Nebulas 星云链技术白皮书

基于价值尺度的区块链操作系统及搜索引擎

Nebulas Team

2017 年 9 月
v1.0.1

摘要

比特币和以太坊系统分别给区块链世界带来“去中心化现金”和“智能合约”技术。如今，区块链技术及产业已经取得了长足的发展和繁荣，各种应用场景、商业需求层出不穷。随之而来的，我们发现，已有的区块链技术已经不能满足日益增长的用户需求，总的来说，区块链技术面临着价值尺度缺失、自我进化及生态建设三方面的挑战。

本文介绍了星云链的技术架构设计，意图构建一个能够量化价值尺度、具备自进化能力，并能促进区块链生态建设的区块链系统，主要内容包括：

- **定义价值尺度的星云指数 Nebulas Rank(NR)** (§2)，通过综合考虑链中各个账户的流动性及传播性，NR 试图为每个账户建立一个可信、可计算及可复现的普适价值尺度刻画。可以预见，在 NR 之上，通过挖掘更大纵深的价值，星云链的平台上将会涌现更多、更丰富的应用。
- **支持核心协议和智能合约链上升级的星云原力 Nebulas Force(NF)** (§3)，帮助星云链自身及其上的应用实现自我进化，动态适应社区或市场变化，从而使得星云链及应用将会有更快的发展速度和更大的生存潜力。开发者亦能够通过星云链构建更丰富的应用，并进行快速迭代。
- **开发者激励协议 Developer Incentive Protocol(DIP)** (§4)，为了更好地建立区块链应用生态环境，星云链将通过星云币（NAS）来激励为生态助力的优秀应用开发者，促进星云链更加丰富多元的价值沉淀。
- **贡献度证明共识算法 Proof of Devotion(PoD)** (§5)，从星云链生态健康自由发展出发，星云链提出了共识算法的三个重要指标，即快速、不可逆和公平性，PoD 通过融合 PoS 和 PoI 的优势，结合星云链中的价值尺度，在保证快速和不可逆的前提下，率先加入了公平性的考量。
- **去中心化应用的搜索引擎** (§6)，基于我们所定义的价值尺度，星云链构建了一个针对去中心化应用的搜索引擎，帮助用户在海量区块链应用中，找到符合用户期望及应用场景的应用。

目录

术语表（按首字母排序）

BFT	Byzantine Fault Tolerant，拜占庭容错算法
DIP	Developer Incentive Protocol，开发者激励协议
NF	Nebulas Force，星云原力
NNS	Nebulas Name Service，星云链域名服务
NR	Nebulas Rank，星云指数排名
NVM	Nebulas Virtual Machine，星云链虚拟机
PoD	Proof of Devotion，贡献度证明
PoI	Proof of Importance，重要度证明
PoS	Proof of Stake，股权证明
PoW	Proof of Work，工作量证明
SCR	Smart Contract Rank，智能合约贡献度排名
SCS	Smart Contract Score，智能合约贡献度评分
WAA	Weekly Active Addresses，周活跃地址

1 简介

1.1 区块链技术简介

2008 年 10 月 31 日，中本聪（Satoshi Nakamoto）提出了比特币 [38] 的设计白皮书，从此我们迎来了一个有区块链的世界。比特币作为区块链的太初应用，践行了其作为“一个去中心化电子现金系统”的初衷。比特币的产生不依赖于任何机构，而是根据特定算法，依靠大量计算产生，保证了比特币网络分布式记账系统的一致性。以太坊 Ethereum [10] 更进一步，为我们提供了一个可以运行具有图灵完备性的代码的通用区块链框架。区块链是以比特币和以太坊为代表的数字加密货币体系的核心支撑技术，通过运用数据加密、时间戳、分布式共识和经济激励等手段，在节点无需互相信任的分布式系统中实现去中心化信用的点对点交易、协调与协作，从而解决中心化机构普遍存在的高成本、低效率和数据存储不安全等问题。

需要说明的是，区块链技术本身不是一个全新的技术创新，而是作为一系列技术的组合 (包括点对点通讯，密码学，块链数据结构等) 产生的模式创新。

1.2 商业和技术挑战

以区块链技术为代表的去中心化，自主治理的系统，正在引起越来越多人的重视和研究。当前全球区块链项目已经超过 2000 个，全球加密数字资产总体价值达到 900 亿美元。区块链/数字资产领域的用户人群也正在快速增加。从 2013 年初的全球 200 万用户，到 2017 年初的 2000 万用户。我们认为，在 2020 年左右，全球区块链/数字资产用户会达到或超过 2 亿。在 2025 年前后，全球用户有望达到 10 亿规模。

随着区块链技术的普及，越来越多区块链技术之上的应用场景被挖掘出来。区块链技术的应用场景已经从最初的数字化货币本身逐步扩展到更多的场景及用户群体中。例如，以以太坊为代表的社区在区块链技术中引入智能合约的概念，Ripple 则使用区块链技术实现了全球的结算系统。随着应用场景的多样化，用户对区块链技术的诉求也日益增加，我们已经看到很多挑战。

价值尺度的缺失 我们认为，区块链世界需要一个普适的价值尺度，来衡量用户和智能合约的价值，上层应用可以在这个普适的价值尺度上结合自身场景挖掘更深层次的价值，这将带来更多的商业模式的创新，就像 Google 在互联网世界的兴起一样。

区块链系统的升级 不同于普通软件的版本迭代，区块链系统由于其天生的去中心化特性，无法强制用户升级其客户端及协议。因此，区块链系统中的协议升级往往会引发区块链“硬分叉”或“软分叉”，从而造成巨大的损失，这更进一步限制了区块链系统的应用场景。以比特币为例，社区关于区块扩容至今仍然存在巨大的争议，导致比特币协议进化缓慢，区块容量严重不足，出现过近 100 万笔交易在交易池等待被写入区块。用户很多时候不得不额外支付高昂的“交易加速费”，严重损害体验性能。另外，从以太坊的“硬分叉”来看，虽然暂时解决了 The DAO 问题，但是产生了 ETH 和 ETC“重资产”和社区分裂的“副作用”。

区块链应用生态环境的建立 随着区块链上各种应用 (DApp) 的快速增长，良好的生态环境是提高用户体验的根本所在。这包括用户如何在海量的区块链应用中检索自己期望的 DApp，如何激励开发人员为用户提供更多的 DApp，以及如何帮助开发人员更快的构建更好的 DApp。以以太坊为例，基于以太坊的

DApp 总数已经数十万个，试想如果区块链世界中的 DApp 接近苹果 App Store 里应用总量规模的话，如何发现并找到用户期望的 DApp 就是个很大问题。

1.3 星云链设计原则

面对上述机遇和挑战，我们要设计一个基于价值激励的自进化区块链系统。具体来说，我们有以下设计原则：

- **公正的排名算法，定义价值尺度**

  我们认为，区块链世界需要一个普适的价值尺度，用于衡量区块链底层简单数据的价值，发现信息的更高维度信息，从而探索并挖掘区块链世界的更大价值。类似 Google 的 PageRank [9][46]，我们也提出区块链世界的 NR(Nebulas Rank，星云指数) (见 §2) 算法，综合考虑了区块链上的资金流动性，以及资金传播的速度、广度和深度，给区块链用户做公正的排名。NR 是星云链赋予区块链世界的价值尺度，用来帮助开发者结合自身场景有效衡量区块链中各个用户、智能合约、DApp 的重要性。NR 有巨大的商业潜力，可以用在搜索、推荐、广告等领域当中。

- **区块链系统及应用的自我进化**

  我们认为，一个良态的区块链系统及其上的应用应该能够实现自我进化。在较少外部干涉的情况下，实现更快的计算、更强的系统、及更好的体验。我们将这种自我进化的能力称之为 NF(Nebulas Force，星云原力) (见 §3)。在星云链的系统架构中，通过在区块结构上的良好设计，基础协议将会成为链上数据的一部分，并通过链上数据的追加实现基础协议的升级；对于星云链中的应用（智能合约），星云链通过在智能合约底层存储支持状态变量可跨合约访问的设计，完成智能合约的升级。具备自我升级进化能力的星云链，未来比其它的公有链具有更快的发展速度和更大的生存潜力，同时使得开发者面对漏洞，能够更快的响应和升级，避免黑客事件给用户带来巨大的损失。

- **区块链应用生态环境的建设**

  在星云链中，我们提出了基于账户贡献度的 PoD(见 §5) 算法，利用 NR 的价值尺度评估找出对生态贡献度较高的账户，平等地赋予记账资格，遏制记账权被垄断，并且融合 PoS 中的经济惩罚，防止公链被恶意破坏，为生态自由发展助力。既能保证较快的共识速度，又能比 PoS 和 PoI 更抗作弊，对区块链生态的发展有良好的促进作用。

  在星云链中，我们提出面向智能合约和 DApp 开发者的 DIP(Developer Incentive Protocol, 开发者激励协议) (见 §4)。DIP 的核心思想是对社区贡献度高的智能合约或 DApp 的开发者，给予他们相应的开发者激励。激励由记账人负责写入区块。

  基于 Nebulas Rank 机制，星云链进一步包含了搜索引擎 (见 §6)，以帮助用户更好的探索区块链中的高价值应用。

考虑到以太坊已经有巨大的生态，是一个非常成功的公有区块链平台。星云链希望尽可能的借鉴以太坊等其他区块链系统的优秀设计，从智能合约编程上完全兼容以太坊，使得基于以太坊开发的产品能够零成本的迁移到星云链上。

基于上述设计原则，我们试图构建一个基于价值尺度的区块链操作系统及搜索引擎。本白皮书详细描述了星云链中关于技术的细节，其中 §2描述了一种可能的价值尺度及其算法 Nebulas Rank，§3描述了星云

链的自我进化能力 Nebulas Force, §4、§5、§6、§7描述了星云链在建设区块链应用生态圈的的设计和构想，最后，§8描述了星云链的代币 NAS。

2 Nebulas Rank 星云指数

2.1 星云指数总体设计

当前区块链技术和生态已具有相当规模，但是人们对其认识还处于扁平化阶段，尚不存在一个合理的方式去评估链上实体（例如用户地址）的价值。因此，我们试图在区块链世界提出一个普适的价值尺度，通过对链上行为的挖掘利用，将每个实体（用户地址）的重要程度量化为 **Nebulas Rank** 的指标形式。**Nebulas Rank** 旨在实现双重目标：

- 作为原生的价值尺度，**Nebulas Rank** 可以成为诸多基础场景的核心算法，如共识算法 (见 §5)、开发者激励 (见 §4) 和区块链搜索引擎 (见 §6)，等等；
- **Nebulas Rank** 可以启发人们对区块链的生态现状定义更多样化的价值尺度，同时产生更深层次、差异化和结构化的认识，进而在商业决策和研究活动中有明确的方向；

基于上述目标，我们为 **Nebulas Rank** 定义了三重价值尺度：

- 流动性，即交易的频次和规模，是 **Nebulas Rank** 的第一个考量维度。本质上来看，金融是通过资本流动推动社会资源的有效配置、推动经济发展的社会活动。区块链构建了一个价值网络，更多的交易数量、更大的交易规模产生更高的流动性，更高的流动性口进一步提升了交易数量、交易规模，从而进一步强化了它们的价值，形成了一个完整的正向反馈机制。
- 传播性，即资产流动的广度和深度，是 **Nebulas Rank** 的第二个考量维度。在社交网络中的传播性，即信息的传播速度、广度和深度，是网络质量和用户增长的关键指标。我们在区块链世界将会看到同样的模式，强大传播性意味着资产流动的广度和深度，会提高区块链世界的资产质量，提升资产规模。
- 互操作性是 **Nebulas Rank** 的第三个考量维度。在互联网的早期，我们只有简单的网站，孤立的信息。现在，网络上的各种平台信息开始相互调用，数据孤岛逐渐被打破。这一趋势可以被理解为识别更高维度信息的过程。我们认为区块链世界也会遵循同样的发展规律，但其过程速度会更快。用户的资产、智能合约和 DApp 之间的信息会越来越丰富，高维信息间的互操作会越来越频繁，因此更好的互操作性将会变得越来越重要。

我们选择使用链上的交易记录作为 **Nebulas Rank** 算法的数据来源。因为相比于现实世界，区块链世界的“行踪”更为清晰和可信：链上的交易数据忠实地记录了用户之间的每笔转账、以及每次对“智能合约”的调用情况。然而根据交易数据设计算法并非易事。因为相比于现实世界，区块链世界的交易具有天然的匿名性，同时数据量规模更为庞大。因此我们试图为 **Nebulas Rank** 刻画如下的性质：

- 可信。实体如果要提高自己的价值，必须付出一定的成本，这样可以确保算法的结果能够甄选出真正具有较高价值的用户。一方面，在共识算法和开发者激励等场景下，可信的结果鼓励用户诚实地贡献以实现正向反馈，另一方面，可信的结果提供了有意义的用户分层结果，能提供更好的决策基础。
- 可计算。作为基础字段，**Nebulas Rank** 的结果应该以最快最直接的方式提供，因此其实现需要较低的计算复杂度；
- 可复现。由于共识算法和开发者激励等场景的需要，**Nebulas Rank** 需要保证在所有客户端的计算结果完全一致。

接下来我们设计 **Nebulas Rank** 的基本框架。首先，我们用图的形式表现交易记录。在交易图的基本定义中，每个节点代表一个实体，每条边代表实体之间的转账行为 [57]。交易图表现了转账行为导致资产流动的性质，有助于表示上述价值尺度中流动性和传播性的概念。同时，图的形式也能方便刻画合约之间的互操作性。得到图之后便可以用复杂网络的中心性测度来对重要网络节点进行排名，对 **Nebulas Rank** 的情景，我们采用 LeaderRank [14][32]，并取得比 Google 的 PageRank 和 NEM [39] 的 NCDAwareRank 更好的排名效果。

2.2 交易图

本小节介绍如何从交易数据构造交易图。

首先, 对于任何一个时刻 t_0，我们记录时间 $[t_0 - T,\ t_0]$ 内 (一般情况下，T 是值为一个月的常量)，所有区块包含的个人用户之间的有效交易记录。其中，每笔交易记录可以表示成一个 4-元组 (s, r, a, t)，s 是转出地址，r 是转入地址，a 是交易金额，t 是这笔交易的区块时间。并且，我们定义任何一个交易是有效的，当且仅当 $a > 0$ 和 $s \neq t$。因而，$[t_0 - T, t_0]$ 内所有的有效交易记录可以表示成一个 4-元组的集合:

$$\Theta(t_0) = \{(s, r, a, t) \mid t_0 - T \leq t \leq t_0 \ \wedge \ a > 0 \wedge s \neq t\} \tag{1}$$

基于 $\Theta(t_0)$，我们可以构造有向加权简单图 $G = (V, E, W)$，其中，V、E 和 W 分别表示节点集合、有向边集合和边的权值集合。V 中的每个节点都代表一个账户的地址，E 中的每条有向边代表两个用户之间的交易关系。我们定义以下公式合并对应交易的最高 K 个金额作为权值 w_e :

$$w_e = \sum_{i=1}^{K} a_i,\ s.t.\ a_i \in A_e \tag{2}$$

其中 A_e 是一个由所有在 $\Theta(t_0)$ 出现的相关交易金额 a 组成的有序集合:

$$A_e = \{a_i \mid e = (s, r) \wedge (s, r, a_i, t) \in \Theta(t_0) \wedge (a_i \geq a_j, \forall i \leq j)\} \tag{3}$$

另外，记 $N = |V|$，$M = |E|$，其中 $|.|$ 表示集合的范数。因而，总共有 N 个地址和 M 条有效交易。在下文中，为了表述便利，所有节点都以一个 1 到 N 之间的整数来表示。

接着，对每个节点，根据其在 $[t_0 - T,\ t_0]$ 的转入转出记录，计算它的“币龄”，记作 C_v；根据转入转出的金额总量，根据图1所示的“鼓励函数”计算出每个节点鼓励值 E_v[1]。最后使用目标节点的 C_v 和 E_V 对边权进行衰减。

最后，取整个交易图的最大弱连通分支，将分支之外的节点删除。被删除的节点不参与后面的排名，默认赋予最低的重要性分数。

上述构造交易图的方式有助于实现 §2.1定义的“可信”性质，具体效果见 §2.4的讨论。

我们收集了 Ethereum 主链从 #3629091（约 2017 年 5 月 1 日）到 #3800775（约 2017 年 5 月 31 日）共 171,684 条区块交易数据，按照本小节介绍的方式构造了交易图（T 为 30 天，$K = 2$），可视化结果如图2所示。

[1]鼓励函数可以表示为两个正态分布的线性加和形式，在转出金额为零时以及转出金额少于转入金额的某点取得最大值

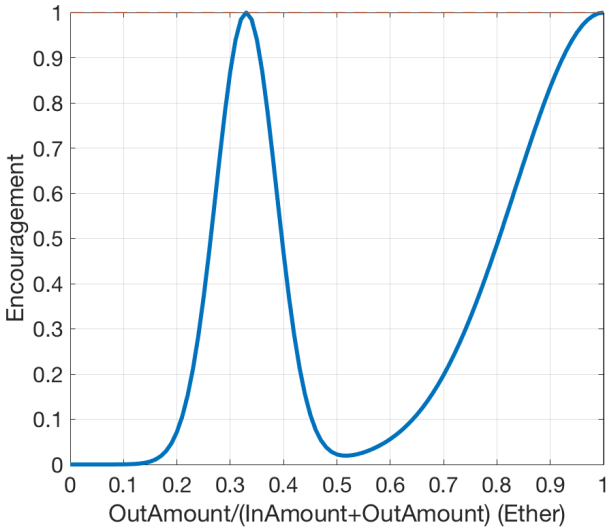


图 1: 鼓励函数

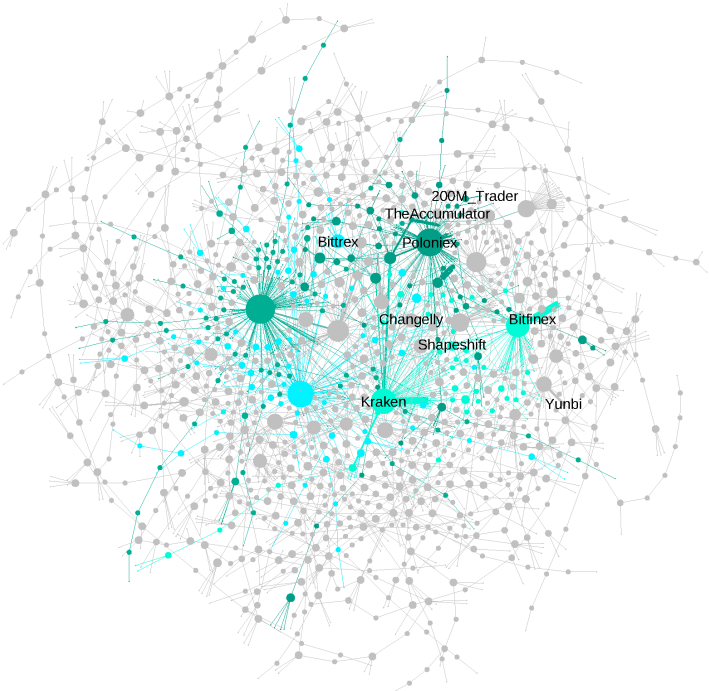


图 2: 交易图（部分节点）可视化。节点对应地址的交易规模（资金流入流出）越大，其节点的出入度越大，图示直径越大；可以看出很多已知交易所地址和大量其他地址有转账交互，但也有部分参与大规模交易的地址身份未知，其角色还有待挖掘。

2.3 排名算法

本小节介绍如何在构造的交易图中进行节点重要性排名。

我们采用 LeaderRank [14][32] 作为核心算法。根据 LeaderRank 的算法，在交易图中添加编号 0 的背景节点，并且建立背景节点 0 和其他任何节点 i 之间的双向边（$1 \leq i \leq N$）。我们定义每一条与背景节点相关的有向边的权值如下：

$$w_{i,0} = \alpha(max\{\sum_{w_{j,i}\neq 0} w_{j,i} - \sum_{w_{i,j}\neq 0} w_{i,j}, 0\} + \lambda C), \forall i \in [1, N] \tag{4}$$

$$w_{0,n} = \beta(\sum_{w_{n,m}\neq 0} w_{n,m} + \mu C), \forall i \in [1, N] \tag{5}$$

其中，C 表示集合 $\{w_{i,j}|w_{i,j} \neq 0, 0 \leq i, j \leq N\}$ 的中位数。α, β, γ 和 μ 都是参数。

上述赋权方式可以理解为，入度更大的节点接收了更多来自背景节点的入边权值，“净流入”（即入度减出度）更大的节点向背景节点输出了更强的链接。

LeaderRank 的计算过程和 PageRank 基本相同，可以理解为求马可夫链的稳定状态。所不同的是，添加了背景节点之后，不再需要考虑 PageRank 的 damping factor[9][46]，并且所有节点天然地就具有少许出入度。按 (7) 构造矩阵 H 后，进行如 (6) 所示的迭代（初始值设置见 (8)），直至收敛为止，最后去掉背景节点的评分即为交易图各点重要程度得分 P^*。

$$P^{t+1} = H \times P^t; P^1 = [\frac{1}{N}\ \frac{1}{N}\ \cdots\ \frac{1}{N}\ 0]^T \tag{6}$$

$$h_{ij} = \frac{w_{(j\ i)}}{\sum_k w_{(j\ k)}} \tag{7}$$

$$\forall v \in V\ P_v^* \leftarrow P_v^* + \frac{P_{\mathcal{G}}^*}{N} \tag{8}$$

我们认为，LeaderRank 可以较好地满足 §2.1定义的价值尺度和算法性质：

- LeaderRank 可以理解为在资金流动网络动态平衡状态下通过每点的流量，这契合了 **Nebulas Rank** 的“流动性”、“传播性”和“互操作性”等尺度；
- (5) 和 (4) 所定义的赋权机制可以加大攻击难度（见 §2.4的讨论），更好地满足“可信”性质；
- LeaderRank 的计算可以用迭代方式完成，由于网络的稀疏性，矩阵运算复杂度不高，能够满足“可计算”和“可复现”的性质。

2.4 抵抗操纵

可信性，即抵抗操纵的能力，是 **Nebulas Rank** 最重要同时也是最具挑战的目标。部分恶意操纵的手段有以下几种：

1. 环形转账，攻击者沿环形拓扑，让同一笔资金不断流过对应的边，以提高边权；

2. 向其他任意账户转钱，提高出度，并且提高资金流出的传播性；

3. 控制多个账户形成独立分支，伪造中心节点；

4. 频繁同权威交易所账户交易，多次在交易所账户中取入取出同一笔资金，获得较好的网络结构位置。

Nebulas Rank 通过以下几个机制，可以弱化操纵效果：

- 由于设置了长为 T 的滑动时间窗，攻击者无法在短期内迅速提高自己的排名；
- 因为边权由最高的几次交易金额决定，在一个环状拓扑内的多次转账不能无限提高各边权值，同时，从 §2.2采集的数据看，有 91% 的双方只进行了一或两次交易，因此 K 取 2 可以最大限度保留边的强度信息，同时抵抗环形转账攻击；
- 为了获得更高"币龄"，用户需要让资金在自己的账户内"停留"一段时间，进而拖慢攻击者的交易速度；
- 为了获得最大的"鼓励值"，如图1所示，账户需要在当前周期内储蓄全部收入，或者只转出少于收入一半的金额，因此伪造资金流动时，攻击者的本金会因各个账户的储蓄效应而迅速衰减；
- 由于取了最大弱联通分支，伪造的独立分支会被视为噪声而过滤清除。从 §2.2采集的数据看，交易图有 $453,285$ 个节点，$970,577$ 条边，有 $1,169$ 个弱分支，其中最大弱分支有 $449,746$ 个节点，占全体节点的 99.2%，次大弱分支有 133 个节点，仅占全体节点的 0.03%，因此取最大弱联通分支可以最大限度保留网络正常成分并且过滤噪声数据；
- 相对于 PageRank 和 NCDawareRank [43] 等网页排名算法，(5) 和 (4) 所定义的赋权机制对低入度的节点评分偏向"保守"，即低入度的节点获得来自背景节点的链接更弱。在区块链交易图中，低收入节点更容易被伪造，而频繁向其他任意地址转账也并不能提高收入，因此使用 **Nebulas Rank** 的方法可以提升操纵难度；

接下来，基于 2017 年 5 月份 Ethereum 的交易图（见 §2.2），我们展示一系列结论。

首先，我们列出 **Nebulas Rank** 排名部分地址，如表1所示[2]，可以看出，交易所账户以及部分交易吞吐量较大地址的排名较为靠前。

[2]备注来源: Etherscan [16]

表 1: **Nebulas Rank** 排名前 10 名及部分其他地址

排名	地址	Nebulas Rank	备注	转出金额 (Ether)	转入金额 (Ether)
1	0x267be1c1d684f78cb4f6a176c4911b741e4ffdc0	0.449275	Kraken_4	3214232.06	350008.00
2	0xd4c5867cec094721aabc3c4d0fd2f2ac7878c79a	0.093798		58000.00	100947.00
3	0x027beefcbad782faf69fad12dee97ed894c68549	0.049277	QuadrigaCX	207440.11	65606.40
4	0x0ee4e2d09aec35bdf08083b649033ac0a41aa75e	0.046831		56465.00	60087.96
5	0xc257274276a4e539741ca11b590b9447b26a8051	0.037628		1071105.93	1434106.72
6	0xa53e0ca7d246a764993f010d1fde4ad01189f4e6	0.033488		7764.68	3201.00
7	0xf259e51f791e9ed26e89b6cae4a7c6296bfbd0b8	0.033481		3307.00	7731.30
8	0xf195cac8452bcbc836a4d32cfb22235af4ac1e9c	0.026343		10863.87	2315.69
9	0x94435d12c51e19d5b5c8656763f9069d37791a1a	0.024970		12938.58	15858.90
10	0x7580ba923c01783115d79975d6a41b3d38eff8d5	0.021670		263000.00	364793.49
16	0xcafb10ee663f465f9d10588ac44ed20ed608c11e	0.004995	Bitfinex_1	360000.00	1435858.40
51	0xd94c9ff168dc6aebf9b6cc86deff54f3fb0afc33	0.000868	yunbi_1	1179224.74	1202539.53
64	0x70faa28a6b8d6829a4b1e649d26ec9a2a39ba413	0.000590	Shapeshift	52501.81	651933.49

接着，对比交易金额和 **Nebulas Rank** 的关系。由于区块链交易可以理解为“资金交换”类型的网络流，根据 Borgatti [5] 的研究工作，节点的度，即邻接边权值之和，是适用于此类网络的一个合适的中心性测度。以每个节点为中心思考，度，即交易吞吐金额（转入转出金额的和），是节点一跳局部信息的体现，直接反映了对应地址的历史资金流量，因此应该作为衡量排名算法的基准。交易金额与 **Nebulas Rank** 的关系如图3所示：没有节点能够以较低的交易金额获取靠前排名，而交易金额较大的节点也要满足一定条件才能取得高的排名，这可以大致印证 **Nebulas Rank** 的可信性。

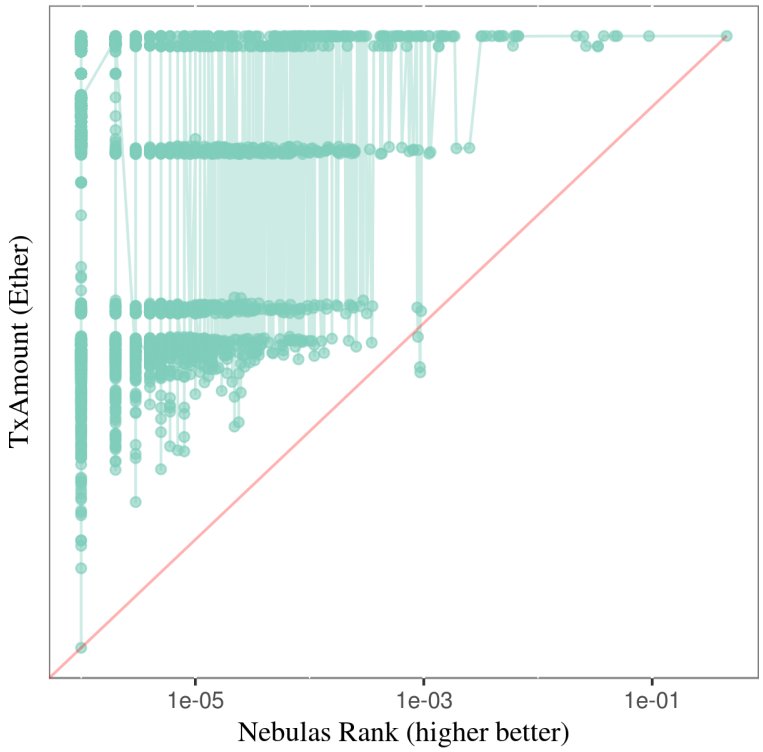


图 3: Nebulas Rank v.s. 交易金额。横坐标为 Rank 值，纵坐标为交易金额，均为对数形式。图中斜线代表交易金额和 rank 值成正比例的情况。好的算法应该使得数据点尽可能少地落在斜线的右下方，以免低交易吞吐金额的节点取得好的重要性排名。

在之前的部分，通过简单的分析可以推断，本小节开始所述的前三种攻击方式各自都可以被特定的处理手段有效过滤掉。因此最后，我们只需仿真最后一类攻击，攻击者选定某权威交易所节点，在排名计算周期内创造 X 次环形交易，每次交易时，攻击者先经由某新建地址向交易所节点转入 YEther，之后经另一新建节点从交易所节点取出 YEther，攻击方式示意图如图4所示。此类攻击利用了某些交易所可以低成本建立转账链接的性质，虽然合法地址也可能和交易所频繁交易，但此类攻击的行为并未促进资产的有效流动，因此应该与合法行为在一定程度上区分开。

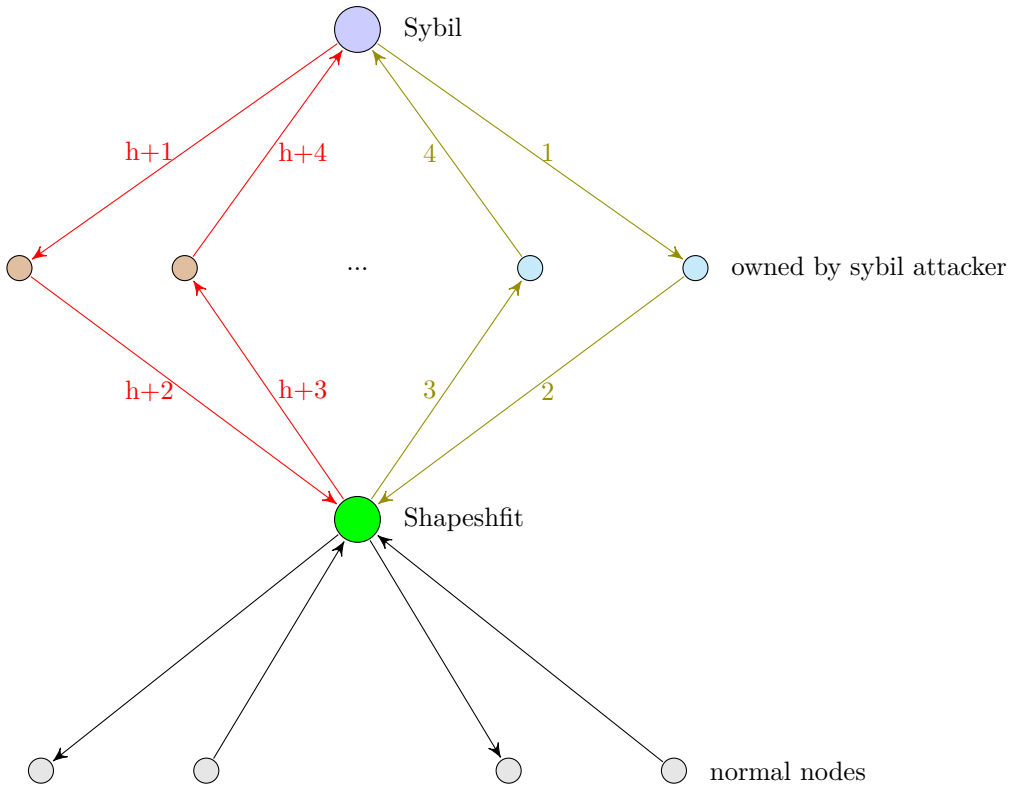
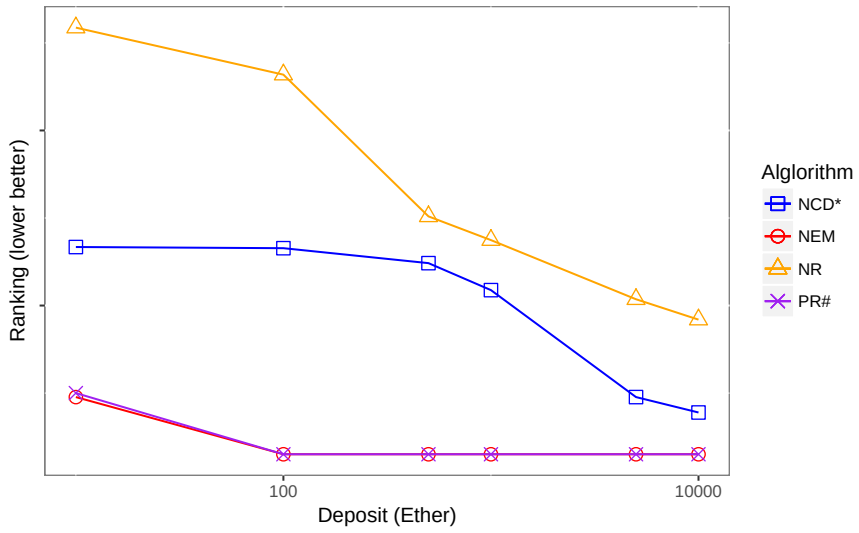
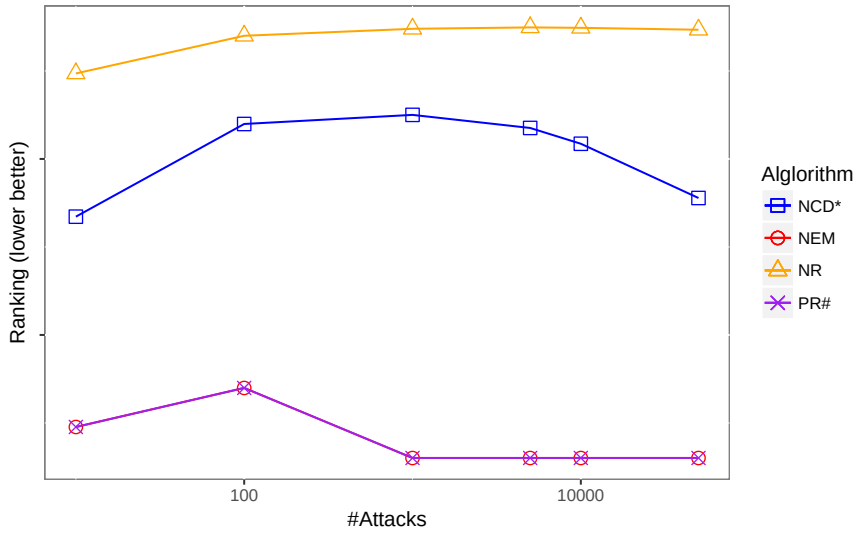


图 4: 将交易所纳入环形交易的攻击示意图。图中给出了第 1 次和第 h 次环形转账的示意，选择的权威交易所节点为 Shapeshift；边的标签代表时序；Sybil 及其附属节点和 Shapeshift 节点之间每条边所代表的转账金额为 YEther；攻击者共进行 X 次环形转账。

测试选择的权威交易所为 Shapeshift（0x70faa28a6b8d6829a4b1e649d26ec9a2a39ba413），结果如图5所示：1）图5a展示的结果表明，随着攻击者投入本金提高，使用任何算法都无法阻止攻击者排名变得越来越好。而使用 §2.2所述的方式构造交易图可以使得攻击效果显著减弱，并且 **Nebulas Rank** 可以既充分重视高交易量地址，又可以在一定程度上更好地抵抗操纵；2）图5b展示的结果表明，随着攻击者攻击次数的增加，§2.2所述的交易图构造方式可以使得攻击者排名下降，原因是我们的交易图加权方法考虑了币龄和鼓励函数等因素，同时 **Nebulas Rank** 可以强化这些因素的影响以抵抗操纵。



(a) 攻击次数固定为 5000 次，攻击投入本金对排名的影响
(横纵坐标均为对数形式)



(b) 攻击投入本金固定为 Ξ1000，攻击次数对排名的影响
(横纵坐标均为对数形式)

图 5: 抗操纵测试结果。攻击方式如图4所示，横轴为攻击成本，纵轴为攻击者主节点的排名（排名越靠后纵坐标越高，算法抵抗操纵能力越好）。NR：交易图如 §2.2所述，排名算法按照 §2.3所述；NCD*：交易图如 §2.2所述，NCDawareRank 排名算法；NEM：交易图如 [39] 所述，NCDawareRank 排名算法；PR#：交易图如 [39] 所述，PageRank 排名算法。PageRank 的 damping factor 为 0.15；NCDawareRank 使用 pscan [12] 社群划分算法，$\eta = 0.75$，$\mu = 0.1$

2.5 相关工作

中心性，作为最核心的节点排名指标，是过去几十年网络科学领域研究最多的一个概念 [40]。丰富的文献引入了大量中心性测度，包括度中心性 [21]，特征值中心性 [4]，Katz 中心性 [28]，接近度中心性 [51]，介数中心性 [22][23][24][44][41]，PageRank [9]，HITS [30]，SALSA [52]，等等。此外还有许多工作试图用统一的框架来对中心性测度作出清晰的分类和综述 [5][6][34]。在设计 **Nebulas Rank** 时，需要首先考察交易图的性质，然后再选用合适的中心性。Borgatti [5] 工作中提到的资金交换网络和区块链交易图最为相近，但是提到的相关算法，如流介数中心性 [24] 和随机游走介数中心性 (又称电流中心性) [41]，复杂度过高，在区块链交易图的规模上不符合 **Nebulas Rank** 的“可计算”性质。

自从比特币 [38] 系统在 2009 年发布以来，研究者们对比特币交易图做了一些试验性和统计上的分析 [50][26][42][2]，同时试图使用交易图的结构来讨论比特币的匿名性问题 [35][45][47][20][19]。在其他加密货币出现并流行之后，交易图分析拓展到了其他的区块链系统 [13][1]。**Nebulas Rank** 采用的交易图概念和这些研究中的大致相同，即 Tschorsch and Scheuermann [57] 所总结的“实体图”。即每个用户实体被映射为一个节点。而每条有向边则代表两个用户之间的交易强度。事实上，早在如比特币一样的区块链系统发明之前，学者们就尝试对银行和国际交易的金融网络进行研究 [49][8][53][3][17][37][7][31][54]。和区块链交易图相比，这些早期研究的网络还包含了额外的借贷活动。并且这些网络的规模非常小。总之，现有研究几乎没有专门针对大规模区块链交易图提出排名方法。

和 **Nebulas Rank** 最相关的工作是 NEM [39] 的 Proof-of-Importance 机制，它采用了 NCDawareRank [43] 作为排名算法。NCDawareRank [43] 利用了网络拓扑的社群效应。Proof-of-Importance 使用 SCAN [58][55][12] 作为社群聚类算法。虽然社区结构在交易网络的确存在并且可以帮助应对欺诈节点，却无法保证同一个实体对应节点一定可以映射到相同社群，因此利用社区划分的结果会提供一定的可操纵空间。此外 Fleder, Kester, and Pillai [20] 使用 PageRank 来帮助发现感兴趣的比特币地址并分析它们的活动，但他们的工作仍然将人工主观分析作为主要方法，PageRank 只起到辅助作用，这也和 **Nebulas Rank** 的目标不符。我们采用的算法是 LeaderRank [14][32]。它是 PageRank 的一种拓展形式。在 PageRank 中，每个节点都有相同的随机跳转概率，LeaderRank 是对跳转概率一种简单但有效的改进，通过在网络中添加背景节点和加权的双向链接，可以使得不同节点具有不同的随机跳入和跳出概率。**Nebulas Rank** 的加权机制部分参考了 Li *et al.* [32] 的设计，使得入度更大的点更有可能被随机跳转到达。通过添加 LeaderRank 算法中的背景节点，可以取得更符合区块链场景的排名结果。

3 Nebulas Force

我们使用星云原力 (Nebulas Force, NF) 来描述区块链系统及应用的进化能力。星云原力做为驱动区块链系统及应用发展的第一推动力，包括三个方面：星云链虚拟机 NVM(Nebulas Virtual Machine)，区块链系统中核心协议的升级，以及运行在区块链系统之上的智能合约的升级。

在星云链中，我们将引入 LLVM 来实现星云链虚拟机 NVM。核心协议和智能合约代码将会编译成 NVM 字节码，通过 LLVM 即时编译 (Just-in-time compilation) 功能，实现其动态编译和优化，最终在沙箱环境中执行。同时借助于 LLVM 的模块化架构，开发者可以用熟悉的编程语言实现更高性能和更安全的智能合约，给用户带来更丰富的去中心化应用。

对于星云链中的核心协议升级，星云链将核心协议加入到区块中，通过对链上数据的追加实现核心协议的升级，避免开发者和社区的分裂或分叉的可能性。随着星云链社区的发展，NF 及基础协议升级能力将逐步开放给社区，由社区定义星云链的进化方向并实现其升级目标。借助于 NF 这个核心技术和开放性的理念，星云链将会具有持续的进化空间和无限的进化可能。例如，NR 算法参数、PoD 激励金额、共识算法及新代币的生产速度等一系列参数，都可以在星云链的发展过程中逐渐调整，而不需要大部分客户端代码的升级。

智能合约通常被认为是永久性的，不支持升级。星云链通过在智能合约底层存储支持状态变量可跨合约访问的设计，完成智能合约的升级，这种解决方案对开发者友好，使得开发者面对漏洞，能够更快的响应和升级，避免黑客事件给用户带来巨大的损失。

3.1 NVM 星云链虚拟机

我们将引入 LLVM [33] 做为 NVM 的核心组件，并使用 LLVM Byte Code 做为 NVM 字节码。NVM 字节码通过 LLVM JIT 完成动态编译和优化，运行在 NVM 沙盒环境中。在这种架构设计下，星云链的核心代码、智能合约能直接享受到 LLVM 带来的性能和安全性的不断提升。

LLVM 早期是 Low Level Virtual Machine 的缩写，是一系列高度模块化的编译器和工具链技术的集合，包括 Google、Apple 在内的公司都使用它做为代码编译框架。LLVM 提供了一套中立的中间表示（LLVM IR）和相应的编译基础设施，并围绕这些设施提供了一套全新的编译策略，包括对 LLVM IR 的优化、LLVM IR 到不同硬件平台的代码生成、LLVM IR 到 LLVM Byte Code 的代码生成以及 LLVM Byte Code 在不同硬件平台上通过 LLVM JIT 直接执行。如图6所示

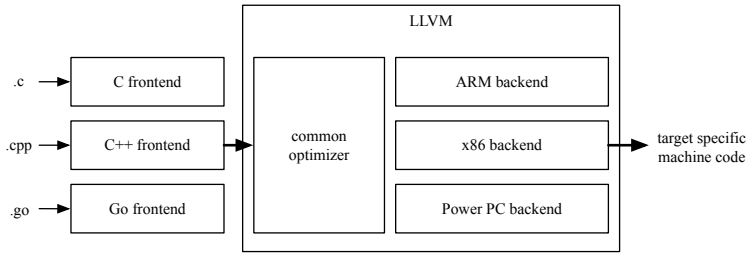


图 6: LLVM

我们依托 LLVM 构建 NVM，如图7所示。首先，我们提供区块链底层 API 库；然后，我们为不同语言（如 Solidity，JavaScript ，C/C++，Go 等）构建生成 LLVM IR 的编译器前端；最后，利用 LLVM 提供的工具链，生成 LLVM Byte Code。最终，LLVM Byte Code 通过 LLVM 的 JIT 引擎运行在 NVM 提供的安全的沙箱环境中。

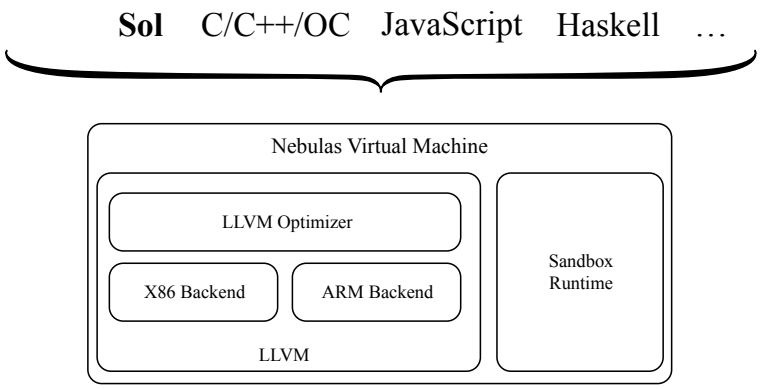


图 7: 星云链虚拟机

NVM 是星云原力的重要基石。新的协议代码或智能合约发布时，NVM 中 LLVM 编译器模块完成新代码的编译得到 LLVM 字节码，然后发布到链上；链上确认后，新代码将由 LLVM JIT 完成编译和优化，然后进入沙箱取代旧代码并被执行，过程如图8所示。

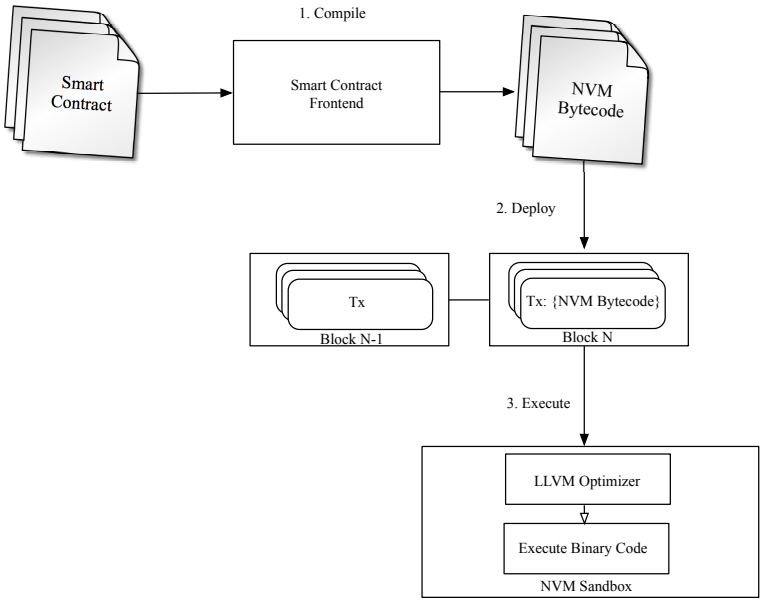


图 8: 星云链虚拟机运行机制

借助于 LLVM（见6），NVM 还支持开发者用其熟悉的编程语言开发智能合约和应用，比如以太坊智

能合约所使用的 Solidity，更加灵活的 JavaScript，甚至是纯函数式语言 Haskell。除了这些通用语言外，NVM 还可以为不同领域和场景提供定制的高级语言，比如面向金融行业的 DSL（领域专有语言）。这类高级语言面向行业、场景高度定制，使得它们更容易被形式化验证，能进一步提高代码健壮性和安全性，更有利于星云链开发者开发出更丰富的智能合约及应用。

3.2 核心协议的升级设计

我们首先给出星云链中的区块结构，然后讨论如何在该区块结构上实现核心协议的升级问题。

区块结构 星云链区块数据结构包含，但不限于以下信息：

- Header：区块头
  - Height：高度
  - ParentHash：父区块哈希值
  - Ts：时间戳
  - Miner：记账人地址
  - Dynasty：区块所处共识朝代
  - Epoch：区块所处共识时代
  - StateRoot：状态根哈希值
  - TxsRoot：交易根哈希值
  - ReceiptsRoot：交易收据根哈希值
  - TransNum：交易数
- Transactions：交易数据 (包含多个交易)
  - From：交易发起人地址
  - To：交易接收人 (普通用户或智能合约) 地址，对于创建智能合约，值为 0 地址
  - Value：转账金额
  - Data：交易的 payload。如果交易为创建智能合约，则为智能合约字节码；如果交易为智能合约调用，包含调用函数名称和入参值
  - Signature：交易签名
  - Gas：燃料上限
  - GasPrice：燃料单价
  - Nonce：标识交易唯一性
- Votes：Prepare 和 Commit 票数统计 (包含多个)，用在 PoD(见 §5) 共识算法中
  - From：投票人
  - VoteHash：投票区块哈希
  - Hv：投票区块所处高度

  - Hvs：投票区块的某祖先高度
  - VoteType：投票类型，Prepare 或 Commit
  - Signature：投票签名
- Protocol Code：核心协议代码 (一个区块中只能有 0 或 1 个)
  - Hash：哈希值
  - Code：核心协议的字节码
  - ValidStartBlock：协议生效起始区块号
  - Signature：签名 (校验是否来自开发者社区保留账号的签名)
  - Version：标识核心协议版本号，每次升级都需要递增，防止恶意记账人回滚到老的 Protocol Code
  - Nonce：标识唯一性
- Nebulas Rank：星云指数 (计算周期为一周一次，大部分区块没有)
  - RankVersion：NR 版本
  - RankRoot：NR 排名哈希值
  - RankRecords：NR 排名记录
    * Address：账号地址标记
    * Score：NR 值

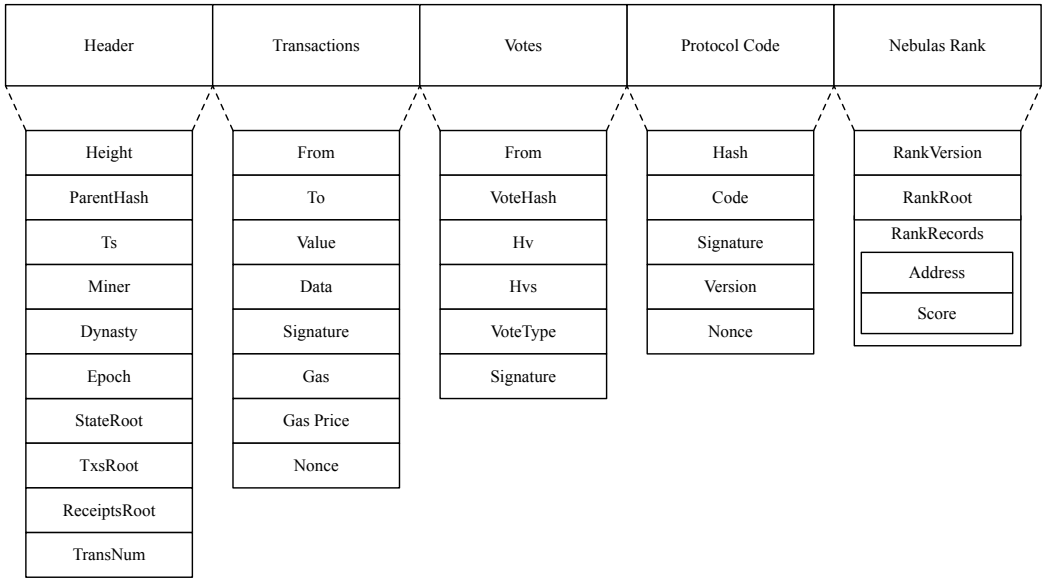


图 9: 区块结构

类似于其他加密数字货币系统，用户和区块链上的互动都是通过特定的交易进行的。用户创建一笔交易，用自己的私钥签名后，发到区块链的任何一个节点中，通过 P2P 网络广播到全网节点。在固定的出块时间

间隔内，由 PoD 共识算法 (见 §5) 指定的记账节点累计这段时间内的所有交易，打包成标准格式的区块，同步到全网所有节点。经各节点独立验证后，加入本地账本，成为全球总账本的一部分。

以太坊中，交易分为两种类型：普通账户交易，智能合约交易。我们在星云链的区块中，增加新的数据类型：核心协议代码和星云指数。核心协议代码作为区块链数据的一部分存储在链上，星云链基础协议的升级，是通过链上数据的追加而实现的。星云指数根据 NR 排名算法，计算出每个周期各个账户的 NR 值存到链上，方便 NR 值的实时调用和历史排名查询。

核心协议的升级 星云链客户端节点从当前最新区块的 Protocol Code 存储区可以取到编译后的虚拟机字节码 (NVM 字节码)，如果当前最新区块没有 Protocol Code 数据，说明核心协议没有变更，就往前追溯到最近区块的 Protocol Code。区块链的核心协议行为都由 Protocol Code 确定，包括验证算法、打包规则、NR 算法、奖励机制等，几乎绝大部分的区块链行为都可以由 Protocol Code 定义。

如果核心协议需要升级，由星云链团队开发，把代码在公开渠道让社区讨论和投票。投票可以通过智能合约或者论坛投票的形式进行，当绝大部分社区成员都同意协议升级，星云链开发组把最新代码打包成 Protocol Code 交易，发布到全网节点，记账节点只要把其包含进区块，就可以在指定区块高度开始生效。这种方式的区块链协议升级，对客户端来说是透明的，无需软、硬分叉。

为了保证核心协议代码是经过授权发布的，Protocol Code 的发布者是星云链核心开发组保留地址，该地址在创世区块内部硬编码无法变更。所有记账节点都会验证 Protocol Code 签名，签名不通过的视为非法数据。

后续的改进措施是把 Protocol Code 的签名校验改成 M-of-N 的多签名形式，这个本身也可以通过 Protocol Code 的升级实现。

3.3 智能合约的升级设计

3.3.1 图灵完备的智能合约编程语言

智能合约是一套以数字形式定义的承诺 (promises)，包括合约参与方可以在上面执行这些承诺的协议。在物理上，智能合约的载体是计算机可识别并运行的计算机代码。比特币脚本语言是一种命令式的、基于栈的编程语言，由于它是非图灵完备的，所以应用上有一定的局限性。以太坊是全世界第一个实现图灵完备的智能合约的区块链系统，编程语言是 Solidity、Serpent，使得应用开发者们可以高效快速地开发各式各样的应用程序。智能合约代码发布到区块链上之后，无需中介的参与，在区块链上自动执行。

星云链中的智能合约编程语言，在初期时完全兼容以太坊的 Solidity，方便开发者为以太坊开发的智能合约应用无缝的迁移到星云链中来。我们在 Solidity 语言中增加一些跟 Nebulas Rank 相关的指令集，方便开发者获取任意用户的 NR 值。后续我们基于 NVM 推出各种编程语言的支持，使得开发者可以用自己喜欢的高级语言编程，例如 Java、Python、Go、JavaScript、Scala 等，甚至定制的应用在特定领域的高级语言。

3.3.2 合约可升级

目前以太坊智能合约的设计是代码一经部署，不可变化，代码逻辑从部署的时刻起，便永远不再具有升级的能力。智能合约如果作为协议来看，不可变化是其要求的，代表着一种协议的约定，运行行为都是确定性的。但是随着智能合约开始获得越来越多的使用，其流程和代码也变得越来越复杂，人们发现，就像现实世界的合同一样，如果没有认真审核的话，在设计和编码过程中难以避免人工失误的产生，一旦被黑客找到漏洞，损失往往是巨大的。2016 年 6 月，The DAO 攻击事件，由于一个代码缺陷，导致以太坊用户损失了共计 6000 万美元的损失；最近 Parity 钱包的漏洞，导致 15 万个以太币的流失，价值 3000 万美元。比特币由于其设计上的非图灵完备性，删减了许多脚本指令，所以其安全性是极高的。

虽然目前有各种智能合约编程的最佳实践，以及更严格的审核流程，甚至出现形式化验证工具，通过数学证明的方式验证智能合约的确定性。但是既然是代码，就不可能没有漏洞。回顾我们现在的中心化的互联网世界，各种互联网服务都是可以升级的，弥补开发过程中发生的各种漏洞。任何一个完美的应用系统，都是演化而不是设计出来的。我们认为，解决智能合约安全性的根本问题，需要有一个好的智能合约可升级设计方案。

以太坊上的智能合约可升级设计有一些解决方案，大体上分为两类：一类是对外公开代理合约 (Proxy Contract)，代理合约的代码非常简单，仅仅把请求转发给后面的真正的功能合约。当需要升级合约时，把代理合约的内部功能合约指针指向新的合约即可；第二类是把合约的代码和存储分离，存储合约负责提供方法，供外部合约读写内部状态，代码合约做真正的业务逻辑，升级时只需要部署新的代码合约，不丢失所有的状态。这两类方案都有其局限性，不能解决所有问题：合约的代码和存储分离在设计上增加了很多复杂度，有时候甚至不可行；代理合约虽然能够指向新合约，但是老合约的状态数据并不能迁移；有些合约在开始开发时，没有良好的设计，没有为以后的升级留下接口。

我们设计一种简洁的智能合约升级方案：在语言层面上，我们支持一个合约的状态变量供另外一个合约直接读写（符合安全约束）。这是一个例子，假如有个 Token 合约，代码如表10所示。

合约部署时，balances 变量用关键字 shared 标识，编译成字节码运行时，虚拟机会为该变量单独设计存储区域。不用关键字 shared 声明的变量，都不可以被其它合约直接访问。

假如原代码的 transfer 函数需要修改一个 bug，对 _value 做检查，部署新的智能合约代码，如表11所示。

新的合约部署以后，老的合约可以选择 selfdestruct，不能再被访问，但是 shared 变量依然被永久保留。新的合约可以完全继承老合约的 balances 资产，全部的状态都不丢失，不需要做额外的迁移工作。但是在开发智能合约时，对关键的状态变量声明为 shared 是必须的，编译器会对变量的存储区域做特殊处理，保证其可以被其它授权的合约访问。

为了保证安全，升级合约和老合约必须是相同的 creator，否则运行时会抛异常。

这种设计存在道德上的问题，因为合约的内容条款一旦拟订，其实是不应该被修改的，至少修改必须征询合约受众的同意。我们计划引入投票机制，以批准智能合约的升级，而不是默默的被合约创建者修改。

通过这种可升级方案，The DAO 或者 Parity 类似的漏洞攻击事件，可以更快的被修复，而不是通过硬分叉的方式。并且修复以后，所有用户的资产都不需要迁移，仍然继续使用。

```
contract Token {
  mapping (address => uint256) balances shared;

  function transfer(address _to, uint256 _value) returns (bool success) {
     if (balances[msg.sender] >= _value) {
       balances[msg.sender] -= _value;
       balances[_to] += _value;
       return true;
     } else {
       return false;
     }
   }
   function balanceOf(address _owner) constant returns (uint256 balance) {
       return balances[_owner];
   }
}
```

图 10: 原合约代码

```
[baseContractAddress="0x5d65d971895edc438f465c17db6992698a52318d"]
//baseContractAddress是老合约的地址
contract Token {
  mapping (address => uint256) balances shared;

  function transfer(address _to, uint256 _value) returns (bool success) {
     if (balances[msg.sender] >= _value && _value > 0) {
       balances[msg.sender] -= _value;
       balances[_to] += _value;
       return true;
     } else {
       return false;
     }
   }
   function balanceOf(address _owner) constant returns (uint256 balance) {
       return balances[_owner];
   }
}
```

图 11: 新合约代码

4 DIP 开发者激励协议

4.1 设计目标

为了更好地建立区块链应用的生态环境，在星云链中，我们提出面向智能合约开发者的 DIP(Developer Incentive Protocol，开发者激励协议)，通过星云币的奖励来感谢为生态助力的优秀智能合约开发者。

4.2 DIP 奖励分配算法

我们认为，一个智能合约是否优秀取决于有多少用户愿意使用它，而且有更多的高价值账户使用的智能合约更加优秀，而作为账户普适价值尺度的 NR 正好可以应用在高价值账户的评估中。DIP 的设计结合 NR 和常用的周活跃用户的概念，使用周活跃用户的价值尺度总和来衡量智能合约的价值尺度，然后使用该价值尺度来评估开发者的贡献度。

DIP 按周期进行一次，和 Nebulas Rank 计算周期一致。对于智能合约 C，假设本周活跃账户地址集合为 WAA(Weekly Active Addresses)，其中根据 §2.3的 NR 排名（取 Top X，值为 1-X），计算周活跃地址的 NR 之和作为合约 C 的贡献度 SCS(Smart Contract Score)，如公式9。

$$SCS(C) = \sum_{addr \in WAA} (max\{X + 1 - NR(addr), 0\}) \tag{9}$$

按照每周贡献值 SCS 从高到低排序得到智能合约贡献度排名 SCR(Smart Contract Rank)，取 Top N 的智能合约，它们对应的开发者将按比例瓜分 M 个星云币作为奖励，为了避免恶意刷榜，DIP 的分配曲线被设计得较为平均，如图12所示，但依旧保证 Rank 1 的收益为 Rank N 收益的一倍以示贡献度大小的区别，比例约束见公式10。

$$\begin{aligned} Coin(C) = \quad & kln(N + 1 - SCR(C)) + b \\ \text{s.t.} \quad & kln(N) + b = 2b \\ & \sum_{x=1}^{N} (kln(x) + b) = M \end{aligned} \tag{10}$$

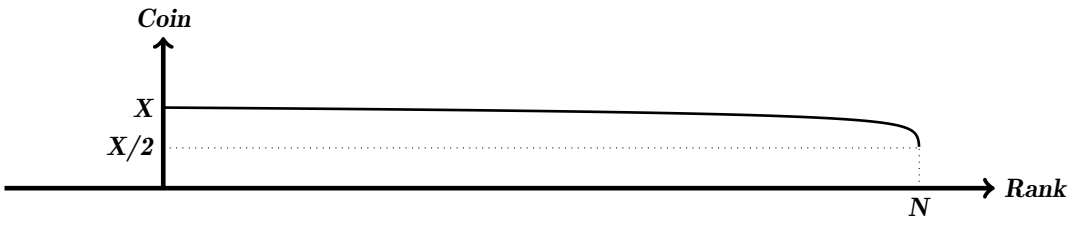


图 12: DIP 奖励分配曲线

DIP 的奖励将会由各个节点单独计算发放，假设星云链平均每 S 秒出一个区块，那么每隔 24*7*3600/S 个区块，所有节点将会计算一次 DIP 的奖励，并且发给对应的智能合约的提币地址中。

为了鼓励星云链生态智能合约的多样性，让更多新生开发者的优秀成果也能获得激励，DIP 规定每个智能合约最多可以接受 K 次奖励。DIP 将会根据排名每次选出还可以接受奖励的 Top N 智能合约给予激励，助推区块链应用生态发展。

4.3 实验结果

我们收集了以太坊上从 2017 年 5 月份交易数据，计算了第一周的 DIP 排名，如表2所示。

表 2: 2017 年 5 月第一周[1]DIP 排名结果 Top 10

Contract Address	Score	Description[2]
0xa74476443119a942de498590fe1f2454d7d4ac0d	264456363.0	GolemToken
0x49edf201c1e139282643d5e7c6fb0c7219ad1db7	207900181.0	TokenCard-ICO
0x48c80f1f4d53d5951e5d5438b54cba84f29f32a5	129625776.0	REP-Augur-OLD
0x6810e776880c02933d47db1b9fc05908e5386b96	108324489.0	Gnosis-TokenContract
0x6090a6e47849629b7245dfa1ca21d94cd15878ef	54429341.0	ENS-Registrar
0x607f4c5bb672230e8672085532f7e901544a7375	48526808.0	RLC
0x8d12a197cb00d4747a1fe03395095ce2a5cc6819	46498412.0	etherdelta_2
0xf7b098298f7c69fc14610bf71d5e02c60792894c	43746158.0	GUPToken
0xaaaf91d9b90df800df4f55c205fd6989c977e73a	42026627.0	TokenCardContract
0xaec2e87e0a235266d9c5adc9deb4b2e29b54d009	41427314.0	singularDTVToken

[1] 区块范围 [3629091, 3665815]

[2] 来自 etherscan.io

可见，DIP 排名靠前的合约较为知名，在计算周期内也更为活跃，符合我们激励生态建设者的初衷。

4.4 作弊分析

智能合约只能被动调用，不能主动和非合约账户建立交易，所以一个作弊者想要让他的智能合约排名上升，就必须要找到足够多的高 NR 排名账户来调用他的合约。

首先，作弊者想要零成本提升 DIP 排名将不可能实现。假设作弊者想要提高合约 C 的排名，并为此伪造了大量账户，但由于在公式9中计算 SCS 时，只有 NR 排名中 Top X 的调用贡献分大于 0，新伪造出来的账户 NR 排名将会在 Top X 之外，即使调用合约 C 也对 DIP 的排名没有任何影响。

其次，如果作弊者愿意为合约的 DIP 排名上升付出一定的成本，那么他有两种方式可以选择。第一种，自己花钱伪造高 NR 的账户来调用合约 C，提升合约 C 的排名。而要伪造高 NR 的账户在 §2.4已有分析，每提升一个账户都需要投入一大笔资金来伪造该账户特殊的拓扑结构，并且由于 NR 的周期性更新，长期

维持高 NR 的成本将会巨大。第二种，作弊者找到大量高 NR 账户，说服或者贿赂他们来调用合约 C，这种链下行为难以规模化，作弊者花了很大精力找到的高 NR 账户也只会占 Top X 中的一小部分，对真正优秀的合约影响不大。

5 PoD 贡献度证明共识算法

5.1 设计目标

共识算法作为区块链的基石之一，快速和不可逆是我们重点关注的目标。除此之外，为了更好地建设公链生态，我们认为公平性同样重要，如果大资本可以轻松占据公链中区块共识的话语权，那么会有很多公链上的开发者和用户的利益无端受损，一个不能保障公链建设者利益的生态，很难沉淀出价值深度，和星云链设计原则相违背。所以我们在设计共识算法时，在优先保证快速和不可逆的情况下，将尽可能追求公平性，维护公链建设者的利益。

5.2 常用共识算法的缺陷

我们试图在较为常用的共识算法中找到符合我们设计目标的选择，但是这些算法和我们的目标多少都有些差距。

PoW (Proof of Work) 工作量证明共识算法为零和博弈，采用竞争性哈希计算来确定记账人，导致了整个生态每次出块时都有大量电能在竞争中被无端消耗，挖矿成本高，而且速度受限。如果把公链参与者作为整体来看，随着参与挖矿的节点增加，每个节点获得记账权的概率将会减小，那么 PoW 协议下生态维持平稳出块的成本将会持续升高。不断增加挖矿难度的 Bitcoin 早晚需要面临矿机收益入不敷出的情形，而 Ethereum 则早已在考虑使用新的 PoS 共识算法 Casper [56] 来逐步取代现阶段的 PoW 共识 [11]。可见，从挖矿速度和经济成本角度，PoW 都不利于公链生态的长期快速发展，和我们“快速”的目标不相符。

PoS (Proof of Stake) 股权证明共识算法试图采用资产的多寡来取代算力的作用，按照币龄或者押金数额来分配获得记账权的概率，现阶段 Peercoin [29] 和 Ethereum 的 Casper 协议都采用了 PoS 共识算法。这种算法解决了 PoW 高能耗的弊端，但很直观地放大了资本对记账权概率分配的影响，相较于 PoW，在 PoS 下大资本更容易占据生态的话语权，形成大集团垄断，可能会对生态的建设者的利益造成损害，不利于公链生态的价值沉淀，同样和我们“公平性”的目标不相符。

PoI (Proof of Importance) 重要度证明共识算法最早由 Nem 提出 [39]，不同于 PoS，PoI 中引入了账户重要程度的概念，使用账户重要性评分来分配记账权的概率。这种算法解决了 PoW 的高能耗弊端，减缓了 PoS 的资本垄断危机，但暴露了 nothing-at-stake 的问题，作弊者逆转一个区块的成本被大大降低，和我们“不可逆”的目标不相符。

综上，鉴于常用共识算法和我们目标存在差距，我们提出了基于账户贡献度的 PoD (Proof of Devotion) 算法，将评估账户综合影响力的 PoI 和具有严格经济惩罚的 PoS 相融合，利用 PoS 强化 PoI 的不可逆性，使用 PoI 反向遏制了 PoS 的垄断性，以此为生态自由快速发展助力。

5.3 PoD 算法设计

5.3.1 新区块产生

类似 PoI 共识算法选取重要性高的账户，PoD 将选取生态中贡献度较高的账户，不同之处在于，PoD 赋予选取出来的账户平等概率的记账权来参与产生新区块 (block)，防止概率倾斜衍生垄断。

在选择贡献度较高的账户时，我们使用了星云链原生的 NR 普适价值尺度评估。在 NR 的算法设计中，着重考虑了账户的流动性和传播性（见 §2.1），我们认为满足这些性质的账户对生态建设贡献度较高。所以在 PoD 中，将选取 NR 排名 Top N 的账户，这些账户自愿缴纳一定数量的 Nas 作为押金后则有资格成为新区块的验证者 (validator)，参与记账。

在给定验证者集合 (validators set) 之后，PoD 算法通过伪随机数来决定验证者集合中谁是新的区块的提议者 (proposer)，提议者产生新区块。验证者集合不是固定不变的，有资格的账户可以选择加入或者退出验证者集合，而随着周期性 NR 的变化，有资格的账户也会不一样。所以我们在 PoD 设计了验证者集合动态变化机制，来实现验证者集合的更迭。

5.3.2 验证者集合更迭

验证者集合的更迭就如朝代变更一样，于是我们将验证者集合按照朝代 (dynasty) 做划分，一个朝代内验证者集合不会发生变化。一个朝代不能更迭地过快，至少要保持一段时间不做变更，因此我们将每 X 个区块定义为一个 Epoch，在同一个 Epoch 中朝代不会发生变化。所以朝代的变更只会发生在 Epoch 交接时，在此时将会考察上一个 Epoch 的第一个区块，如果此区块到达了 finality 状态，那么当前 Epoch 进入下一个朝代 D1，否则延续上一个朝代 D0 不变，如图13所示。

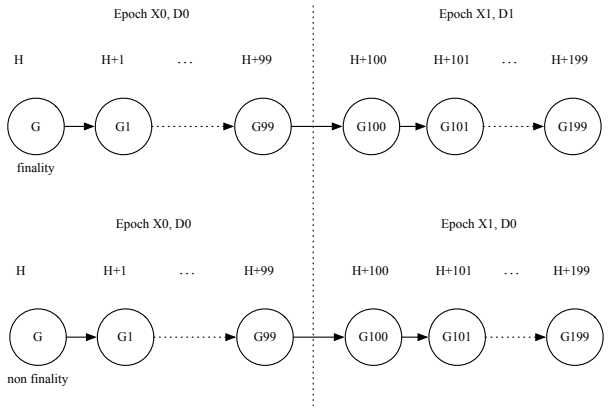


图 13: 验证者朝代更迭（假设 X=100）

由于网络延迟，各个节点可能在朝代更迭时，看到的区块 G 是否 finality 的状态不一致，所以参考 Casper 的动态验证集策略，要求每一个朝代的共识过程将由当前朝代和上一个朝代的验证者集合共同完成。因此在任意一个朝代，有资格的账户只能申请加入或者退出 D+2 朝代的验证者集合，当朝代变更到 D+2 时，才可加入新区块的共识过程。

5.3.3 共识过程

新的区块被提出后，当前朝代验证者集合中所有人将会参与一轮 BFT (Byzantine Fault Tolerant) 方式的投票，来确定此区块的合法性。在投票最开始，每一个参与此区块共识的验证者将会被从押金中收取 $2x$ (x 为激励奖金比例) 的保证金，然后进入两阶段的投票过程。

- **第一阶段**，所有验证者需要对新区块投 $Prepare$ 票，投完 $Prepare$ 票的验证者将获得 $1.5x$ 的奖励，如果在当前朝代和上一个朝代中都有超过 2/3 的押金总额的验证者对新区块投了 $Prepare$ 票，那么该区块进入投票的第二阶段。此处需要说明，新区块的提议者将被默认对新区块投 $Prepare$ 票。
- **第二阶段**，所有验证者需要对新区块投 $Commit$ 票，投完 $Commit$ 票的验证者，可以再获得 $1.5x$ 的奖励，如果在当前朝代和上一个朝代中都有超过 2/3 的押金总额的验证者对新区块投了 $Commit$ 票，那么该区块到达 finality 状态。

为了加速整个生态向前延展，如果区块 b 的 $Prepare$ 和 $Commit$ 票的时间戳和区块 b 的时间戳相差超过 T，那么这些票将被视为过期，直接忽略。

5.3.4 分叉选择

PoD 算法以每个高度上区块的得分来选择权威链，总是选择得分最高的区块加入权威链，在高度 h 的区块 b 的得分如下，

$$Score(b,h) = \sum_{(b',h') \in children(b)} Score(b',h') + \sum committed\ deposit\ in\ b \tag{11}$$

即为该区块及其所有后代区块收到的 commit 票对应的押金总和，如图14所示。

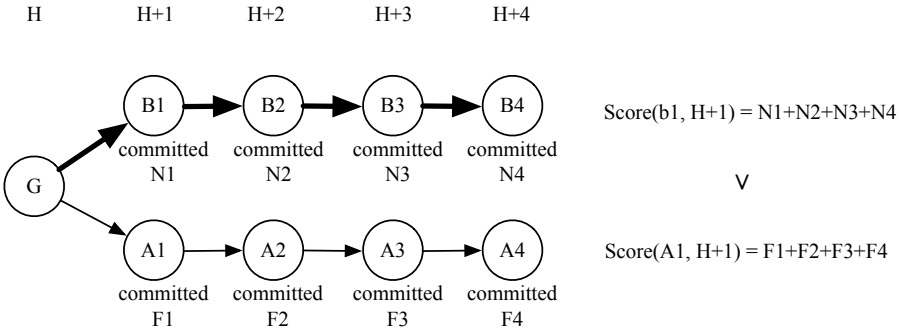


图 14: 分叉选择

5.3.5 投票规则

为了避免共识过程被恶意破坏，导致共识过程没法完成，阻碍生态发展，PoD 参考 Casper 的最小惩罚规则 [36] 来约束验证者的共识活动。

假设共识过程中的 $Prepare$ 和 $Commit$ 票结构如下，

- $Prepare(H, v, vs)$，其中 H 为当前区块 hash，v 表示当前区块高度，vs 表示 v 的某个祖先区块高度
- $Commit(H, v)$，其中 H 为当前区块 hash，v 表示当前区块高度

PoD 算法为整个投票过程制定了如下 4 条基本规则，

- 单个区块的两阶段共识过程存在严格的先后顺序，只有在第一阶段 $Prepare(H, v, vs)$ 票总权值达到 2/3 后，验证者们才可以投出第二阶段的 $Commit(H, v)$ 票，
- 多区块间不强制一个区块共识结束后才能开始后一个区块的共识，允许交织共识 (interwoven consensus)，但是不能完全没有秩序，只有高度 vs 完成了第一阶段过程，拥有 2/3 的 $Prepare(H_anc, vs, vs')$ 后，才可以基于 vs 对其后代区块投 $Prepare(H, v, vs)$ 票，保证交织稳步向前
- 为了避免有节点利用交织共识恶意跨多区块投票，要求基于高度 u 投出了 $Prepare(H, w, u)$ 票之后，对于高度在跨度 u 和 w 之间的所有区块，不能再投出 $Commit(H, v)$ 票，保证共识过程的高效有序
- 为了制止节点用同一笔押金在多个分支上同时下注，导致 nothing at stake 的问题，要求在一个高度投出 $Prepare(H1, v, vs1)$ 票之后，不能再投出不一样的 $Prepare(H2, v, vs2)$ 票

违反上述规则的验证者一旦被举报核实，将会被罚掉所有押金，举报者们将会共享罚金的 4% 作为奖励，罚金的剩余部分将会被销毁。

5.4 PoD 经济分析

5.4.1 激励分析

参与 PoD 算法的验证者，在每一个合法区块上可以获得 1x 的星云币奖励，如果网络不畅或者有人作弊导致 Prepare 阶段没有办法完成进入 Commit 阶段，那么所有验证者将损失 0.5x。因此成为验证者的价值节点在保持网络畅通，不参与作弊的情况下，将共享大量记账收益。

5.4.2 作弊分析

双重支付攻击 (double spend)

假设商户 merchant 等到新区块到达 finality 状态就确认交易发货，那么 fraud 要在 PoD 共识算法下完成双重支付攻击实现零成本购物要付出的最小代价如下：

首先，fraud 需要提高自己的 Nebulas Rank 到 Top N，然后缴一定数的 NaS 作为押金成为验证者，并申请参与 D+2 朝代区块的验证。

然后，fraud 需要被伪随机算法选中为新区块的提议者，此时 fraud 提出两个高度相同的新区块，一个哈希值为 hash1 包含 fraud 向 merchant 的转账交易，另一个哈希值为 hash2 包含 fraud 向 fraud 自己的转账交易。

最后，为了让 hash1 和 hash2 区块都到达 finality，如图15所示，fraud 至少需要花费所有押金的 1/3 来贿赂 1/3 的验证者，让他们给两个区块都投 $Commit$ 票。

所以要完成一次成功的双重支付攻击，fraud 需要花费一定的精力和财力来提升自己的 Nebulas Rank 排名（见 §2.4抵抗操纵），然后等到幸运地被选为提议者时，至少花费总押金的 1/3 来让两个块同时到达 finality 状态。

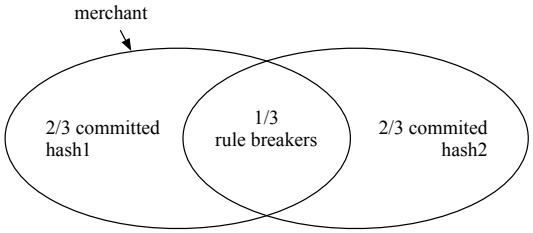


图 15: 双重支付经济惩罚

51% 攻击 (51% attack)

在 PoW 中要发起 51% 攻击需要 51% 的算力，在 PoS 中则需要 51% 的押金，而在 PoD 中，则需要验证者集合中 51% 的账户，这意味着拥有足够多的高声望用户进入 Nebulas Rank 的 Top N，并且需要支付对应的押金，因此在 PoD 中 51% 攻击将更为困难。

短程攻击 (short-range attack)

PoD 中的每个高度上的区块都有共识有效期，如果某个高度距离最新高度超过 100 时，该高度的所有区块在共识过程中将被视为过期，那么这些区块上的所有新的共识活动将会被直接忽略。因此要在 PoD 中完成长程攻击 (long-range attack) 几乎不可能，但是在有效期内依旧存在发起短程攻击的可能性。

短程攻击者 Attacker 试图在高度 H+1 的区块还没有过期的情况下，伪造 A 链来替代 B 链成为权威链，Attacker 需要让区块 A1 的得分比 B1 更高。由于多投会被严惩，所以 Attacker 将不可避免地要贿赂验证者，否则无法完成短程攻击。为了展现 PoD 共识算法的安全性，下面分别分析使不同数量的区块失效时，Attacker 需要付出的代价。

如果 Attacker 想要使 B1 失效，最小代价的情况如图16，就相当一次双重支付攻击，Attacker 幸运地成为了 H+1 高度的区块提议者，那么至少需要贿赂朝代 D0 中 1/3 的验证者多投使 A1 达到 finality，最小代价为所有押金的 1/3。

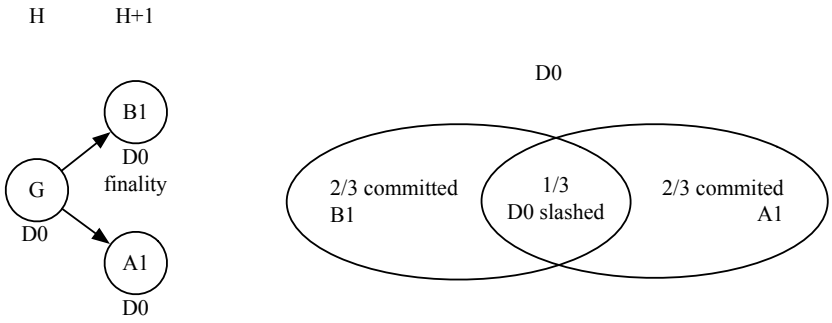


图 16: 短程攻击使一个区块失效的情形

如果 Attacker 想要使 B1-B2 失效，假设 B1 和 B2 都已到达 finality，块中交易都已生效，为了让这些交易失效，这里考虑两种情况。第一种如图17中 (a) 所示，高度 H+1 和 H+2 在同一个 Epoch 中，朝代相

同，那么 Attacker 首先需要贿赂 D0 中 1/3 的验证者使 A1 达到 finality，此时这 1/3 的验证者将会被惩罚，押金被罚完。在 A2 的验证中整体押金总和只有 A1 中的 2/3，此时 Attacker 想要让 A2 到达和 B2 同价值的 committ 票，需要贿赂剩下所有没有作弊的验证者，合起来至少需要损失总押金的 3/3，即使如此也不能保证 A1 得分比 B1 高，攻击失败风险高。第二种情况如图17中 (b) 所示，高度 H+1 和 H+2 正好在不同的 Epoch 中，且朝代不相同，那么此时 Attacker 需要贿赂 D0 中的 1/3 来让 A1 到达 finality，然后贿赂 D1 中的 1/3 来让 A2 达到 finality，完成一次这样的攻击至少需要损失总押金的 2/3。综上，想要发起短程攻击导致两个 finality 区块失效，至少需要花费总押金 2/3 的代价。

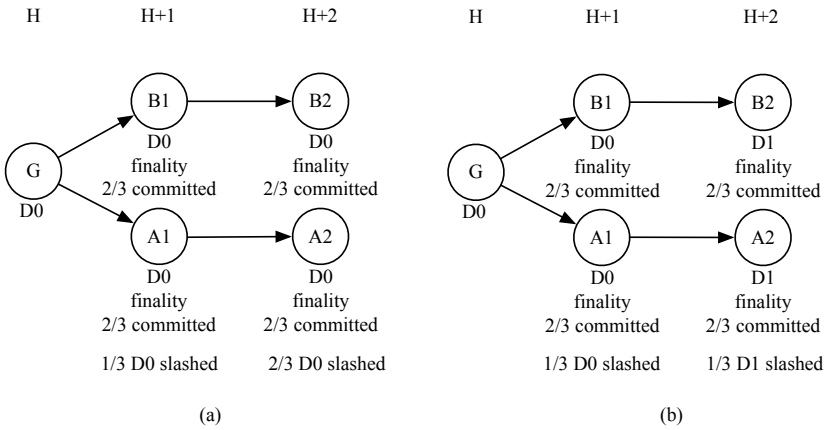


图 17: 短程攻击使两个区块失效的情形

如果 Attacker 想要使 B1-B3 失效，如图18所示，Attacker 首先需要贿赂 D0 中 1/3 的人完成 A1 的 finality，然后贿赂 D1 中 1/3 的人完成 A2 的 finality，最后需要贿赂 D1 中剩下 2/3 中的所有人来完成 A3 的 finality，综上至少要损失总押金的 4/3。要完成这些攻击准备将会十分困难，而且即使有幸做到了，也不定能保证 A1 的得分比 B1 高，攻击也可能会失败。

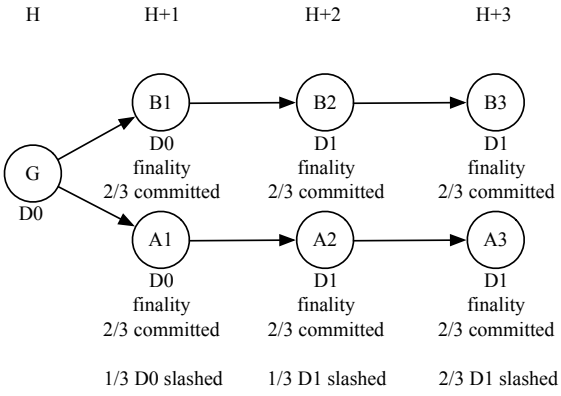


图 18: 短程攻击使三个区块失效的情形

如果 Attacker 想要使 B1-BN 失效，其中 N 受到区块共识有效期的限制，不会很大，由于 $N = 3$ 时

当前朝代所有验证者的押金就会被全部罚完，所以 $N >= 4$ 时，将没法完成攻击让 B1 得分比 A1 高，使 B1-BN 失效，发起这样的攻击没有任何意义。

6 区块链搜索引擎

6.1 简介

随着越来越多的智能合约被开发者部署，用户面对海量的智能合约，搜索的需求也变得突出起来。智能合约仅仅是代码，并不包含任何功能描述，很难利用搜索引擎技术为智能合约建立索引。我们将用多种手段为智能合约建立合适的索引：

- 爬取智能合约相关网页资料，为其和区块链智能合约建立映射关系；
- 鼓励开发者上传验证过的智能合约源代码，分析其代码功能和语义，为源代码建立索引，提供相似合约搜索；未提供源代码的智能合约，进行反编译得到其源代码；
- 为智能合约建立规范，任何符合此规范的智能合约，都能被检索并被用户搜索到，鼓励开发者在创建智能合约时，提供合约信息描述。

```
contract SearchableContract {
    string public language;
    string public author;
    string public name;
    string public title;
    string public description;
    string public tags;
}
```

6.2 搜索基础架构

我们认为，搜索服务为了提供最佳的用户体验，中心化的架构更为适合。星云链开发组将实现一个搜索服务，实时检索所有智能合约，做多语言的分词，建立全文索引，最终提供一个友好的 Web 界面给用户使用。**NR 排名算法的公正性和每个节点的可验证性保证了中心化搜索服务的公正性**，所有搜索后台代码都将开源给社区。第三方开发者也可据此创建独立的搜索服务。搜索服务架构如图19所示。

- **Crawler** 区块链搜索引擎爬虫数据源分为两种，一种从区块链上收集区块信息和智能合约代码等信息，一种从公开网址上爬取智能合约相关资料，包括介绍、Dapp 用户评论、新闻资讯等。
- **Extractor** 包括 Text Extractor, Block Extractor 和 Code Extractor，分别提供文本信息、区块信息和智能合约代码提取服务。
- **Analyzer** 包括 Text Analyzer, Tx Analyzer, Contract Analyzer，分别为文本信息、区块交易信息和智能合约分析器，其中智能合约分析器包括合约反编译、源代码功能、语义分析，网页解析等。

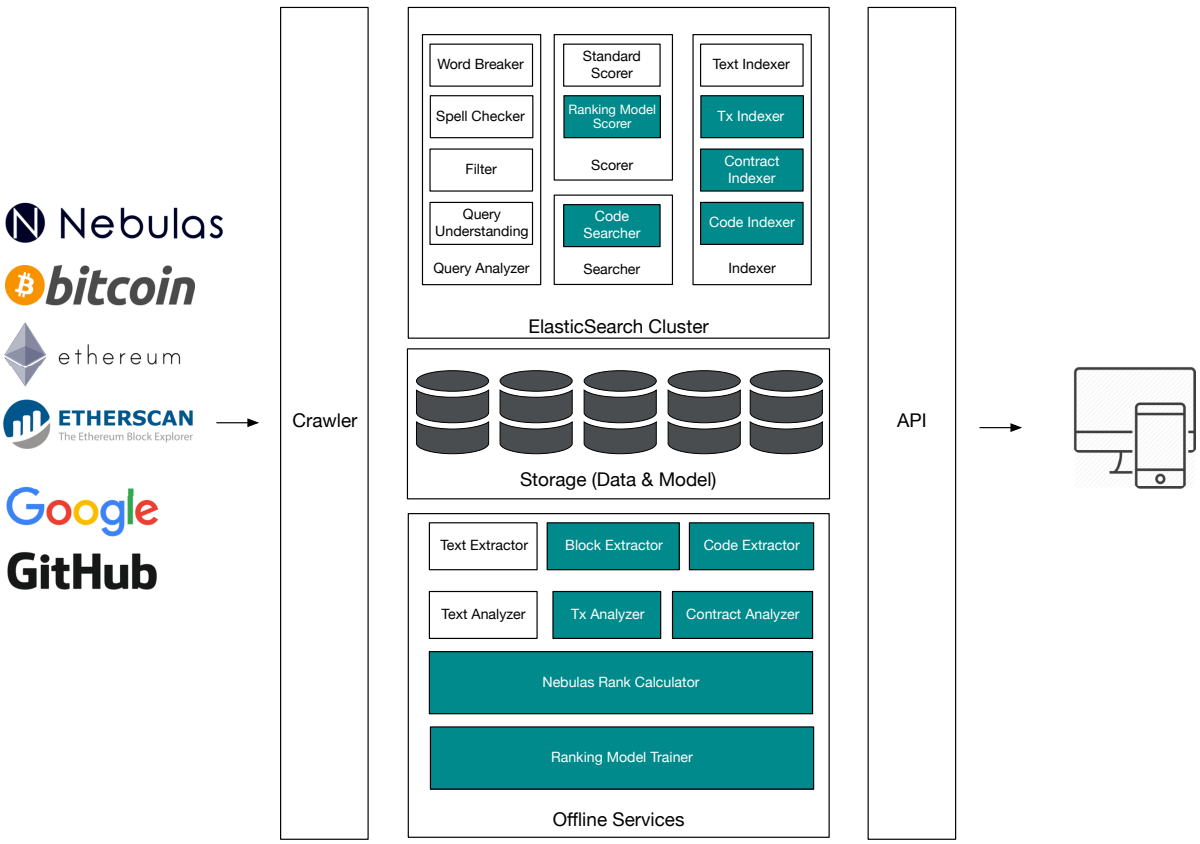


图 19: 搜索服务架构

- **Nebulas Rank Calculator** 星云指数计算服务，离线计算每个非合约账户和合约账户的星云指数。
- **Ranking Model Trainer** 排序模型训练。考虑到交易 Graph，排序规则考虑了多个因素：匹配字段，文本相关度，合约的 NR Rank 值，合约的交易数量、频度和深度，与合约发生交易的用户 NR Rank 值，合约安全性等等。我们根据用户的实际使用情况，用机器学习算法 (GBDT、人工神经网络都是候选) 训练排序打分模型，根据用户的反馈不断优化。训练后的模型被搜索服务的 Scorer 使用。
- **Query Analyzer** 关键词分析服务，包括多语言分词器 Word Breaker，拼写检查器 Spell Checker 等。
- **Indexer** 从 Analyzer 中建立合适的索引，支持全量和增量索引。
- **Scorer** 分为两个 level：Level-1 Standard Scorer 从 ElasticSearch 中召回候选结果集，目标是尽可能把潜在的候选结果，在 ElasticSearch 集群中通过快速有效的排序打分召回，Level-1 召回的结果数量有限，控制在几千以内；Level-2 Ranking Model Scorer 使用 Offline Rank Model，计算每个 Level-1 结果候选集的得分，重新排序，这个结果较为精确，可以直接给用户使用。
- **Searcher** 负责和 ElasticSearch 集群通信，以及对搜索结果包装返回给搜索前端。
- **API** 对外提供完善的搜索 API 服务。
- **ElasticSearch Cluster** ES 服务器集群。星云链团队考虑使用开源搜索引擎 ElasticSearch 作为其全文索引支持。

6.3 趋势榜单

星云链将结合星云指数构建趋势榜单，提供给用户可视化的区块链多维度价值。

- **非合约账户 Nebulas Rank 榜单** 呈现每日 NR 榜单，以及 NR 快速提升和快速下降榜单，并且可视化呈现各个账户的 R 变化趋势，以及整个网络健康程度变化趋势。
- **合约账户 Nebulas Rank 榜单** 根据非合约账户 NR 值，计算合约账户的 NR 榜单，以及快速提升和快速下降榜单，各个合约的变化趋势，以及整个网络智能合约数量和使用频次的趋势图。除此之外，我们还将呈现不同领域的智能合约榜单，比如 Token 合约榜单、预测市场合约榜单等等，以展示更多维度的信息。
- **智能合约开发者榜单** 根据合约账户榜单，计算合约开发者的贡献度榜单，以及贡献度快速提升榜单，展示优秀合约开发者及其应用。

6.4 关键词搜索

用户提供关键词，描述智能合约的标题、作者、功能等文本信息，在海量的智能合约中找到匹配的合约。对于文本搜索，目前有非常成熟的算法和技术，基于自然语言处理和倒排索引技术，我们可以在海量的智能合约索引库里高效的检索和排序。关键的技术如下：

1. 面向主题的分布式爬虫技术
2. 多语言分词技术：对于西文词汇，分词较为简单。对于中文分词，有多种算法可进行分词：正向最大匹配，逆向最大匹配，最短路径分词，统计分词等；
3. 查询词纠错、语义理解
4. 倒排索引，分布式搜索架构
5. 排序算法，给搜索结果排序

其中排序算法将结合 Nebulas Rank 来设计，我们把区块链世界中用户之间的转账类比互联网世界的网页引用关系，构建出区块链交易图，然后利用2.3中提出的 NR 排名算法，计算非合约用户的 NR 排名，然后结合4.2中介绍的合约排名算法，计算合约的排名结果，最后应用于搜索结果排序。

6.5 相似智能合约搜索

开发者和某些用户，可能有根据合约代码片段，搜索具有相似功能的智能合约的需求。不同于普通的关键词搜索，代码相似度有其特殊性。我们需要有一定的算法，通过数值或者百分比的形式对代码相似度进行度量，从而提供相似智能合约搜索的功能。

目前学术界对代码相似度算法主要有字符串编辑距离、Token 序列相似度、抽象语法树相似度和程序依赖图相似度四个流派，他们分别从不同维度描述了代码文本、结构和语法上的相似度，我们结合主流 4 个流派的思路，提出了星云链合约代码相似度算法的 12 种特征，如 Skeleton Tree、Type Signature 和 Libaray Calls 等，详情见附录B。

相似智能合约搜索结果，和关键词搜索结果一样，使用同样的合约排名算法排序，给出最终结果。

7 基础服务及开发工具

7.1 域名服务

在区块链的世界中，应用都以账号为中心，由于区块链的匿名性，账号地址是非常长的无意义的字符串，并不是用户友好的。用户可能容易出错，意外地投入资金导致丢失，或者与错误对象互动。如果用户能够使用容易记忆的域名，将是巨大的优势。星云链团队将会在链上用智能合约实现一个类似 DNS 的域名系统，称为 NNS(Nebulas Name Service)，保证其自由、免费、开放。任何第三方开发者可以独立地或者基于 NNS 实现自己的域名解析服务。

例如，alice 为她的账号地址 0xdf4d22611412132d3e9bd322f82e2940674ec1bc03b20e40 申请了域名 alice.nns，Bob 如果想转账给 alice，在收款人信息中只要填写 alice.nns 即可，钱包会通过 NNS 服务，把资金转账到正确的收款人地址。

NNS 的申请规则如下：

- 一级子域名保留，不可申请，例如 *.nns，*.com，*.org，*.edu 等等，用户只能申请二级子域名。
- NNS 服务开放后，用户可以查询域名是否已经被占用。如果未被占用，可以通过智能合约提出竞价。竞价是公开的，任何人都可以查询其他人的竞价并且随时更新自己的竞价。
- 竞价期结束以后，价高者得此域名，智能合约会锁定用户的竞价资金。域名有效期为一年。一年后用户自由决定是否续约，同意续约后，有效期再延长一年。如果不续约，竞价资金自动退回用户账户，域名数据清空，重新变为可用。
- 用户可以随时放弃域名所有权，竞价资金自动退回用户账户，域名数据清空，重新变为可用。
- 用户可以无偿或者有偿转让域名所有权，域名之间的交易，星云链不做任何干预。

7.2 闪电网络

当前所有的公有链网络都面临系统的扩展性问题，例如比特币网络每秒只能处理 7 个交易，以太坊每秒处理 15 个交易。引入类 PoS 的共识算法，可以避免挖矿计算，大大加快共识速度，但是面对现实世界的海量微支付场景，公有链仍然面临极大的挑战。闪电网络 [48] 于 2015 年 2 月被提出，设计思路是为交易双方建立一个微支付通道网络，双方大量的支付都可以在链外直接多次、高频、双向地通过轧差方式实现瞬间确认。当交易结果需要结算时，再把最终结果提交到区块链确认，理论上可以实现每秒百万笔的转账。双方若无直接的点对点支付通道，只要网络中存在一条连通双方的、由多个支付通道构成的支付路径，也可以利用这条支付路径实现资金在双方之间的可靠转移。闪电网络在比特币和以太坊上都已经有概念性的验证。

星云链把闪电网络当做区块链的基础设施予以实现，并且提供足够的灵活性设计。任何第三方开发者，都可以在星云链上利用闪电网络的基础服务，做高频交易场景的应用开发。星云链也将发布世界首款支持闪电网络的钱包 App。

7.3 开发者工具

完善的开发者工具，对于区块链应用开发者来说是非常重要的。目前各种公有区块链的开发者工具链都不够完善，对于大多数开发者是很大的阻碍。星云链开发组将提供丰富的开发者工具，包括独立的智能合约开发 IDE，区块浏览器，各种流行 IDE 的插件支持 (包括 Eclipse, JetBrains, Visual Studio, Sublime Text, VIM, Atom 等)，调试器，模拟器，智能合约形式化验证工具，各种高级语言的后台 SDK，移动端 SDK 等。

8 星云链代币 NAS

星云链网络包含自身的内置代币星云币 (NAS)。星云币在网络中体现两个作用：第一，星云币作为网络中的原生代币，提供用户之间的资产流动性，也作为 PoD 记账人和 DIP 开发者激励计划的奖励代币；第二，作为运行智能合约的运算费用收取。星云币的最小单位是 $10^{-18}NAS$。

星云币最初在以太坊平台上以 ERC20 代币的方式发售，代币总量最大值为 $X = 10^8$ 枚，发行模式如下：

1. **社区建设：**

   在星云链发起团队主导下，将会有 $80\%X$ 的代币用于星云社区建设，包括星云社社区区块链应用（DApp）生态孵化和激励、开发者社区建设、商业合作和产业合作、市场营销推广、学术研究、教育投资、法律法规、社区及机构投资。其中面向社区影响力投资人售卖 $5\%X$，还有 $5\%X$ 作为星云社区发展基金，剩下 $70\%X$ 待做预留。

2. **创始 & 开发团队激励：**

   星云链的创始 & 开发团队将在星云链的发展过程中，从项目组织架构，技术研发，生态运营上持续做出人力、物力资源的贡献。在代币分配机制上，预留出 $20\%X$ 的部分作为团队激励。这部分星云币初始为锁定状态，自星云币首次面向社区售卖完成一年后开始解除锁定，分三年逐步分发至创始 & 开发团队。

星云链网络正式上线后，所有持有以太坊 ERC20 NAS 币的用户可以根据凭证，领取星云链网络的等量的星云币，同时以太坊的 ERC20 NAS 币被回收。随着星云链网络的演化，星云币增长如下：

1. **记账人激励：** 每年奖励给记账人的星云币数量为 $3\%X$；
2. **DIP 激励：** 每年奖励给优秀智能合约开发者的星云币数量为 $1\%X$。

9 总结

我们认为

区块链从一种高度抽象的角度来看是一种**用去中心化的方式对于数据的确权**，代币本身是对于**确权价值的载体**。互联网解决了数据的通讯问题，而区块链则在互联网上层进一步解决了数据的确权问题。区块链前所未有第一次让大家的数据真正变成自己的数据，而不再被 BAT 等大公司任意分析并使用。

以公有链为代表的区块链本质精神是：**社群 + 代币 + 工具**。社群本质上是自下而上，秉承的是开放、开源、共享、非盈利的理念，和现有自上而下的商业生态有着根本的不同。代币即是对于确权价值的载体，未来会有更多的应用场景，而远非是仅仅面向虚拟货币，电子现金的属性。工具仅仅是对于区块链应用场景具体技术实施，如缺少前两者的结合，其单独来看**并不能完全体现区块链系统的魅力所在**。

以公有链为代表的区块链体系才是区块链的未来，因为其本身所具有的**"非信任"、"无特权"**的基本特性才是区块链系统真正的价值所在。恰恰相反，作为联盟链/企业链大多具有**"基于信任"**和**"基于特权"**的属性，不能突破既有的范式，属于改良式创新。而公有链系统颠覆了既有的协作关系，属于**颠覆式创新**，是区块链价值最大化的真正体现。

我们致力于

作为全球首个区块链搜索引擎，星云链致力于**发掘区块链世界价值新维度**，打造基于价值尺度的区块链操作系统、搜索引擎及其他相关扩展应用。

基于此，我们提出 Nebulas Rank 星云指数来构建区块链世界的价值尺度，设计 Nebulas Force 星云原力来赋予区块链自我进化的能力，推出 Developer Incentive Protocol 开发者激励计划和 Proof of Devotion 贡献度共识证明来激励区块链的价值升级，打造 Nebulas Search Engine 区块链搜索引擎来帮助用户发现区块链上沉淀的多维度价值。

我们坚信

正在发生科技浪潮会带领人们抵达更为**自由、平等、和美好**的生活。区块链作为其中重要的技术之一，会愈来愈散发出其特有的光彩和能量。能参与并投身其中，是我们最大的快乐和成就。

类似互联网对于世界的改变，区块链也即将面对其用户/应用临界值爆发的阶段。区块链技术会是下一代**"智能网络"的基础协议**，整体用户规模会在 5-10 年内达到或超过 10 亿。未来的 5 年内会面临重大的机遇与挑战！

在未来的巨大生态面前，在当下，不要问区块链能为你做什么，而是要问你能为区块链做什么。因为，**区块链本身就是生命体，区块链本身就是经济体**！在区块链技术探索的道路上，与诸君共勉！

参考文献

[1] Luke Anderson *et al.* "New kids on the block: an analysis of modern blockchains." In: (2016). arXiv: 1606.06530. URL: http://arxiv.org/abs/1606.06530.

[2] Annika Baumann, Benjamin Fabian, and Matthias Lischke. "Exploring the Bitcoin network." In: *WEBIST 2014 - Proceedings of the 10th International Conference on Web Information Systems and Technologies* 1 (2014), pp. 369–374. ISSN: 9789897580239 (ISBN). DOI: 10.5220/0004937303690374. URL: https://www.engineeringvillage.com/blog/document.url?mid=cpx%7B%5C_%7D9ce5505146fd48dcbdM557010178163125%7B%5C&%7Ddatabase=cpx.

[3] Morten L Bech and Enghin Atalay. "The topology of the Federal Funds markets." In: *Economic Policy Review* 14.2 (2008).

[4] Phillip Bonacich. "Factoring and weighting approaches to status scores and clique identification." In: *Journal of Mathematical Sociology* 2.1 (1972), pp. 113–120.

[5] Stephen P. Borgatti. "Centrality and network flow." In: *Social Networks* 27.1 (2005), pp. 55–71. ISSN: 03788733. DOI: 10.1016/j.socnet.2004.11.008.

[6] Stephen P. Borgatti and Martin G. Everett. "A Graph-theoretic perspective on centrality." In: *Social Networks* 28.4 (2006), pp. 466–484. ISSN: 03788733. DOI: 10.1016/j.socnet.2005.11.005.

[7] Michael Boss, Martin Summer, and Stefan Thurner. "Contagion Flow Through Banking Networks." In: *Lecture Notes in Computer Science 3038* (2004), pp. 1070–1077. ISSN: 03029743. DOI: 10.1016/j.jfi.2008.06.003. arXiv: 0403167v1 [arXiv:cond-mat].

[8] Michael Boss *et al.* "The Network Topology of the Interbank Market." In: *Quantitative Finance* 4.6 (2004), pp. 677–684. ISSN: 1469-7688. DOI: 10.1080/14697680400020325. arXiv: 0309582 [cond-mat]. URL: http://arxiv.org/abs/cond-mat/0309582.

[9] Sergey Brin and Lawrence Page. "Reprint of: The anatomy of a large-scale hypertextual web search engine." In: *Computer Networks* 56.18 (2012), pp. 3825–3833. ISSN: 13891286. DOI: 10.1016/j.comnet.2012.10.007. arXiv: 1111.6189v1.

[10] Vitalik Buterin. "Ethereum: A next-generation smart contract and decentralized application platform." In: *URL https://github. com/ethereum/wiki/wiki/% 5BEnglish% 5D-White-Paper* (2014).

[11] Vitalik Buterin *et al. Ethereum white paper*. 2013.

[12] Lijun Chang *et al.* "pSCAN: Fast and Exact Structural Graph Clustering." In: *IEEE Transactions on Knowledge and Data Engineering* 29.2 (2017), pp. 387–401.

[13] Tao Hung Chang and Davor Svetinovic. "Data Analysis of Digital Currency Networks: Namecoin Case Study." In: *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems, ICECCS* (2017), pp. 122–125. DOI: 10.1109/ICECCS.2016.023.

[14] Duan Bing Chen *et al.* "Identifying influential nodes in large-scale directed networks: The role of clustering." In: *PLoS ONE* 8.10 (2013), pp. 1–10. ISSN: 19326203. DOI: 10.1371/journal.pone.0077455.

[15] Michel Chilowicz, Etienne Duris, and Gilles Roussel. "Syntax tree fingerprinting for source code similarity detection." In: *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*. IEEE. 2009, pp. 243–247.

[16] *Etherscan - The Ethereum Block Explorer*. https://etherscan.io/. Accessed: 2017-08-01.

[17] Giorgio Fagiolo. "The International-Trade Network: Gravity Equations and Topological Properties." In: (2009). arXiv: 0908.2086. URL: http://arxiv.org/abs/0908.2086.

[18] Jeanne Ferrante, Karl J Ottenstein, and Joe D Warren. "The program dependence graph and its use in optimization." In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 9.3 (1987), pp. 319–349.

[19] Danno Ferrin. "A Preliminary Field Guide for Bitcoin Transaction Patterns." In: *Texas Bitcoin Conference* (2015). URL: http://texasbitcoinconference.com.

[20] Michael Fleder, Michael S. Kester, and Sudeep Pillai. "Bitcoin Transaction Graph Analysis." In: *… -Transaction-Graph-Analysis. Pdf"…* (2015), pp. 1–8. arXiv: 1502.01657. URL: http://arxiv.org/abs/1502.0165%7B%5C%%7D5Cnhttp://people.csail.mit.edu/spillai/data/papers/bitcoin-project-paper.pdf%7B%5C%%7D5Cnhttp://arxiv.org/abs/1502.00165%7B%5C%%7D5Cnhttp://arxiv.org/abs/1502.01657.

[21] L Freeman. "A set of measures of centrality: I. Conceptual clarification." In: *Soc. Networks* 1 (1979), pp. 215–239.

[22] Linton C Freeman. "A set of measures of centrality based on betweenness." In: *Sociometry* (1977), pp. 35–41.

[23] Linton C Freeman. "Centrality in social networks conceptual clarification." In: *Social networks* 1.3 (1978), pp. 215–239.

[24] Linton C Freeman, Stephen P Borgatti, and Douglas R White. "Centrality in valued graphs: A measure of betweenness based on network flow." In: *Social networks* 13.2 (1991), pp. 141–154.

[25] Sudipto Guha *et al.* "Approximate XML joins." In: *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM. 2002, pp. 287–298.

[26] Bernhard Haslhofer, Roman Karl, and Erwin Filtz. "O Bitcoin Where Art Thou? Insight into Large-Scale Transaction Graphs." In: ().

[27] Vineeth Kashyap *et al.* "Source Forager: A Search Engine for Similar Source Code." In: *arXiv preprint arXiv:1706.02769* (2017).

[28] Leo Katz. "A new status index derived from sociometric analysis." In: *Psychometrika* 18.1 (1953), pp. 39–43.

[29] S King and S Nadal. "Peercoin–Secure & Sustainable Cryptocoin." In: *Aug-2012 [Online]. Available: https://peercoin.net/whitepaper* ().

[30] Jon M Kleinberg. "Authoritative sources in a hyperlinked environment." In: *Journal of the ACM (JACM)* 46.5 (1999), pp. 604–632.

[31] Lothar Krempel. "Exploring the Dynamics of International Trade by Combining the." In: December (2002), pp. 1–22.

[32] Qian Li *et al.* "Identifying influential spreaders by weighted LeaderRank." In: *Physica A: Statistical Mechanics and its Applications* 404 (2014), pp. 47–55. ISSN: 03784371. DOI: 10.1016/j.physa.2014.02.041. arXiv: arXiv:1306.5042v1.

[33] *llvm*. https://llvm.org/. Accessed: 2017-08-01.

[34] Linyuan Lü *et al.* "Vital nodes identification in complex networks." In: *Physics Reports* 650 (2016), pp. 1–63. ISSN: 03701573. DOI: 10.1016/j.physrep.2016.06.007. arXiv: 1607.01134.

[35] Sarah Meiklejohn *et al.* "A fistful of Bitcoins: Characterizing payments among men with no names." In: *Proceedings of the Internet Measurement Conference - IMC '13* 6 (2013), pp. 127–140. ISSN: 15577317. DOI: 10.1145/2504730.2504747. URL: http://dl.acm.org/citation.cfm?id=2504730.2504747.

[36] *Minimal Slashing Conditions*. https://medium.com/@VitalikButerin/minimal-slashing-conditions-20f0b500fc6c. Accessed: 2017-08-01.

[37] L Morten, J Robert, and E Walter. "The topology of interbank payment flows." In: (2006).

[38] Satoshi Nakamoto. "Bitcoin: A Peer-to-Peer Electronic Cash System." In: *Www.Bitcoin.Org* (2008), p. 9. ISSN: 09254560. DOI: 10.1007/s10838-008-9062-0. arXiv: 43543534534v343453. URL: https://bitcoin.org/bitcoin.pdf.

[39] *NEM Technical Reference.* http://nem.io/NEM_techRef.pdf. Accessed: 2017-08-01.

[40] Mark Newman. *Networks: an introduction.* Oxford university press, 2010.

[41] Mark EJ Newman. "A measure of betweenness centrality based on random walks." In: *Social networks* 27.1 (2005), pp. 39–54.

[42] Dá Niel Kondor *et al.* "Do the Rich Get Richer? An Empirical Analysis of the Bitcoin Transaction Network." In: *PLoS ONE* 9.2 (2014). DOI: 10.1371/journal.pone.0086197.

[43] Athanasios N. Nikolakopoulos and John D. Garofalakis. "NCDawareRank." In: *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13* February 2013 (2013), p. 143. DOI: 10.1145/2433396.2433415. URL: http://dl.acm.org/citation.cfm?doid=2433396.2433415.

[44] Jae Dong Noh and Heiko Rieger. "Random walks on complex networks." In: *Physical review letters* 92.11 (2004), p. 118701.

[45] Micha Ober, Stefan Katzenbeisser, and Kay Hamacher. "Structure and Anonymity of the Bitcoin Transaction Graph." In: *Future Internet* 5.2 (2013), pp. 237–250. ISSN: 1999-5903. DOI: 10.3390/fi5020237. URL: http://www.mdpi.com/1999-5903/5/2/237/.

[46] Lawrence Page *et al. The PageRank citation ranking: Bringing order to the web.* Tech. rep. Stanford InfoLab, 1999.

[47] Thai Pham and Steven Lee. "Anomaly detection in bitcoin network using unsupervised learning methods." In: *arXiv preprint arXiv:1611.03941* (2016).

[48] Joseph Poon and Thaddeus Dryja. "The bitcoin lightning network: Scalable off-chain instant payments." In: *Technical Report (draft)* (2015).

[49] Marc Pröpper, Iman van Lelyveld, and Ronald Heijmans. "Towards a network description of interbank payment flows." In: (2008).

[50] Dorit Ron and Adi Shamir. "Quantitative Analysis of the Full Bitcoin Transaction Graph." In: ().

[51] Gert Sabidussi. "The centrality index of a graph." In: *Psychometrika* 31.4 (1966), pp. 581–603.

[52] Computer Science and The Technion. "The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC E ect." In: (2001).

[53] M. Ángeles Serrano, Marián Boguñá, and Alessandro Vespignani. "Patterns of dominant flows in the world trade web." In: *Journal of Economic Interaction and Coordination* 2.2 (2007), pp. 111–124. ISSN: 1860711X. DOI: 10.1007/s11403-007-0026-y. arXiv: 0704.1225.

[54] Ma Angeles Serrano and Marián Boguñá. "Topology of the world trade web." In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 68.1 Pt 2 (2003), p. 015101. ISSN: 1063-651X. DOI: 10.1103/PhysRevE.68.015101. arXiv: 0301015 [cond-mat].

[55] Hiroaki Shiokawa, Yasuhiro Fujiwara, and Makoto Onizuka. "SCAN++: efficient algorithm for finding clusters, hubs and outliers on large-scale graphs." In: *Proceedings of the VLDB Endowment* 8.11 (2015), pp. 1178–1189.

[56] *The Stage 1 Casper Contract*. https://github.com/ethereum/casper/. Accessed: 2017-08-01.

[57] Florian Tschorsch and Björn Scheuermann. "Bitcoin and Beyond : A Technical Survey on Decentralized Digital Currencies." In: *IEEE COMMUNICATIONS SURVEYS & TUTORIALS* PP.99 (2015), pp. 1–1. ISSN: 1553-877X. DOI: doi:10.1109/COMST.2016.2535718.

[58] Xiaowei Xu *et al.* "Scan: a structural clustering algorithm for networks." In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2007, pp. 824–833.

[59] Kaizhong Zhang and Dennis Shasha. "Simple fast algorithms for the editing distance between trees and related problems." In: *SIAM journal on computing* 18.6 (1989), pp. 1245–1262.

附录 A　星云链账号地址设计

A.1　地址校验码

类似于比特币和以太坊，星云链的帐号也是由椭圆曲线算法作为其基础加密算法。用户的私钥是随机生成的 256 位二进制数，通过椭圆曲线乘法生成 64 字节的公钥。比特币和以太坊的地址都是由公钥通过确定性的哈希算法运算得到，不同的是：比特币地址有校验码设计，防止用户输错了几个字符而把比特币误发给其它用户，以太坊却没有校验码设计。

我们认为从用户的角度看，校验码的设计是合理的，因此星云链的地址也包含了校验码，具体计算方法如下：

```
Data = sha3_256(Public Key)[-20:]
CheckSum = sha3_256(Data)[0:4]
Address = "0x" + Hex(Data + CheckSum)
```

公钥的 SHA3-256 摘要的后 20 位字节作为地址的主要组成部分，对其再做一次 SHA3-256 摘要，取前 4 位字节做校验码，其相当于在以太坊的地址尾部加上 4 字节的校验码。

例如：

1. Alice 的以太坊钱包标准地址是 0xdf4d22611412132d3e9bd322f82e2940674ec1bc ；
2. 最后星云链钱包地址是：0xdf4d22611412132d3e9bd322f82e2940674ec1bc**03b20e40** 。

以太坊地址包括 0x 前缀共 42 位字符，星云链地址增加了 8 位字符的校验码，共 50 位字符。

A.2　扩展地址验证

除了支持 50 字符的标准地址，为了用户转账的安全，我们还支持扩展地址。借鉴了传统银行转账的设计：转账时除了验证对方的银行卡账号，汇款人还必须输入对方的姓名，只有银行卡账号和姓名都匹配，转账才能正确进行。扩展地址的生成算法如下：

```
Data = sha3_256(Public Key)[-20:]
CheckSum = sha3_256(Data)[0:4]
Address = "0x" + Hex(Data + CheckSum)


ExtData = Utf8Bytes({Nickname or any string})
ExtHash = sha3_256(Data + ExtData)[0:2]
ExtAddress = Address + Hex(ExtHash)
```

扩展地址在标准地址的尾部追加了 2 字节的扩展验证，共 54 位字符。加入扩展信息，使得在星云链钱包应用中可以增加另一个要素验证。例如：

1. Alice 的钱包标准地址是 0xdf4d22611412132d3e9bd322f82e2940674ec1bc03b20e40，加入昵称“alice”后的扩展地址是 0xdf4d22611412132d3e9bd322f82e2940674ec1bc03b20e40**e345**；

2. Alice 告诉 Bob 扩展地址 0xdf4d22611412132d3e9bd322f82e2940674ec1bc03b20e40e345 和昵称 **alice**；

3. Bob 在 Wallet 应用中，输入 0xdf4d22611412132d3e9bd322f82e2940674ec1bc03b20e40e345 和 alice；

4. Wallet 应用验证钱包地址的一致性，避免 Bob 错误的输成了其它人的帐号。

标准地址和扩展地址均可以用于转账，我们的星云链钱包应用将会强制使用扩展地址，转账时需要提供对方的昵称，验证地址一致性，进一步加强安全校验。

附录 B　相似智能合约搜索

代码相似度的主要难点在于智能合约高级语言的结构性特点和逻辑表达形式的多样性。目前学术界对代码相似度的算法大概有下面这些流派：

- 字符串编辑距离

  把输入 query 代码和候选源代码都当做文本，利用字符串编辑距离来衡量字符串之间的相似度。编辑距离 (Edit Distance) 指的两个字符串之间，由一个转成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。一般来说，编辑距离越小，两个串的相似度越大。这种字符串编辑距离算法既可以用在源代码比较，也可以用在中间语言 (intermediate representation)，甚至可以用在机器语言中。

  为了提高字符串编辑距离算法的健壮性，会对源代码做一些不改变语义的转换，例如：去除空白字符，去除注释，局部变量名统一用'$' 代替，代数表达式的归一化表达等。该算法运算速度快，简洁高效，但是对于复杂程序适应性较差，完全没有考虑到代码的语法和组织结构。

- Token 序列

  Token 序列表示法是将输入的源代码通过词汇分析器分析，将源代码转换成 Token 序列。两个程序的相似度就是 Token 序列的相似度，可以用最长公共子串，或者相关匹配算法 (如后缀树匹配算法) 度量程序间的相似程度，这样做可以检测出具有不同句法但却有相似功能的代码段，但该方法在程序相似度度量时隐藏了程序的组织结构。

- 抽象语法树 (Abstract Syntax Tree，AST)

  抽象语法树 (AST) 是对源代码用语法分析后形成的中间表达形式，通过比较子树与其他子树之间的相似程度对程序的相似度进行度量。比较两棵树之间的相似度可以用树的编辑距离算法 [59]，精确的树编辑距离算法复杂度较大，文献 [25] 提供了一种近似的快速算法，文献 [15] 提出对语法树做哈希指纹，使得语法树比较算法可以在海量数据集上进行高效检索。

- 程序依赖图 (Program Dependency Graph, PDG)

  程序依赖图 PDG [18] 可以表示程序内部数据以及控制依赖关系，能够在语义级别上对程序代码进行分析。相似代码规约成为寻找同构子图，这是 NP 完全问题，算法复杂度很高，目前只能用一些近似算法。

我们认为上述各种算法都在不同维度描述了代码的文本、结构、语法上的相似度。Source Forager [27] 提供了一个很好的工程思路，把各种不同维度的相似度指标刻画为不同的特征，每个特征都表征了代码某种方面的相似度衡量，最后用向量的相似度来做总体的相似度衡量，综合了以上各种算法的优点。星云链也借鉴这种思路，实现智能合约的相似搜索。我们把函数作为智能合约代码搜索的基本单位。

表3定义了候选的代码相似度特征，我们接下来进一步描述每个特征的定义和它们的相似度计算函数：

- **Type–Operation Coupling**：该特征是一个二元组集合，二元组包含变量类型，以及变量类型的算符，即 (type, operation) 对。原生数据类型一般和算术运算符、逻辑运算符以及关系运算符配对，例如 $(int, \geq)$；自定义类型 (例如 struct) 一般和成员函数配对，例如 (Bar, .foo) 表示数据类型 Bar 的字段 foo 被访问。按照这种方法，把一个函数的代码体内部所有变量操作都变成二元组，并且去重以后，

表 3: 代码相似度特征族表

Feature-Class	Brief Description
Type–Operation Coupling	types used and operations performed on the types
Skeleton Tree	structure of loops and conditionals
Decorated Skeleton Tree	structure of loops, conditionals, and operations
3 Graph CFG BFS	CFG subgraphs of size 3, BFS used for generating subgraphs
4 Graph CFG BFS	CFG subgraphs of size 4, BFS used for generating subgraphs
3 Graph CFG DFS	CFG subgraphs of size 3, DFS used for generating subgraphs
4 Graph CFG DFS	CFG subgraphs of size 4, DFS used for generating subgraphs
Library Calls	calls made to libraries
Type Signature	input types and the return type
Local Types	types of local variables
Numeric Literals	numeric data constants used
String Literals	string data constants used

就用一个二元组序列体现了这段代码的 Type–Operation Coupling 特征。我们认为相似功能的代码，也应该具有相似的变量操作集合。但是我们并不关心二元组的先后顺序，所以该特征丢失了代码的逻辑结构信息，只能部分代表代码的特征。

Type–Operation Coupling 特征之间的相似度可以由雅克比相似度定义，即如果给定两个集合 S_1 和 S_2，雅克比相似度由如下公式定义：

$$sim_{Jacc(S_1,S_2)} = \frac{\mid S_1 \bigcap S_2 \mid}{\mid S_1 \bigcup S_2 \mid} \tag{12}$$

- **Skeleton Tree**： 基于代码的抽象语法树，但是只保留了循环 (for, while, do...while) 和条件语句 (if...else)，其余节点从树中全部移除。我们认为相似功能的代码，应该具有相似的循环和条件语句结构。

  Skeleton Tree 的相似度计算基于树的编辑距离。定义 d_r 为两棵树的预估编辑距离，此距离仅仅由树的大小决定，即：

$$d_r(T_1, T2) = \frac{\mid size(T_1) - size(T_2) \mid}{max(size(T_1), size(T_2))} \tag{13}$$

  假设 D_T 是编辑距离的阈值，设定为 0.5。我们进一步得到两棵树的近似编辑距离公式：

$$d_t(T_1, T2) = \begin{cases} d_r(T_1, T2) & if\ d_r(T_1, T2) \geq D_T \\ \dfrac{max\begin{pmatrix} ed(pre(T_1),\ pre(T_2)), \\ ed(post(T_1),\ post(T_2)) \end{pmatrix}}{max(size(T_1), size(T_2))} & otherwise \end{cases} \tag{14}$$

pre(T) 代表树的先序遍历序列，post(T) 代表树的后序遍历序列，$ed(S_1, S_2)$ 代表序列 S_1 和 S_2 的编辑距离。Skeleton Tree 的相似度可以由以下公式计算:

$$sim_{Tree}(T_1, T2) = 1 - d_t(T_1, T2) \tag{15}$$

- **Decorated Skeleton Tree**：和 Skeleton Tree 相似，除了循环和分支节点以外，还保留了大部分运算符 (例如 +, -, <)。但是赋值运算符去除了，因为这些大都是噪音。
- **K-Subgraphs of CFG**：基于函数的 CFG 图的 k-子图实现。k-子图按如下方法定义：给定 CFG 图和某节点，我们做广度优先遍历 (BFS) 或者深度优先遍历 (DFS)，直至遍历节点数为 k，形成的子图为 k-子图。如果遍历结束后，节点数不足 k，丢弃该子图。遍历 CFG 图的每个节点，我们可以得到所有的 k-子图。对于每个 k-子图，我们用 k^2 位整数表示，参考图20。所有的 k-子图形成了一个整数集合。

  3 Graph CFG BFS: k = 3，BFS 遍历

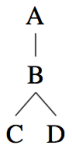
  4 Graph CFG BFS: k = 4，BFS 遍历

  3 Graph CFG DFS: k = 3，DFS 遍历

  4 Graph CFG DFS: k = 4，DFS 遍历

  相似度可以由广义雅克比相似度公式计算：给定向量 $\vec{x} = (x_1, x_2, ...x_n)$ 和 $\vec{y} = (y_1, y_2, ...y_n)$，广义雅克比相似度定义为

$$J(\vec{x}, \vec{y}) = \frac{\sum_i min(x_i, y_i)}{\sum_i max(x_i, y_i)} \tag{16}$$



	A	B	C	D
A	0	1	0	0
B	0	0	1	1
C	0	0	0	0
D	0	0	0	0

图 20: 4-graph 示例，邻接矩阵的元素为二进制串“0100 0011 0000 0000”，十进制数为 17152

- **Library Calls**：合约中如果发生调用其它库合约，所有库合约的地址也被记录下来，由雅克比相似度公式计算相似度。
- **Type Signature**：该特征由输入参数类型和返回参数类型组成，由雅克比相似度公式计算相似度。例如对于下面的智能合约代码，函数 getBalance 的 Type Signature 特征为向量 (address, uint256)。

```
contract addressTest {
  function getBalance(address addr) returns (uint) {
        return addr.balance;
  }
}
```

- **Local Types**：该特征为函数体所有局部变量类型的集合，由雅克比相似度公式计算相似度。
- **Numeric Literals**：所有数值常量的集合作为 Numeric Literals 特征，由雅克比相似度公式计算相似度。
- **String Literals**：所有字符串常量的集合作为 String Literals 特征，由雅克比相似度公式计算相似度。

特征族可扩展，可以方便的把各种新的特征加入进来。基于每个特征都有一个相似度计算，我们把所有特征的相似度做加权和，得到最终的代码相似度：

$$sim_{combined}(\vec{A}, \vec{B}) = \frac{\sum_{c=1}^{n_{cl}} sim_c(\vec{A_c},\ \vec{B_c}) \cdot w_c}{\sum_{c=1}^{n_{cl}} w_c} \tag{17}$$

其中，$\vec{A}$ 和 $\vec{B}$ 是特征向量；n_{cl} 是特征族个数; sim_c 是针对特征 c 的相似度计算函数；$\vec{A_c}$ 和 $\vec{B_c}$ 是特征 c 的特征向量；w_c 是特征 c 的权重。权重是可以在大量测试集上，通过机器学习算法训练得到的。

附录 C　白皮书版本更新日志

- [1.0.1] - 2017.09
    1. 细化了几处语句表达；
    2. 调整了部分图表格式；
    3. 改进了公式的可读性；