OPEN SOURCE COIN

开源社区的信任和可持续性

DRAFT 4.0

ALEXIS SELLIER† ELEFTHERIOS DIAKOMICHALIS† JAMES HAYDON†

Abstract. 开源运动的成功植根于点对点的分布态势和个体与个体的协作耦合。开源，诚然已从软件开发的理想主义孤篱生长成行业标准的高阁，但缺乏可持续的融资手段仍是横亘在此的负重。在这里，我们提议采用点对点协议创造的开源社区货币 (oscoin)，帮助我们一同开创可持续的，自由的开源软件经济。

1. 背景

> 正是在这种情况下我们才更要明白：只有光复开源文化的清朗，才能让一切由开源孵育的可能性峥嵘而出，才能使集体中的分散智慧迸发人性之光，这时候人间居所也将宛如天堂——这个世界的一切皆为了人
>
> — 万物开源宣言

数字稀缺性的出现，可以在无需信任的第三方参与的情况下，更简单透明地为网络参与者们提供经济激励和动机。比特币的出现和紧跟而来的以太坊引领了众多替代（加密货）币在网络效用和经济机制上的争流热潮。

如果说比特币是依据交易确认和保护网络的行为奖励运营者，那么 Zcash 和 Decred 等项目则进一步让开发者和其他的价值创造者们拿到了行为红利。进步虽可观，但我们还缺乏一种措施，为所有无论是直接进行代码贡献还是间接参与开源基础设施建设的人提供可持续的激励。

这只是免费软件面临的众多动机缺乏问题中的一例，也是本文希望着重讨论的问题 [4]，动机问题将很大程度上影响到整个软件生态。

1.1. 开发者激励和可持续性.

在我们深溯这项问题前，让我们回顾一下免费开源软件诞生伊始的条件。我们日常使用的软件大部分都是基于免费公开的代码写成。在数字化程度渐深的现代社会，免费软件项目已经成为了社会商品服务衍生与创新的重要支柱。

Github 这类软件托管站点和 Stack Overflow 这类社区的出现让开源成为了广受欢迎的软件工程范式，人人触手可用的高质项目也因此纷纷涌现。这一现象让很多公司的开发周期缩短，市场反应速度加快，而且在开源软件的教育背景下技术人员储备池得以更新，人才招聘因此受益。

当今的很多免费软件项目都是由个人或小团队发起，成立之初往往是为了解决个人，社会抑或是技术相关的问题。在检视他们的动机时，我们常常看到的理由包括个人成就感，为了声誉或为了获取知识，也包括随信仰而来的责任感，以及社群归属感等。

大部分开源软件项目似乎都是基于上述原因开始的，然而那些真正具有里程碑式意义的项目则需要大量的时间与财富资源才能运转下去。这就暴露出一个基本矛盾：现在公众使用的软件设施是基于自愿这一前提开发的。虽有少部分开发者们找到了一些支援工作进行的经济来源，而大部分开发者们还在为响应社区的需求而殚精竭虑，在挤出空闲时间维护代码中捉襟见肘。因此他们不堪重负，放弃项目的结局已屡见不鲜。这里暴露的问题也就是我们想强调的，即，如何发展出一套健康可持续的免费软件运维和融资方式。

2. Oscoin 网络

在本文中，我们将介绍 oscoin，这是一种专为解决开源可持续性问题而设计的加密货币。

Oscoin 是一个由参与对一个共享转账账本展开的共识协议的计算机组成的公共网络。该账本可以实物化为全局状态 $\mathcal{S}$，包括所有参与协议的开源项目的最简注册表 $\mathcal{R}$，包含所有通证持有者余额的账户集合 $\mathcal{A}$，网络中实体间相对关系，包括注册项目之间软件依赖关系的网络图 $\mathcal{N}$。

这个网络的目的是为 oscoin 这个数字货币提供保障，同时通过去中心控制和去中介的方式奖励网络中最能创造价值的项目

2.1. Oscoin 区块链.

为了设计一个所有网络参与者共享的安全、开发、无需许可的货币，我们采用 Nakamoto [1] 与 Wood [2] 类似的方案，将 oscoin 设计为一个区块链协议。

区块链协议通过去中介的资产交易方式解决了大规模信任的问题，同时也实现了抗审查性——此为开放网络的一个本质特性。我们的解决方案中的 oscoin 这一加密数字货币符合这一特性，它受控于所有的网络参与者而非一个中心权威。我们相信这对 oscoin 的成功至关重要，而且区块链技术也是能够承载其实现的最具前景的解决方案。

虽然说 oscoin 具体采用什么共识机制不是核心问题，但我们相信从长远来看，一个最能够让用户开放参与的网络才是最佳选择。进一步说，我们也应让轻客户端的网络参与成为可能，而不是让本应受益于网络的他们因为成为全节点的成本过于高昂就望而却步。现在的区块链协议中有这些特性的一系就是工作量证明机制。当然我们也充分认识到大规模工作量证明对环境的影响，所以随着研究的深入，我们也倾向于切换至更高能效的协议机制。[1] [2]

2.2. Oscoin 财政系统.

财政系统（图一 1）持续依据网络中项目的重要程度为项目进行排名，并用 oscoin 进行奖励。这一机制是网络的重要构成部分，作用即是为维护者持续提供参与激励。

2.2.1. 概览.

- 系统维护者和代码贡献者在网络中注册项目并在软件项目开发中协同合作。
- 系统维护者合并提交的代码贡献，向账本同步项目状态，包括项目的依赖关系信息和代码贡献元数据 (§4.4.2)
- 基于上条中提到的元数据，每隔 κ 个区块系统计算一次 osrank。(§3)
- 根据 osrank 的排名进行 oscoin 奖励，奖励由项目的相关智能合约控制派发。(§5)
- 依据项目的智能合约各项系数，奖励由代码贡献者和系统维护者共享。
- oscoin 的代币奖励量遵循固定的锁仓时间表，只有满足锁仓时间才可由项目自由动用。

每一个区块都将伴随着一定量 B_r 的新 oscoin 铸造出来，作为“区块奖励”或“币基”。这些新币会由财政系统分配给两个角色：网络运营者 ($\mathcal{M}$) 和开源项目 ($\mathcal{P}$)。

我们必须提到的一个关键问题是有多少比例的 B_r 将会分配给网络中的的单个开源项目。关于此，我们利用了 oscoin 网络中的网络图来描绘项目与贡献者之间的价值流，并为双方都分配一个权重。这个权重，或称 osrank，代表着网络中每一个主体的重要程度。

Date: February 2019.

†Monadic, {alexis,ele,james}@monadic.xyz.

[1] 时至今日，正如比特币所展示的那样，工作量证明的区块能在大部分移动设备上被便宜且安全地验证。相反，作为工作量证明协议替代品的权益证明协议依赖于通证持有者的签名，这需要取得账户余额才能进行验证。因此，第三方需要下载区块数据并验证，或者必须重复对账户余额发起请求。这些在受限环境下都不可行。

[2] 权益证明协议家族中较有前景的一个候选者是由 Filecoin [3] 提出的方法。

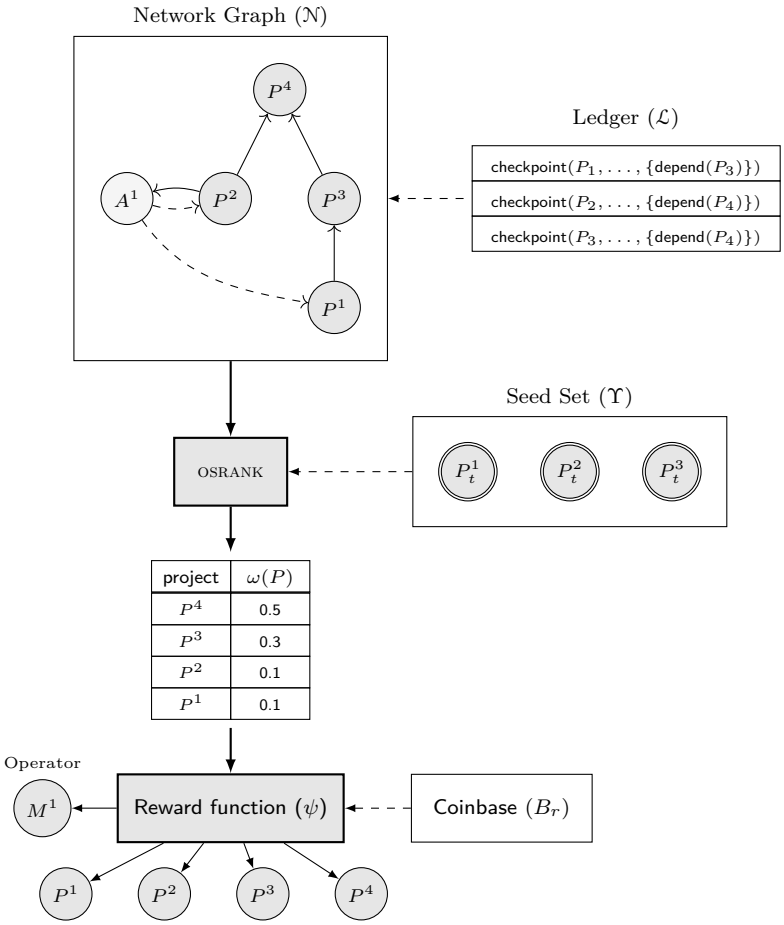

FIGURE 1. The OSCOIN Treasury System

2.2.2. *算法*. 用 t 来代表每 k 个周期（区块）中待分配给 n 个项目的代币总量。对于每一个给定的项目 P，t_P 表示在某一周期 κ 中 OSRANK 函数 $\omega(P)$ 计算出的应分配给项目 p 的 OSCOIN 数量，t_P 为 t 和 $\omega(P)$ 的 函数，见下：

$$\psi(t, \omega(P)) \to t_P.$$

为了防止小规模的女巫攻击，支付函数将考虑对 $\omega(P)$ 设置一个最小阈值，低于该值的项目将不会受到奖励。。未来可以通过调整扩大或缩小奖励范围，达到均衡目的。

2.2.3. *锁仓*. 为了调控激励政策，被用作奖励的那部分 OSCOIN 的释放将遵循固定的锁仓时间表。这其实意味着激励发放不是实时，而是延时到账的。这样的设置是为了抑制投机行为，从长远上给 OSCOIN 的持有者一个乐观的愿景。如果不做这样的锁仓要求，那么 OSCOIN 的流动将全无限制。

宏观上讲，OSCOIN 财政系统可以解决开源变现和可持续性的问题。项目在准许他人免费使用软件的同时，可以得到 OSCOIN 作为收入回报。这样的财政系统不需要开源项目为了求生存而转变他们的工作方式，不需要他们再授权或者成立公司。这是与驱动开源项目成功的诉求相吻合的一种新模式。[3]

3. OSRANK

OSCOIN 的基础机制之一是将区块奖励的一部分分发给网络内的高质量项目，所以 OSCOIN 协议必须对哪些项目质量高达成一致。为了选出收到 OSCOIN 奖励的项目并决定其奖励份额，OSCOIN 网络使用 OSRANK 算法对项目之间的关系图和其对网络的贡献进行评价。OSRANK 是著名的 PageRank 算法的变种，其输出形式是给每个项目 P 一个对应的分数 $\omega(P)$。

3.1. **出发点.** 虽然判断一个开源软件包的重要性是高度主观的行为，但是通过研究开源软件的依赖关系和贡献结构，能够得到一个项目相对重要程度很有价值的信息。

与 Google 围绕分析网页超文本结构的 PageRank 算法开发出的大规模搜索引擎类似，开源软件依赖与贡献关系的图结构也能对表示其在生态中被信任程度和相对价值提供重要的依据。

3.2. **原理.** 一个系统作为整体产生的价值可能十分明了，但是特定子系统贡献的价值往往并不清晰。而且不同的子系统将价值转换成收益的能力本质上就有所不同。那些位于与其他域相邻的边界位置的子系统往往在这个方面更有优势，尽管它们产生的价值可能更多汲取自其他子系统。这种子系统价值流输入输出的不平衡对系统整体不利。

幸好人类活动并不发生在虚空之中：无论是脑力劳动、内容创作、金融活动等等，实体之间有意义的互动都被每个实体拥有的相对于系统整体而言的相对价值深深影响。重要的一点是，在活动的过程中产生了一系列的人类活动产物（超媒体链接、合同、转账、依赖关系等等），这些痕迹是不同实体之间关系具象的证明。

由 PageRank 以及类似的度量方式带给我们的一个基本思考是，通过对系统内代理人有意义活动痕迹的近似模拟，产生与观察相符的成系列的人类活动产物，我们就可以研究子系统中的价值流动，进而估算内含的价值分布。就 PageRank 而言，其模拟的是人类在互联网上搜索相关度高的数据，得到的是一系列链接。一个高级的算法在选择下一个链接的时候会使用尽可能多的启发式的方法（正如人类行为那样），比如，衡量链接文本的相关性。如果计算资源稀缺，粗糙的算法会简单地随机选择下一条链接。这就是产生经典的 PageRank 公式的原因。该方法已经在互联网上被证明十分成功。

[3]见 Eric S. Raymond 的《大教堂与市集》

3.3. **标记符号.** 在这一部分，我们将使用如下表示方法：图指有限有向图。对于图 G，顶点的集合表示为 $V(G)$, 边的集合表示为 $E(G) \subseteq V^2$。G 的一条边 $e = (x, y)$ 表示为 $x \xrightarrow{e} y$。对于一个给定的 G 顶点的子集 X，$G[X]$ 表示 X 的诱导子图，即以 X 为顶点集、以 $\{x \xrightarrow{e} y \mid e \in E(G), \{x, y\} \subseteq X\}$ 为边集的图。图 G 内一次长度为 n 的途径是图的态射 $L_n \to G$，L_n 是第 n 个线性图：$V(L_n) = \{i \in \mathbb{N} \mid i \leq n\}$, $E(L_n) = \{(i, i+1) \in \mathbb{N}^2 \mid i < n\}$。

3.4. **Oscoin 里的排序实体.** 被 OSCOIN 账本捕捉到的与 OSS 开发相关的人类活动产物有：

- 项目的维护，
- 项目之间的依赖关系，
- 项目内经过签名的代码贡献。

我们将这些交互总结为一个带权有向图 G。权重反映相应边的相对重要性，并设置为算法的超参数。以一个顶点为起点的所有边的权重之和为 1。

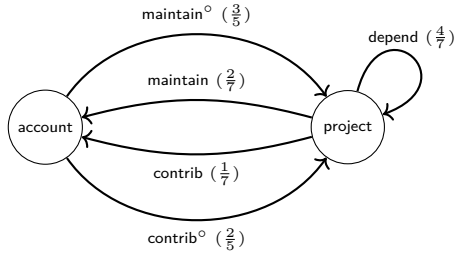

FIGURE 2. 开源软件开发中实体关系的图解（权重仅为示例）

值得注意的是，某些关系会是双向的，但每个方向的权重并不一定相同。例如，contrib 和 ⟦contrib° 从属于同一个人类活动产物：一个账户对一个项目的代码贡献。contrib 边表示的命题为：对于项目而言，贡献者有价值。反向的 contrib° 边表示的命题为：项目对于贡献者来说是有价值的。对于这个不太明显的反向流动的解释是，开发者往往不会在他们觉得没有用的项目上贡献时间。

虽然依赖关系图是一个项目从另一个项目汲取价值最直接的指示，但添加对应贡献和维护的边仍然能够服务于两个目的：

- 依赖关系图是有向的：方向从用户端应用指向核心库。这会将分数极大地向基础库和开发工具倾斜。将贡献包括进去将带来反向的流动，标识出有价值的用户端应用。例如，一个功能全面的开源文本编辑器对于开源开发者来说十分有价值，但是它可能永远不会在另一个软件明确的依赖关系中出现。但是，向这个文本编辑器贡献代码的开发者也有可能向这个编辑器使用的包贡献代码。这将使得算法从代码库向用户端包反向流动价值。
- 仅仅依靠依赖关系无法充分地将不同域连接起来。开发者经常向相邻的生态贡献代码和工具，使原本独立的项目能够有所联系。这样，一个将依赖关系和代码贡献结合起来的图可能会充分地联系起来，揭示开源项目潜在的价值。

我们提议将这些人类活动产物（依赖关系、代码贡献等）记录为区块链的转账。这样，就可以在 G 之上可验证地构建一个能够达成共识的有限有向图 $\mathcal{N}$（即在图 Graph/G 的一个切片范畴内），用来表示 OSCOIN 网络图。

反过来，这使得全局价值分数能够依据模拟随机漫步的特定随机过程分配给实体，即 $\mathcal{N}$ 内的活动轨迹结合从 G 得到的取样概率。例如，一个贡献者决定在一个高价值的项目上进行开发，贡献一个变动。贡献者在这个过程中将另一个项目加入了依赖关系。这个依赖关系引入了一个来自上游包的错误。因此这个贡献者转换方向来解决这个存在在路径中的错误，以此类推。这个过程很像人类点击链接搜索信息过程的模拟。这个模拟活动告诉了我们在广泛的 OSS 生态系统中子系统之间的价值流动。

这个我们称为 OSRANK 的分数被用来将判断项目对 OSCOIN 网络的价值，并将区块奖励的一部分分发给项目。

3.5. **定义.** 为了实现 OSRANK，我们采用如下蒙特卡洛类算法：对于网络内的每个顶点，都将开始 R 个随机漫步。在选择下一顶点时，首先会由取样的边基于 G 内的权重选择边的类型。下一步行动会由边的类型决定：

- 对于依赖与维护边，所有顶点的可能性相同。
- 对于从贡献者 x 指向项目 p 的边 $x \xrightarrow{\mathsf{maintain}^\circ} p$，项目 p 被选择的概率是 $\frac{p_x}{N_x}$ 。p_x 是 x 贡献给 p 的数量，N_x 是 x 贡献的总数。
- 对于 $p \xrightarrow{\mathsf{contrib}} x$ 边，贡献者 x 被选中的概率为 $\frac{p_x}{N}$。N 是 p 从所有贡献者处收到的贡献的总数。

对于每一步，漫步在项目顶点处结束的概率是 $1 - \epsilon_{\mathsf{project}}$，在账号顶点则为 $1 - \epsilon_{\mathsf{account}}$ (这里的 $\epsilon_{\mathsf{project}}, \epsilon_{\mathsf{account}} \in (0, 1)$ 是阻尼因子)。这就产生了 nR 个随机漫步的集合 W。n 是顶点的总数。一个项目 x 的 OSRANK 为：

$$\omega(x) = \frac{W_x(1 - \epsilon_{\mathsf{project}})}{nR}$$

W_x 是 W 内经过 x 的漫步的次数，即：

$$W_x = \sum_{w \in W} |w[\{x\}]|.$$

相似地，账户 a 的 OSRANK 为：

$$\omega(a) = \frac{W_a(1 - \epsilon_{\mathsf{account}})}{nR}.$$

3.6. **女巫攻击.** 为了维持 OSCOIN 网络图的完整性，非法的生态系统必须被剔除。这意味着以利用 OSRANK 为目的发起的欺骗性的项目与贡献结构必须被算法惩罚。这种针对 PageRank 的女巫攻击已经在之前被细致地研究过。

为了解决这个问题，我们以 TrustRank [7] 的逻辑为基础，提出了对 PageRank 算法的修改。这包括两个阶段：

- 在第一阶段，所有的随机漫步开始于起始种子集合 Υ 内的一个顶点。是为了这个目的而从所有项目和账户顶点中选出的一个子集。这样得到的分数会偏向与种子集合的交互，但是这些分数不会直接被奖励机制使用。这个过程是为了得到一个阈值 t，进而确定实体的合法性：任何分数低于阈值的实体都不会继续进行下一阶段。这一阶段的输出是网络图的一个子图 $\mathcal{N}_t$。
- 在第二阶段，将会在子图 $\mathcal{N}_t$ 上以同样的方式运行算法，只不过不再使用种子集合。第二阶段输出的结果为 OSRANK。

在选择种子集合的时候必须十分小心，以确保其足够大并且足够分散，使得从这些顶点开始的随机漫步能够能够触及到网络内所有的合法项目。维持这样一个种子集合有很多方式。这个部分将在今后进行更深入的探讨。

3.7. **信息可靠性与 osrank.** 实施 OSRANK 时需要考虑的一个重要因素是账本上信息的可靠性，即，依赖关系。这是网络图 N 实质化的基础。

为了使 OSRANK 成为在网络内价值的良好代表，账本上的信息必须准确。虽然提出的算法能够允许一定程度的误差，但我们也必须依靠大部分所声明的关于依赖关系与贡献的事实的准确与及时。按事实情况来说，项目与维护者存在固有的社会层面的动机来及时更新依赖信息，也就是说，依赖信息对每个人都是公开可见的，项目也需要在依赖者间维持自己的声誉。此外，项目存在着被分叉的风险，而 OSRANK 又与经济激励有关，所以对于项目来说，来自自己周身生态环境的信任就显得很重要。

然而，从长期来看，这些激励可能不足以阻止不诚实方从这个系统中取得不正当利益。我们将在第七章中简要讨论可能的解决方案。§7

3.8. **实施.** 为了确保 OSRANK 的计算不会变得过分昂贵，以至于运营者无法进行计算，系统使用了以下几个方法。

- 增量蒙特卡洛。为了使 OSRANK 值能在网络图改变时及时更新，我们使用了增量算法。这个算法依赖的基础事实是在两次计算之间，只增加或减少了少量顶点或边，所以无需每次从零计算 OSRANK。因此，大多数在之前计算中实施的随机漫步在更新后的图中仍然有效。例如，如果从上次计算至今只增加了 $p_1 \xrightarrow{\mathsf{depend}} p_2$ 一个依赖关系，那么只有在上次计算过程中经过 p_1 的随机漫步才会失效。

  在实际运行过程中，网络的运营者会将上次计算的随机漫步集合存储在缓存内，并及时在更新时删除无效漫步，加入新的漫步。关于增量 PageRank 算法的细节请见 *。[8]

  由于所有计算必须是确定性的，这些漫步并不是真随机。这些随机漫步的启动必须借助一个内置的伪随机数产生器[4]。

[4]在无需许可设定中设计一个适当的随机数生成器是一个正在被活跃地研究的领域。现存解决方案存在近期的文献中有所描述，这部分内容超出了本文的范围。

- 长周期。在这个场景下，根据 OSRANK 做出的支付被控制得频率很低。这就必须将奖励周期值设定得很大，比如，每月一次。这种情况下，滥用者可能在奖励发放区块之前修改数个项目的网络结构。为了缓解这个问题，边的权重会考虑其在上个周期内的寿命。例如，如果一个依赖关系在奖励区块之前 n 个区块被添加，那么它的权重比例就只有 n/κ。

正如 §2.2解释的那样，在账本上存储 OSRANK 是 OSCOIN 的货币政策的基础。而且，OSRANK 与 $\mathcal{N}$ 将会可以被智能合约调用，项目能够使用这是数据定制特性。

4. OSCOIN 账本

OSCOIN 网络包含一系列节点，或称副本，用来执行去中心化协议。这些节点共同组成复制状态机，元素包括一个状态集合 $\mathcal{S}_0, \ldots, \mathcal{S}_n$，一个转换函数 Ξ，一系列的输入 $B_0, \ldots, B_n$，或称“链”，其中 B_0 为创世区块，以及一系列输出。

OSCOIN 账本，即 $\mathcal{L}$，是经过全网达成共识的有序交易集。更新账本的过程就是在网络中提交交易确认的过程。所有被确认的交易共同组成状态 $\mathcal{S}$，即现阶段全局交易状态。

本章将描述如何在账本上更新交易，以及交易如何影响全局状态 $\mathcal{S}$。

4.1. **供应.** OSCOIN 的供应量遵从固定的供应曲线，且每个周期都有 $B_r > 0$ 的增量。这确保了 OSCOIN 持续流入财政系统，奖励网络中有价值的工作。

4.2. **账户.** 账户持有者用签名解锁账户中的 OSCOIN，账户中保存着可以用来收发 OSCOIN 的地址。这是一种与比特币的 *utxo* 不同的类以太坊账户的设计。

通过转账即可创建新账户，账本余额为 0 时可以销户。所有账户的集合为 $\mathcal{A}$。

账户账本余额 A_b 被记录在全局状态 $\mathcal{S}$ 中，账户地址 A_a 可查询该账户余额。记为，$\mathcal{S}(A_a) \to A_b$。

每个账户还关联一个智能合约，记为 A_{contract}，可确认和处理账户中流入和流出的 OSCOIN。

4.3. **转账 oscoin.** 集合 $\mathcal{A}$ 中的每一个账户都有一个 OSCOIN 余额：

$$\mathsf{bal}(a) \in \mathbb{N}_{\geq 1} \mid a \in \mathcal{A}.$$

账户的转账操作表示如下

$$\mathsf{transfer}(x, y, n) \mid x, y \in \mathcal{A}, n \in \mathbb{N}_{\geq 1}$$

其中 x 是提取 OSCOIN 的账户，y 是 OSCOIN 存入的地址，n 是交易量。只要 $\mathsf{bal}(x) \geq n$（转出账户余额大于转出数量）则交易合法，x 与 y 各自关联的智能合约将授权执行这次转账交易。

4.4. **项目.** 项目 P 可表示为一列元组数据：

$$P = \langle P_a, P_h, P_s \rangle$$

其中 P_a 指的是项目的地址，是独特身份标识。P_h 为项目当前哈希值，P_s 以标准典式记录项目 URL。

项目地址（字母数字字符串）是每个项目在账本交易中的独立身份标识。每个项目都拥有一个与地址相关联的特殊账户，叫做项目资金。项目哈希是每个项目被纳入账本时的源代码摘要。记录项目 URL 是出于便利性考虑，为了方便检索代码源。注意，通过 P_s 攫取的源代码的哈希值必须等于 P_h，否则项目将被视为非法。

4.4.1. *注册和维护.* 项目须先进行注册交易才能加入网络：

$$\mathsf{register}(P_a, P_s)$$

被秘钥 k_1 签名过的注册交易才可在之后被更新到账本中。只有当地址 P_a 没被启用，且 P_s 是有效 URL 时，转账才有效。

注册成功后，账户中一小部分 OSCOIN 经 k_1 签名后锁定。一定程度上防止大量弃用项目的出现扰乱账本。以上程序执行后，项目元组数据 $P = \langle P_a, P_h, P_s \rangle$ 将被部分重置：$P_h = \varnothing$。

项目 P 现有的秘钥串 P_K 为 $\{k_1\}$，我们将这些秘钥串称为项目的维护者。秘钥才是项目的实际控制者，享有项目管理权。可以进一步向 P_K 中添加新秘钥：

$$\mathsf{addkey}(P_a, k)$$

添加前，k 是不存在于原秘钥串 P_K 中的合法秘钥。秘钥也可以被移除：

$$\mathsf{removekey}(P_a, k)$$

移除前，k 存在于 P_K 中。当项目无人运维时，可以从账本中注销：

$$\mathsf{unregister}(P_a).$$

项目注销时，之前质押的 OSCOIN 经 k_1 签名退还回账户中。以上所有步骤都须经 P_K 中的合法秘钥签名实现。在账本中注册的项目可通过注册条目 $\mathfrak{R}$ 中的项目地址检索出来，记为：$\mathfrak{R}(P_a) \to P$。

4.4.2. *检查点.* 处于开发进程中的活跃项目的源代码会有经常性的变动。这意味着项目哈希 P_h 需要时时进行同步，才能反映当前的项目状态。哈希更新和通过如下

$$\mathsf{checkpoint}(P_a, P_{h'}, P_{s'}, C^*, D^*)$$

交易实现，其中 $P_{h'}$ 为项目新哈希。$P_{s'}$ 可进行源代码检索的 URL，C^* 是代码贡献的哈希链表，D^* 是依赖关系更新的清单。

代码贡献是元组数据表示为：

$$\langle C_{\mathsf{prev}}, C_{\mathsf{commit}}, C_{\mathsf{author}}, C_{\mathsf{signoff}} \rangle$$

其中：

- C_{prev} 是过去贡献的哈希，如果项目之前无任何提交代码，则用 来表示。C^* 的首个元素，必须关联到项目前一次检查点的最后一次贡献，这样就不会割裂对贡献的记录。
- C_{commit} 是对应的代码提交的哈希
- C_{author} 是代码提交者（贡献者）
- C_{sig} 是代码提交人的签名
- C_{signoff} 是验收秘钥，属于秘钥集合 $\in P_K$

每个贡献都必须由验收秘钥进行签名。签名过程表明此次贡献已经由维护者进行复核与确认。检查点本身也必须由 P_K 中的一把秘钥进行签名。

因为检查点记录了所有的源代码的变动，所以我们可以用一张哈希互联的清单来重现整个项目。当我们对项目的代码库进行交叉检验时，代码的作者以及将代码贡献签署生效的维护者都会体现在历史记录中。这样的可审计且不可篡改的历史记录为 OSRANK 提供了基础信息保障。注意这里只有贡献的元数据才会被存储上链。

理论上，如果项目 P' 在 P 的输入（Input）中，即，在 P 在源代码中引用或部分引用了 P' 或 P' 是一个构造/测试依赖成分，则我们称项目 P 依赖于项目 P'。例如，如果一个项目使用了纯粹功性能的包管理工具 *Nix* [9]，那在 OSCOIN 中声明的依赖关系应该与 Nix 中的依赖关系一一对应。

依赖关系更新列表 D^* 是一个存放依赖关系更新的列表。依赖关系更新有

$$\mathsf{depend}(P'_a, n) \quad \text{or} \quad \mathsf{undepend}(P'_a, n)$$

两种，都是指项目 P' 的第 n 个检查点（第一个检查点记为 n=0）。depend 更新增加一个新的依赖关系，而 undepend 更新移除一个依赖关系。更新将按 depend 的顺序执行，并且只有当项目没有该依赖关系时添加行为才有效，而 undepend 支队现存的依赖关系有效。当更新列表有重复时，检查点将无效。

作为项目维护者，增加一项依赖关系标志着项目实质的一连串改变：

- P 依赖于特定版本的 P' 真是存在并已被确认；
- P' 适宜作为 P 的一个依赖关系，例如，如果 P 有非常高的安全要求，P' 也能完全满足。

因为对一个项目的贡献承载着额外的权重，很有可能增加项目的 OSRANK，因此维护者有动力对项目规律地添加检查点。与此类似，增加依赖关系可能增加网络内的连通性，从而反过来间接地增加项目的 OSRANK。

4.5. **智能合约.** 最后，账本允许智能合约来对交易进行一定程度的调解。我们将用接下来的篇幅讨论智能合约自动化执行资金分配并进行准入管理的操作。

5. 智能合约

OSCOIN 协议将资金转给了一个由可更新的智能合约控制的特殊账户：项目资金，以此来确保资金通过 OSCOIN 财政系统进行分配，且项目以透明和负责的方式消费资金。除此之外，协议已经规定了项目资金账户只能由其智能合约控制。当贡献者决定向某项目贡献代码时，他们可以事先检查这个合约是否到位，是否可以进行合理更新，他们可以借此了解项目的奖励模型，确保他们可以正常收到报酬。

5.1. **定义.** 智能合约是一套函数，或者说处理程序，会在常规转账处理过程中被调用。特定的转账，比如 transfer，使用智能合约来扩展其的行为，从而允许项目在金融、所有权模型或治理等方面进行配置并自动化运行。

5.2. **接收 oscoin.** 当一个项目接收到 OSCOIN 时，与项目资金关联的只能合约就会被调用。智能合约内调用的特定处理程序取决于 OSCOIN 的发送者。当 OSCOIN 来自于财政系统时，将调用 RECEIVEREWARD 处理程序；当转账来自于别的来源时，RECEIVETRANSFER 将调用处理程序。这允许项目能够对来自财政系统的资金和收到的捐款有不同的处理方式

5.2.1. *来自财政系统的奖励.* RECEIVEREWARD 处理程序每隔 κ 个区块被调用一次，以便项目得到奖励。这个函数有三个参数：p 是包含项目数据的字典；$r \in \mathbb{N}_{\geq 1}$ 表示项目在本周期内收到的 OSCOIN 奖励的数量；$k \in \mathbb{N}$ 是现在所处的周期。这个函数必须返回一个元组集合，内容为将一定数量的 OSCOIN 分发给一组账户。这个集合称为分布 *distribution*。只要这个函数分发的 OSCOIN 总数不超过 r，函数就仍有效。任何没被分发的 OSCOIN 都将被销毁。

当一个项目首次注册时，RECEIVEREWARD 将返回空分布，即，整个奖励 r 都将被销毁。

handler RECEIVEREWARD(p, r, k)
  return $\varnothing$

一旦项目维护者决定了一个政策，他们可以发起一次转账来更新处理程序。例如，他们可以决定所有贡献者得到等量的未来奖励，并将余数存入项目基金。

handler RECEIVEREWARD(p, r, k)
  $n \leftarrow |p.\text{contributors}|$
  $q \leftarrow r \mathbin{/\!\!/} n$
  $m \leftarrow r \mod n$
  $x \leftarrow \{(c.\text{addr}, q) \mid c \leftarrow p.\text{contributors}\}$
  return $x \cup \{(p.\text{fund}, m)\}$

这让潜在的贡献者能十分容易地看清他们是否会因自己的贡献而得到奖励，以及奖励的份额。

5.2.2. *转账与捐献.* 当收到一笔不是来自于财政系统的 OSCOIN 转账时，RECEIVETRANSFER 处理程序就会被调用。这个程序需要与 RECEIVEREWARD 一样的三个参数，以及一个额外的参数 s，来表示转账的发起者或来源账户。

handler RECEIVETRANSFER(p, r, k, s)
  return $\{(p.\text{fund}, r)\}$

与 RECEIVEREWARD 的例子类似，这个处理程序内也可以包括任意逻辑。例如，因为对项目的捐献将调用这个处理程序，所以这个项目可以靠这个程序将每个捐款一部分分给项目贡献者，剩下的给维护者。

5.3. **合约处理程序的更新.** 更新项目智能合约需要使用

$$\mathsf{updatecontract}(P_a, h, c, \nu, v)$$

转账。此类转账必须由项目维护者签名。参数中的 h 代表要被更新的处理程序，c 代表新处理程序的代码，v 是一组投票：维护者为此次升级收集道德一组签名，v 是防止投票重放攻击而设置的随机数。

更新合约可能影响既有的贡献者，比如，可能更改他们工作收到的奖励的份额。因此，决定了哪些处理程序可以被更新的规则也存储在项目合约内的 UPDATECONTRACT 处理程序中。更新合约转账的有效条件会在该处理程序中标明。

UPDATECONTRACT 函数需要三个变量：p, 项目数据；h，被更新的处理程序的名字；v，对合约更新转账投票进行了签名的公钥列表。这个函数必须返回一个布尔值，表明该次更新是否有效。该处理程序的缺省代码如下：

handler UPDATECONTRACT(p, h, v)
  return $\{o.\text{addr} \mid o \leftarrow p.\text{maintainers}\} \subseteq v$

在缺省代码中标明，所有现任维护者必须对所有合约更新进行签名。

为了指定合约升级必须被大多数贡献者接收，该处理程序可以被升级为如下形式：

handler UPDATECONTRACT(p, h, v)
  $h \leftarrow |p.\text{contributors}| \mathbin{/\!\!/} 2$
  $v' \leftarrow \{c.\text{addr} \mid c \leftarrow p.\text{contributors}\} \cap v$
  return $|v'| > 1 + h$

如果处理程序返回 $\top$，合约升级转账就被认为是有效的，转账中指定的处理程序 h 就被升级为新代码 c。

5.4. **特定花费.** 虽然大部分 OSCOIN 的分配能以 RECEIVEREWARD 和 RECEIVETRANSFER 处理程序的形式自动执行，但有时仍有手动将 OSCOIN 从项目基金转移到另一个账户的需要。例如，在项目收到一笔大额捐赠的情况下，维护者们为了将来的组织花费，可能会想将一定量的资金转为储备。

为了以特定方式将项目资金花掉，转账需要一组额外的参数。

$$\mathsf{transfer}(P_a, a, n, \nu, v) \mid n \in \mathbb{N}_{\geq 1},$$

在这里，P_a 是支出项目资金的账户，a 是接收资金的账户，n 是 OSCOIN 的数量，剩下的两个参数是一组投票。

转账的验证由 SENDTRANSFER 处理程序执行。该处理程序与 UPDATECONTRACT 的工作方式类似。例如，函数可能指定项目维护者可以在无需社区同意的情况下每月花费一小笔数目为 x 的 OSCOIN，而大额花费需要超过半数的贡献者和捐赠者同意。

handler SENDTRANSFER(p, n, a, v)
  if spentThisEpoch$(p) + n > x$ **then**
    $h \leftarrow |p.\text{contributors} \cup p.\text{donors}| \mathbin{/\!\!/} 2$
    $c' \leftarrow \{c.\text{addr} \mid c \leftarrow p.\text{contributors}\}$
    $d' \leftarrow \{d.\text{addr} \mid d \leftarrow p.\text{donors}\}$
    $v' \leftarrow v \cap c' \cap d'$
    return $|v'| > 1 + h$
  else
    return $\top$

利用得当的话，智能合约就是社区的强大利器，这种强大体现在：

(1) 通过自动，明确的奖励分配与执行方式，为项目构建内在信任，实现透明性。
(2) 可以让贡献者们在提交贡献前就对项目产生认知。
(3) 可以以较低的风险移交项目以及 OSCOIN 给新的维护者
(4) 以社区分权的方式，为项目成员和协作者提供了稳定性，使得作恶更加困难。

6. 应用

Oscoin 平台在促进软件社区的新应用诞生方面颇有潜力。相较于以太坊这类应用不可知的虚拟机来说，Oscoin 围绕着开源软件的协作流提供了基元设计，让开发者们可以在此之上进行更灵活的组合操作。当然我们无法穷举出 Oscoin 所有的应用案例，但下面列出的几个场景无疑从 Oscoin 平台中受益：

6.1. **治理和集体决策.** 我们相信 oscoin 的智能合约可以帮助开发者们做出集体决策。因为 oscoin 平台的网络图能够梳理各个项目参与者之间的利益往来并有效激励参与行为。其次，团队内也可以配合其他决策工具使用智能合约来进行争端解决。

6.2. **最小化信任的软件开发过程.** 贡献历史被记录上链会催生新的软件开发模式，这种模式几乎无须网络参与者们彼此建立信任。比起原有的须依赖第三方中介的脆弱的信任模式，oscoin 用密码学保证代码提交可信性的方式更能为资源的整合赋能。oscoin 平台的这类特性格外适合对安全需求有要求的软件开发项目，比如航空航天应用和生物医学固件项目，也包括如 oscoin 这类加密货币项目。

6.3. **激励.** 我们认为第三类小有前景的应用方向是为开发者提供激励。在 oscoin 网络内，这些应用模式可能包括付酬给开发者，为项目之间清算服务协议和依赖关系，或者是利用数字货币及抵押资产创建新的众筹模式，如 Patreon。

6.4. **去中心化自治组织.** 最后，如果将以上思路与 oscoin 财政系统 (§2.2) 结合起来，那么开源社区就可以逐步衍化成一个源源不断的接受 oscoin 资助的去中心化自治组织（DAOs）。这样网络内外的所有贡献者都能享受应得的激励，组织内的决策过程将完全透明，开源软件社区运动秉承的真正自由的精神得以永驻。

7. 未来的研究

在准确而清晰地勾勒 OSCOIN 协议的愿景时，我们暂且为一些问题的答案留白。我们将在下面对这些问题进行简短陈述，并准备在未来的工作中逐步解决。

- 之前在第三章已经强调过信息可信性的问题。我们希望能纳入以预言机为基础的解决方案，并同时鼓励用户们把可疑的项目标记出来。为进一步抑制项目的作恶动机，当检测到项目的不诚实行为时，协议将扣除他们未锁仓的奖励余额作为惩罚。
- 在第三章中，我们用 OSRANK 算法提出了分两步为项目进行排名，第一阶段中选种子集合 将作为顶点集，目的是为了摆脱女巫攻击可能带来的困扰。然而，集合如何选出并随轮更新，将作为开放问题以待后续研究。
- 第二章讨论中提到了我们将继续探索能保障网络安全的 proof-of-work 共识的候选机制
- OSRANK 算法中比较重型的工作是对算法各种参数的实验和调整，比如边的权重，阻尼系数，阈值等等，这些实验和调整将持续下去。
- 当前来看，OSRANK 是衡量软件代码库相对价值的一个很棒的指标，但对于使用专有设置的面向用户的应用和软件，OSRANK 就无法给出有力的变现指标。我们还在寻找能让 OSRANK 适用类似场景的方案，这样就可以服务更多的开源项目

致谢

感谢 Sam Hart 帮助解决本文中的一些核心问题，谢谢他给出的宝贵想法与意见。感谢 Aaron Levin 在最初构思 OSCOIN 时发人深省的一些讨论和做出的贡献。最后由衷感谢 Monadic 团队和我们的同事，你们对这项工作给予的支持与反馈是无价的。

References

[1] Nakamoto, Satoshi. Bitcoin: A Peer-to-Peer Electronic Cash System. May 2009
[2] Wood, Gavin. Ethereum: A Secure Decentralised Generalised Transaction Ledger. December 2018
[3] Protocol Labs. Filecoin: A Decentralized Storage Network. July 2017
[4] Eghbal, Nadia. Roads and Bridges. The Unseen labor behind our digital infrastructure. July 2016.
[5] Brin, S.; Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine (PDF). *Computer Networks and ISDN Systems.* 30: 107–117.
[6] Cheng, A. and Friedman, E. 2006. Manipulability of PageRank under Sybil strategies. In First Workshop on the Economics of Networked Systems (NetEcon06).
[7] Z. Gyöngyi, H. Garcia-Molina, J. Pedersen: Combating Web Spam with TrustRank
[8] Bahmani, Bahman and Chowdhury, Abdur and Goel, Ashish. Fast Incremental and Personalized PageRank. Proc. VLDB Endow. December 2010.
[9] Dolstra, E., de Jonge, M. and Visser, E. Nix: A Safe and Policy-Free System for Software Deployment. In Damon, L. (Ed.), 18th Large Installation System Administration Conference (LISA '04), pages 79–92, Atlanta, Georgia, USA. USENIX, November 2004.