一种基于权益的区块链投票共识协议

Yj1190590*†

摘要. 本文介绍了一种基于权益的区块链投票共识协议，在协议中，我们使用投票聚集权益的方法，弥补了权益证明的主要缺陷。文章通过对投票、竞争和主链判定过程的描述介绍了创建一种低开销、结构简单、完全客观而且具有确定最终性的权益证明机制的大致实现方法。同时还介绍了这种设计的其它特性，例如在去中心化的程序开发和可扩展性方面的进步。

关键词

1.权益聚集证明(Proof of Accumulated Stakes). 2. 投票.
3. 最终性. 4. 收益分配

1. 引言

区块链共识协议发展至今，大致可以分为 PoW（工作量证明）、PoS（权益证明）和 BFT（拜占庭容错）几个体系。本文将主要针对 PoS 体系中的问题进行探讨，因为对区块链的最原始和最重要的应用领域——虚拟货币来说，PoS 体系虽然没有 BFT 的高通量、低手续费等特性，但是在代码复杂度、去中心化程度和客观性上更具优势，更加适合承载对安全性和容错性要求极高的货币功能。只要解决了自身的两个最大的问题：NaS（Nothing at Stake）问题和财富分配集中化问题，相对于消耗大量能源和矿池越来越中心化的 PoW，PoS 都具有极大的优势，取代 PoW 将成为必然的趋势。

NaS 问题作为 PoS 诞生之日起便存在的问题被很多人认为是无法避免的，因为相对于 PoW 模式，PoS 的矿工在争取报酬时不需要付出高昂的电力成本。但是，由于 PoS 的竞争资源——权益的总量是固定不变的，利用这个属性，我们不仅可以确定出唯一的主链，而且还以一种非常便捷的方式拥有了 PoW 不具备的最终性。这让我们能够解决 NaS 引起的的其中一个主要问题，即历史攻击[1]的问题；而 NaS 导致的另一个多重投票[2]的问题，在协议中也通过公开投票的机制得以解决。

财富分配的中心化倾向也是 PoS 与生俱来的问题。按照 PoS 的逻辑，当付出相同的劳动时间后，富人会获得更多，所以富人倾向于花更多的时间工作而穷人则相反，导致富人更富，穷人更穷，这个过程会不断加速，最后成为少数人的“贵族游戏”。而且由于财富分布遵从了 Pareto 法则，即少数人掌握着大部分的财富，上述“劫贫济富”的程度会比想象中更加严重。我们通过一种聚集权益的方法来降低数量众多的低权益持有者工作的难度，在不降低整体竞争强度的情况下保证更多的人参与到财富分配中来，使这个问题得以解决。

聚集权益指的是，参与者通过主动或被动的方式将权益指派给少部分人，让他们代替自己进行工作，达到让多数人活跃起来的目的。聚集权益的方式可以有多种，其中一种利用“网络分散度”属性的方式在某些状况下（交易量足够大，参与人数足够多）时，可以作为主要手段，因为其客观性和易用性优于其它方式。关于网络分散度的概念，需要考虑以下事实：因为网络延迟的缘故，分散在网络中的终端越多，它们共同收集网络中随机散布的信息的速度越快。我们把这种分散的程度称为“网络分散度(Network Dispersity)”，利用网络分散能力来聚集权益的方式，可以提供一种不可模拟的竞争机制。

我们把这种利用权益和聚集达成的共识称为 PoAS（Proof of Accumulated Stakes）。

†Yj1190590 (3171228@qq.com)

2. 场景和角色

整个网络可以想象为投票和竞选的场景，节点按照分工不同有以下三种角色

权益人(Stakeholder)—权益人作为货币的拥有者，是网络中每笔交易的主体，因此每笔交易的广播都由一个权益人发起。权益人节点负责响应第一个向它发起“拉票”请求的采集者，然后将自己的投票结果打包到交易结构并发布到网络。权益人的权益视为“选票”，获得选票的多少影响到矿工的竞争力。

矿工 (Miner)—矿工节点通常拥有自己的采集者为他们收集选票，使用选票参加出块权竞争，此外还要验证交易、区块等。矿工还拥有对主链的投票权。

采集者(Gatherer，可选)—附属于各个矿工节点，负责侦测附近的权益人节点，收集尽可能多的选票。采集者多数是以嵌入 App 客户端或网页源代码的形式运行于网络终端。

网络结构(有采集者)如下图所示:

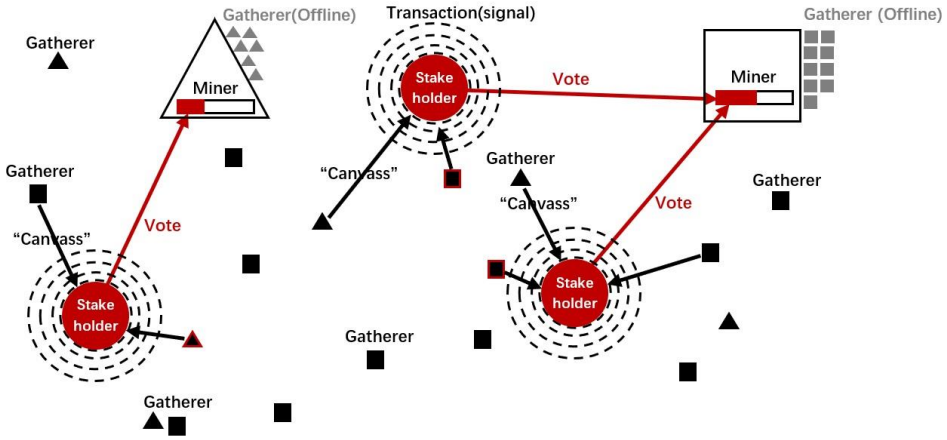


图 1. 节点及网络结构示意图

如上图所示，采集者更多的矿工，有更高的几率获得选票（权益）。[3]

3. 共识过程

我们依靠两种投票来达成共识，分别为权益人对矿工的投票，和矿工节点对当前主链的投票。下面分别进行介绍。

3.1.1 对矿工的投票

对矿工的投票目的是聚集权益，特别是分散在小份额权益人手中的权益。投票阶段的具体步骤如下：

(1) （可选）每笔交易发布之前，权益人节点发出一个广播信号，附近的采集者收到广播信号后向权益人节点发出“拉票”请求，提供所属的矿工节点地址；

(2) 收到第一个请求后（如果有第(1)步），权益人将矿工节点地址写进交易结构，广播本次交易。

投票的权益会用作两个用途：出块权的竞争和主链的确认。因此在计票时，将这两种用途的权益分开统计，记为 x 权益（出块权）和 y 权益（主链），两种权益在每个账户初始状态时都等于货币数量。计票阶段的具体步骤如下：

(1) 把过去一个投票周期内（比如 6000 个区块）的区块中每个权益人的 x 权益和 y 权益分配给它们所投票的矿工节点，如果权益人在一个块有多次投票，则均分到每一次投票；

(2) 把过去一个投票周期内的所有交易的集合记为 T，从 T 中找到每个矿工相关的交易，从中抛弃最后一次出块之前的部分，收集剩余的部分，记为集合 T'，T'⊆T；[4]

(3) 通过统计集合 T'中的矿工节点在它们在第(1)步中获得的 x 权益，得到各个矿工所获得的 x 投票权益数，记为集合 X；

(4) 通过统计集合 T 中的矿工节点在它们在第(1)步中获得的 y 权益，得到各个矿工所获得的 y 投票权益数，记为集合 Y。

按照以上步骤，权益通过投票集中到了矿工的手里，计票结果为集合 X、Y。对于任意时刻，比如区块 a 位置的计票，是指从 a 往前一个投票周期内的统计结果，记为 X_a 和 Y_a。

为了控制投票频率，需要对账户所拥有的投票能力（x 权益和 y 权益）分别进行调节。对 x 权益，以每个账户的投票间隔作为调节系数，权益从 0 开始，每当间隔增加一个时间单位 t（比如 10 个区块），就增大一定比例，间隔增加到一个投票周期（比如 6000 个区块）后权益达到到最大值；对 y 权益，上述时间单位 t 等于投票周期，即一个投票周期间隔内 y 权益只生效一次。每一笔交易的货币对应的 x、y 权益的大小，也随转账间隔线性递增，直到一个投票周期。

为了提高投票的灵活性，我们应该有投票交易，这种情况不需要附加交易信息。但是这样的投票交易没有交易费奖励给矿工，我们需要提供一种方法来作为激励。我们将块中所有纯投票交易的有效权益作为参数，用来调整下一次出块的计算周期，权益越高，计算周期越短，否则越长，比如让计算周期在 0.9 秒-1.1 秒之间浮动，这样会直接影响下一次出块的速度。如此一来，打包尽可能多的交易才是更好的选择。该参数每隔一段时间应该根据平均值做一次调整。

3.1.2 出块权竞争

竞争的主要目的是决定由哪个矿工来创建区块，所有的工作由矿工节点来完成：

(1) 矿工节点周期进行一个基于常量（时间戳、个人签名等）的数学运算，期望结果达到某个目标，满足出块要求，记为：hashProof()<target*d*effective(x)，其中 target 是目标， d 是难度调节参数[5]，x 是当前矿工所获得的 x 权益数（x∈X）,effective()函数计算出 x 的有效部分，将在 3.2.2 小节中详细介绍；

(2) 当达到出块要求后，矿工节点将他收到的交易打包并生成区块，发布到网上。同时打包的还有各节点收益和其他验证参数。挖矿所得的收益由矿工和所有参与的权益人按比例分配。

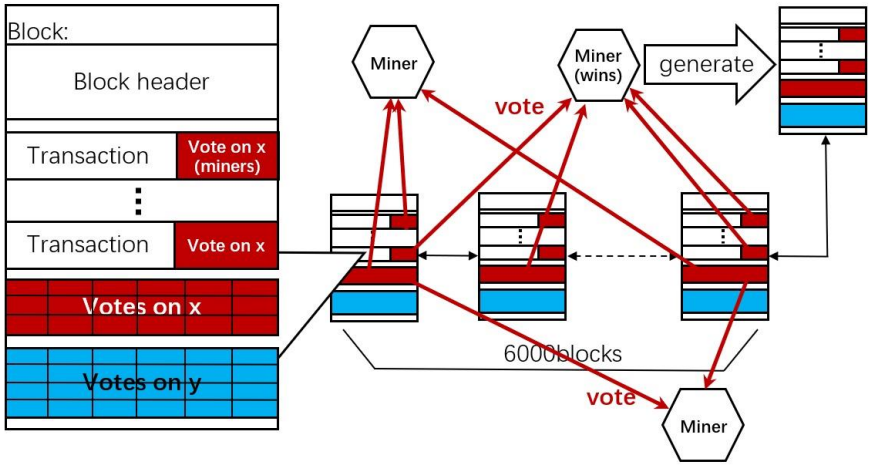


图 2. 对矿工投票生效过程

如图所示，在已产生的若干区块中统计选票，获得选票（权益）最多的矿工节点将有最高的几率赢得出块权。

3.2.1 对主链的投票

在 PoAS 中，分支的优先级不是取决于它们的长度，而是取决于它们的“重量”，而“重量”在这里并不等于区块个数，而是等于区块所获得的投票数。在这里我们参考 GHOST 协议的描述，定义分支的重量为“分支的根节点的所有子孙节点所获得的投票之和”。

为了确定出主链，矿工节点周期性（例如 60 个区块，为了区别另一种投票周期，记为“周期 2”）地对当前的主链顶端区块地址签名并发布到网络，此数据就是对主链的投票，将会和交易一样被打包进区块。对于这些投票，生成区块的矿工会尽可能地收集到区块中。矿工收集这些投票信息是符合自身利益的，因为收集当前分支的投票可以增加分支的重量；而收集其它分支的投票可以减少其对应的矿工在当前分支的竞争力，这一点会在下一节做解释。对于在其它分支生成的区块，矿工也会尽力的在区块中去引用它们（区块头），这也是符合自身利益的，因为这样做会极大地降低这些块的生成者在本分支的竞争力，这一点也会在下一节做解释。这样一来，每个分支的区块中实际上包含了所有分支的节点和投票情况，每条链只需要从自身的视角统计一次就可以了。

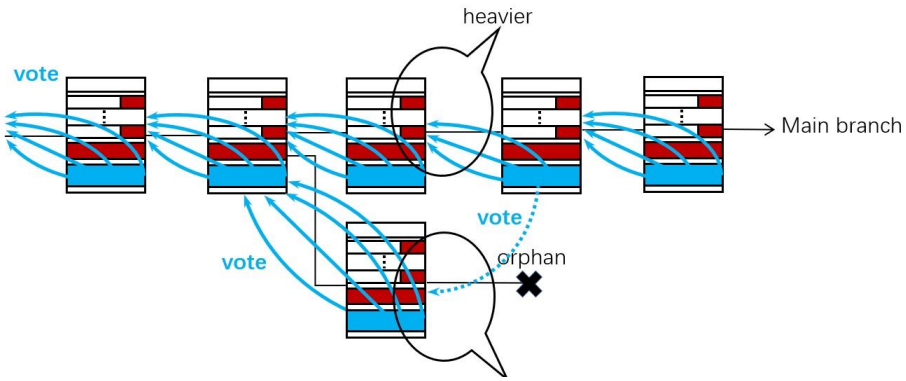


图 3. 主链投票生效过程

如图所示，矿工节点会对所有分支进行投票，并记录在区块中，这些记录决定了各个分支所获得的选票（权益）多少，也就决定了分支的重量，最重的分支即为主链。

假设设要计算的分支段根部为 c，d 是 c 的一个子孙，且 c→d 这条链长度小于一个投票周期 2，我们要计算在 c→d 的视角下这段分支的重量，计算方法如下：

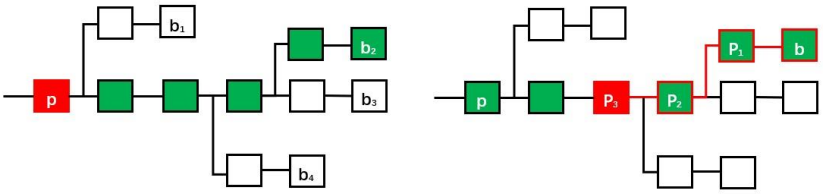
(1) 首先统计 Y_d，然后将 Y_d 中各个矿工的 y 权益平均分配到他们在 c→d 中记录的每个投票；

(2) 统计其中指向 c 的子孙的投票分得的 y 权益，最后将结果求和，即为此段分支的重量。

如果 c→d 的长度超过一个投票周期 2，需要分成以投票周期 2 为单位的小段，再分段统计。根据以上计算方法，每个矿工对任意一段小于投票周期 2 的分支都只能投票一次。由此一来，如果分支在一个投票周期 2 之内获得了整个系统总权益一半以上的重量，那它就不可能存在竞争分支，我们称这个分支被“确定（Finalized）”了下来，分支根部的区块称为“保存点（Save Point）”，保存点之前的所有区块都是不可替换的，保存点之后的所有块都必须是它的子孙。创世块是第一个保存点，其余保存点都在前一个保存点的基础之上建立，方法如下：

(1) 假设 p 是最新的保存点，b_i 是最新产生的区块而且都是 p 的子孙，两两比较 b_i 的优先级来确定当前主链[6]；

(2) 假设新的主链头部区块为 b，p_j 为 b 的祖先且 p_j→b 的长度小于一个投票周期 2，依次计算 p_j 作为根节点的分支的重量，当重量超过总权益的 2/3（保证适当容错）时，p_j 成为新的保存点，令 $p=p_j$，返回第(1)步。



(1) (2)

图 4. 主链确定过程(1)和保存点生成过程(2)

3.2.2 激励与惩罚

为了保证每条链上视角的基本一致，需要激励矿工积极的引用所有分支的投票和区块。要达到这个目的，我们采取一些激励和惩罚措施：

(1) 如果一个矿工在其它分支生成了区块并被引用，那么视为这个矿工也在本分支也生成了区块，他的 x 权益的统计立即清零。换句话说，只要矿工生成了区块，他在所有分支都暂时不能参加出块权的竞争，这对于其它矿工来说相当于一种激励，保证了所有区块都能会尽可能的被引用。

(2) 在 3.1.2 节中用到的 effective()函数，实际上是为了激励矿工进行投票，而且尽量要投给正确的分支。它是这样工作的：在当前链最顶端一个投票周期 2 的长度内将当前矿工的 x 权益平均分配到他的每一次投票中，然后统计其中指向当前链的投票分得的权益，即为 effective(x)。简单的说，effective()函数让矿工的出块能力与他们的正确投票成正比。这样一来，记录其他矿工的错误投票就会降低别人的出块能力，这对矿工来说是一种激励，保证所有的错误投票都能尽可能多的被打包进区块中。而对正确的投票，如 3.2.1 节中提到的那样，可以增加当前分支的竞争力，同样也能保证尽可能的被记录。

(3) 如果矿工为了保证投票的正确性，每次投票都投向之前的相对成熟的区块的话，对当前主链的认定就会造成影响。对于这种情况，我们规定：凡是指向当前链的投票，都必须指向顶端区块，否则不能成为有效记录。

3.3 小结

按照以上共识流程，持有权益的用户可以把竞争出块的工作交给矿工们去完成，自己要做的只是在规定周期内简单投票即可。对于低权益的用户来说，不用花太多的成本就能保证所持权益不缺席投票活动，获得相应的回报，这样就能基本避免前面所说的财富集中化的问题；对于高权益的用户来说，不用随时保持在线状态也意味着更高的账户安全性。而且由于降低了权益持有人参与网络维护的成本，我们就可以减少挖矿奖励，避免严重的通货膨胀。

对于矿工的多重投票的问题，由于投票都在统计之前已经保存在了区块中，任何人做过的选择都会被公开，因此不会存在隐藏的竞争分支，无法进行多重投票。对于构造全新分支的历史攻击，我们提供了两层保护。第一个是“保存点”，凡是在保存点之前出现的历史攻击都会被拒绝掉；一个是分支优先判定策略，PoAS 的分支是以所获得的投票多少来决定优先级的，在线权益较高的分支一定较重，所以除非攻击者掌握的历史权益超过当前主链的全部在线权益，否则无法成功。由此目前已知的 NaS 问题都得到解决。

另外，矿工节点加采集者的结构可以在普通 App 和网页开发中进行嵌入，这些程序当中很多都能使用网络分散度来收集权益，成为矿工，进入门槛很低，保证了系统的可持续性。

4. 争夺权益人

为了拥有更强的出块能力，矿工会尽可能地收集到更多的权益，除了通过网络分散度的能力以外，一些矿工也可能通过其它方式与权益人合作。我们在交易结构中提供三种投票的方式：A.以网络分散度的形式接受采集者拉票； B.指定一个矿工，直接让他赢得选票； C.指定一个属于自己的矿工，独自工作并获得全部收入。在本质上这三种矿工是利用在三种不同方式下获取的权益资源对出块权进行竞争，再加上动态调整机制的引入，平衡三种方式的竞争力，攻击者通过统治其中一种能力就控制网络的风险就会降低。

4.1 钱包应用

因为钱包应用密切的参与到共识活动之中，所以钱包开发者能以多种方式从系统获得到回报。这会有两个好处：激励开发团队开发优秀的钱包程序，这在客户端的开发上向去中心化前进了一步；或者更重要的，激励他们开发侧链项目，这为解决扩展性问题提供了帮助。

但是基于自身利益的考虑，钱包应用只会要求用户选择 B 类交易指定他们自己的矿工，这样就没有足够的空间留给使用网络分散度进行竞争的矿工了。因此我们需要有一个机制来鼓励钱包应用接受 A 类交易：给 A 类交易增加钱包账户字段，把一部分收入分配给钱包应用的开发者，而 B 类交易则没有这种奖励。

5. 收益分配

挖矿所得的收入如果分配比例不同，用户在利益驱使下的选择会造成网络结构的变化。我们可以通过动态调整分配比例来平衡用户的分布。总的收益分为两部分：手续费和挖矿奖励，下面分别进行解释。

5.1 手续费

用户需要为每笔交易支付一定的手续费，以弥补矿工在记录和执行此交易时所消耗的资源。在 PoAS 中不像比特币那样由出块矿工直接获得生成块中所有交易的手续

费，而是支付给这些交易所投票的矿工，当他们成功出块后获得。但是出于安全性和鼓励更高手续费的目的，我们将手续费的一小部分，比如 10%，奖励给生成当前块的矿工。

手续费作为收入分配时，矿工和钱包按固定的比例进行分配，比如矿工 29%，钱包 31%[7]，剩余部分由参与出块的所有 A 类和 B 类交易的权益人分享。考虑到权益人可能数量很多的情况，我们将收入分为几份，分多次在权益人中进行抽奖，权益人的权益占比与赢得抽奖的几率相同。

5.2 挖矿奖励

为了激励更多高权益的用户参与维护，系统会在手续费之外额外再增发一部分的货币作为挖矿奖励。因为 C 类挖矿的权益人是无法获得手续费的，但是能获得全部的挖矿奖励，所以挖矿奖励的多少可以影响权益持有人参与 C 类挖矿的积极性。而 C 类挖矿代表了利用权益直接挖矿的竞争者，是对平衡性重要的调节砝码，所以奖励的数值需要根据当前参与比例动态计算得出。如果 C 类挖矿的参与程度低于预期值，则增加挖矿奖励，反之降低。比如在矿工和钱包总共分成挖矿奖励的 30%和手续费的 60%，且参与比较均衡的情况下，挖矿奖励应该为手续费 1.2 倍左右[8]。

挖矿奖励数值是重要的平衡参数，它能够调节 C 类挖矿活动的参与人数，削弱用户聚集能力的统治力，防止出现中心化的趋势。

挖矿奖励由矿工和钱包按固定比例进行分配，比如矿工 14%，钱包 16%[7]，剩余部分由参与出块的所有权益人按比例分享。对于 C 类交易的权益人，挖矿奖励全部由他们的矿工获得；对于 A 类和 B 类交易的权益人，和手续费的分配一样，按照权益占比在权益人中进行抽奖。

6. 作弊和攻击

(1) 模拟采集者，即试图通过模拟创建大量的采集者节点在投票网络中接入许多虚拟节点，从而提高成功拉票的几率

为了应对此情形，我们可以在创建 p2p 连接的时候加以控制，比如每个客户端都只与和自己响应速度最快的前若干个节点建立连接即可。

(2) 超级采集者，如果有采集者节点处于靠近骨干网络的位置，它会因为更低的延迟而获得更多的投票，如果大量的矿工都使用这种超级节点，依靠网络终端数量来竞争的矿工就会失去竞争力

虽然我们可以在投票网络中加入各种防范措施，而且这类攻击的投入产出比例并不好，但是这种可能性始终存在，这也可以看作对网络分散度公平性的一个威胁。尽管如此，这对共识机制本身来说，并没有造成很大影响，因为网络延迟和带宽作为一种能力同样也能带来公平的竞争，而且不会消耗过多的社会资源，这并没有违背我们的设计初衷。

(3) 51%攻击

如果某个用户掌握了 50%以上的权益，系统就很容易被其攻击，这点和普通 PoS 协议是相同的。但是由于 PoS 系统下的用户必须运行全节点并且保持在线才能参与出块和维护网络，所以实际上权益无法达到很高的在线率，使安全性打了折扣。但在 PoAS 系统中降低了用户参与的成本，使得在线权益的比例增加，提高了系统的安全性。

7. 结论

本协议相对于 PoS、PoW 和 BFT 协议，具有以下优点:
(1)不存在算力竞赛，避免了高能耗和算力中心化隐患;
(2)不存在 NaS(Nothing at Stake)导致的多重投票和历史攻击等缺陷;
(3)提供足够的竞争强度来保证网络的安全，同时不会陷入财富分配集中化和通货膨胀的问题;
(4)同时具备最终性、客观性和完全的去中心化，而且不需要任何特殊节点和额外开销;
(5)激励钱包开发者，为系统的可持续性和可扩展性打下基础;

利益冲突

本设计已申请专利，专利号：CN201811633256.0

注解

[1] 历史攻击：攻击者从一个历史区块开始创建一条新的分支并试图取代现有分支。

[2] 多重投票问题：矿工的一种在每个分支上都投票的策略。

[3] 理论上获胜的几率和在线采集者数成正比。

[4] 矿工出块后，获得的 x 权益清空，这样能防止 stake grinding 类的攻击；但 y 权益不变，始终保持一个投票周期的统计区间。

[5] 由于本协议对当前分支上的活动权益可以有精确的客观统计，所以对难度的调节不用依赖于近期的出块速度，可以直接根据活动权益的大小对参数 d 进行设置。

[6] 比较两个叶子节点的优先级，需要从它们分叉的位置，即最后一个共同的祖先开始，分别统计分叉之后的两个分支在各自视角下的重量，较重的一个优先级较高。

[7] 为了鼓励钱包应用少参与挖矿竞争，需要保证钱包的收益略大于挖矿的收益。

[8] 用户在用 C 类交易单独挖矿时不会被矿工和钱包提取分成，但会失去获得手续费的机会，挖矿奖励越高，用户选择 C 类挖矿的优势越明显，因此通过控制挖矿奖励的多少，就能调节选择 C 类挖矿的用户的比例。假设矿工和钱包总共分成挖矿奖励的 30%和手续费的 60%，手续费为 f，挖矿奖励为 F，要使一个用户在 A、C 两类挖矿时的收入相当，则有 f×90%×(1－60%)+F×(1－30%)=F,得出 F/f=1.2。

参考文献

[1] Yj1190590 (/yj1190590). “PoND(Proof of Network Dispersity) BlockChain Project.” Github (accessed 29 April 2018)
https://github.com/yj1190590/PoND/blob/master/README.md

[2] Paul Firth. “Proof that Proof of Stake is either extremely vulnerable or totally centralised.” BitcoinTalk.org (accessed 1 March 2016)
https://bitcointalk.org/index.php?topic=1382241.0

[3] Vitalik Buterin. “Long-Range Attacks: The Serious Problem With Adaptive Proof of Work.” blog.ethereum.org (accessed 15 May 2014)
https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work/

[4] Yonatan Sompolinsky and Aviv Zohar. “Secure High-Rate Transaction Processing in Bitcoin” No Publisher (2013) https://eprint.iacr.org/2013/881.pdf

[5] Husam Ibrahim . “A Next-Generation Smart Contract and Decentralized Application Platform” Github (2018) https://github.com/ethereum/wiki/wiki/White-Paper#modified-ghost-implementation