1} + b_{i2})^2 > b_{i1}'^2 + b_{i2}'^2$, we have

$$\Gamma_{i1} > \Gamma'_{i1} + \Gamma'_{i2}$$

So we give the constraint of contributory values for a rational enough voter. Generally, most voters belongs to the first case in Section 5.2, that is, they simply distribute their contributory values that are supposed for d_1 to split DApps. In either case, we have

$$\Gamma_{i1} \geq \Gamma'_{i1} + \Gamma'_{i2}$$

Define S'_1, S'_2 to be the two split DApps' ranking scores respectively. By definition

$$S'_1 = \sum_{i=1}^{m} \sqrt{\Gamma'_{i1}}, \quad S'_2 = \sum_{i=1}^{m} \sqrt{\Gamma'_{i2}}, \quad S_1 = \sum_{i=1}^{m} \sqrt{\Gamma_{i1}}$$

Define U'_1 to be the final reward of d_1's developer after splitting. By definition

$$U'_1 = \frac{S_1'^2 + S_2'^2}{S_1'^2 + S_2'^2 + \sum_{j=2}^{n} S_j^2} \lambda M, \quad U_1 = \frac{S_1^2}{S_1^2 + \sum_{j=2}^{n} S_j^2} \lambda M$$

Note that given $S_2, \ldots, S_n$,

$$U_1 \geq U'_1 \Leftrightarrow S_1^2 \geq S_1'^2 + S_2'^2$$

In order to show whether splitting increases the utility, we only need to compare the following two terms

$$S_1^2 = (\sum_{i=1}^{m} \sqrt{\Gamma_{i1}})^2, \quad S_1'^2 + S_2'^2 = (\sum_{i=1}^{m} \sqrt{\Gamma'_{i1}})^2 + (\sum_{i=1}^{m} \sqrt{\Gamma'_{i2}})^2$$

Actually, $S_1^2 \geq S_1'^2 + S_2'^2$ can be proved according to the shortest distance theorem. As shown in Figure 3, we construct a grid whose length and width are divided into m segments, the i-th segment of which has length $\sqrt{\Gamma'_{i1}}$ and $\sqrt{\Gamma'_{i2}}$ respectively.

Then, $S_1'^2 + S_2'^2 = A_0 A_m^2$, that is, equals to the square of the blue segment's length. In the meanwhile,

$$S_1^2 = (\sum_{i=1}^{m} \sqrt{\Gamma_{i1}})^2 > (\sum_{i=1}^{m} \sqrt{\Gamma'_{i1} + \Gamma'_{i2}})^2 = (\sum_{i=1}^{m} A_{i-1} A_i)^2$$

, which equals to the sum of squares of all red segments. Since the shortest distance

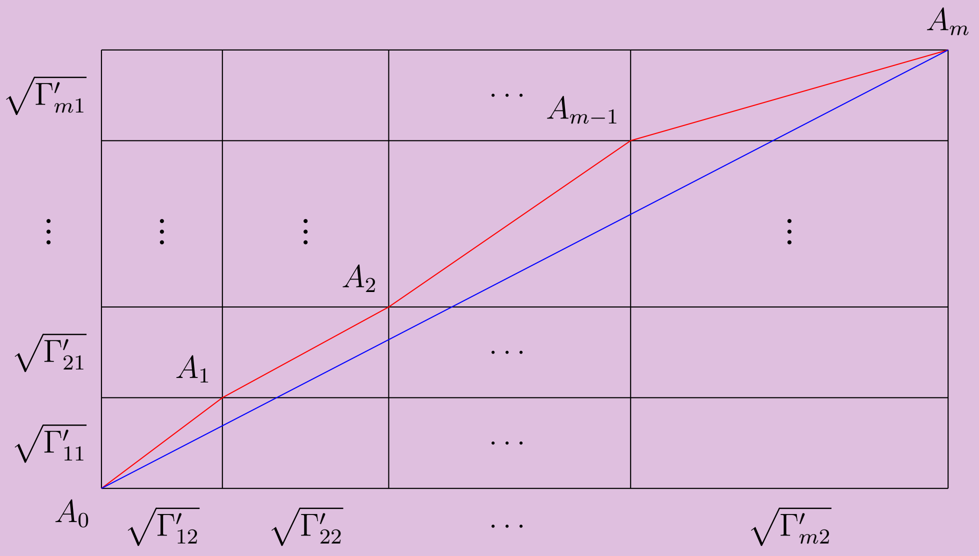

Figure 3: Proof by shortest distance

between two points is a line-segment, it holds that $S_1^2 > S_1'^2 + S_2'^2$.

For the cases where $k > 2$ DApps are split, we can regard it as successive splits and iteratively use the result on $k = 2$.

So the property is proved.

$\square$

A.3 Proof of Corollary 2

Proof. For any voter that is bought over by d_1's developer, before splitting, we can regard the voter as a normal voter who values all DApps with weight vector $(1, 0, 0, \ldots, 0)$, as he gives his all voting capacity to d_1. Suppose now d_1 is split into k DApps and the bough-over voter's contributory values to the k DApps are $\Gamma_{t1}, \ldots, \Gamma_{tk}$, whose sum is fixed. According to the condition that the equality holds for Cauthy's inequality in the proof of Property 3, the voter can be regard as a normal voter who values all DApps with weight vector $(\sqrt{\Gamma_{t1}}/C, \sqrt{\Gamma_{t2}}/C, \ldots, \sqrt{\Gamma_{tk}}/C, 0, 0, \ldots, 0)$, where $C = \sum_{j=1}^{k} \sqrt{\Gamma_{tj}}$. That is, the voter values the split DApps with weights according to kind of proportion and values all other DApps with 0 weights. [16]. Since

$$\sum_{j=1}^{k} \sqrt{\Gamma_{tj}}/C = 1$$

[16]Note that scaling all weights by a constant does not effect the results, since the total contributory value of the voter only depends on the proportions of weights to total weights

so the case in this corollary can be reduced to the case about normal voters(Property 4). So the corollary is proved. □

A.4 Proof of Property 5

Proof. We first consider the case that a voter splits his account into two sub-accounts. We fix the actions of other voters. Let c to be the original account and a, b to be the split sun-accounts, S, S' to be the ranking score of the DApp that the voter plans to vote before and after splitting respectively, U, U' to be the final reward of the developer of the DApp that the voter plans to vote before and after splitting respectively. By definition, we have

$$S = \sqrt{\Gamma_c} + O, \quad S' = \sqrt{\Gamma_a} + \sqrt{\Gamma_b} + O$$

, where O is the sum of contributory values by other voters, which is fixed.

By 11 it holds that $S < S'$. That is, the rank of the DApp does not increase.

In the meanwhile, by definition,

$$U = \frac{S}{S+P}\lambda M, \quad U' = \frac{S'}{S'+P}\lambda M$$

, where P is the sum of squares of other DApp's ranking score, which is fixed.

Since $S < S'$ it holds that $U \leq U'$. That is the developer's final reward does not increase.

For the cases where $k > 2$ sub-accounts are split, we can regard it as successive splits and iteratively use the result on $k = 2$.

□