一种基于网络分散度的区块链共识协议

Yj1190590*†

摘要. 本文介绍了一种基于网络终端和传输延迟的区块链共识协议，在协议中，我们构造了一种权益投票的共识机制，既没有高能耗的问题，也避免了权益证明的缺陷。这种新的机制有助于创建一种可扩展的多层区块链结构，有可能是加密货币未来的一种发展方向。此外，利用网络终端的数量和分散程度进行的竞争将存量巨大的移动开发人员作为潜在的矿工，为网络的维护和扩展提供了丰富的储备。

关键词

1.网络分散度证明(PoND). 2. 投票. 3. 收入分配. 4. 个人利益最大化.

1. 引言

因为能耗问题，工作量证明(Proof of Work)模式正越来越被人们所诟病。然而抛开能耗问题，PoW 模式仍然是所有解决方案里面最高效、最稳定和最体现本质的。工作量证明以其简单直接的“能者多得”逻辑完美的解决了开放式分布系统的共识问题，其它可行的全分布式的共识协议几乎都沿用了这个核心逻辑，比如活动证明(Proof of Activity)、燃烧证明(Proof of Burn)、存储证明(Proof of Storage)、消逝时间证明(Proof of Elapsed Time)等等，其中权益证明(Proof of Stake)共识因为不依赖任何外部资源，而且完全抛弃了耗能的“算力”竞赛而被最广泛的接受。因为PoS将核心逻辑从“能者多得”的基础上发展为“富者多得”，使得区块链的记账权竞争只需要在一个个静止的内部状态之间进行，不需要消耗电力。但是PoS共识也存在自己的缺陷，比如“nothing at stake”导致的多重投票、历史攻击等等，其中一个无法避免的缺陷，就源于它的核心逻辑“富者多得”：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”。而且由于财富分配遵从 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。尽管有的PoS协议实现了完全公平的利息制度，取代了挖矿活动，但这样的协议降低了所有用户参与网络维护的积极性，容易受到攻击。

本文的目的，是想找出另一种不耗电力的能力证明协议替换权益证明，基于一个简单的事实：因为网络延迟的缘故，分散在网络中的终端越多，他们共同收集网络中随机散布的信息的速度越快。而这也是一种“能力”的体现，我们称之为“网络分散度”，这是一种无法通过提高单机的性能来提高的能力，因而避免了能耗问题。权益属性由于其无法复制的特性，在协议中作为度量单位起到重要的作用。持有权益的用户也可以通过权益来参与网络的维护，用户可以自由的选择用权益还是用能力来争夺记账权，用利益分配法则来维持矿力的平衡，最大程度的保证公平性和安全性。

本文中讨论的所有情形均是针对完全开放式的分布系统，故对 PBFT、DPoS、Ripple 等协议都不在讨论和比较的范围之内。

2. 场景和角色

整个网络可以想象为拉票竞选的场景，节点按照分工不同有以下三种角色

†Yj1190590 (3171228@qq.com)

投票节点(Voter)—网络中每笔交易的广播源节点即为一个投票节点，负责响应第一个向它发起“拉票”请求的工蜂，然后将自己的投票结果打包到交易结构并发布到网络。投票节点的权益视为“选票”，获得选票的多少影响到矿工的竞争力。投票节点还拥有主链的投票权。任何用户都可以成为投票节点。

矿工节点(Miner)—矿工节点拥有自己的工蜂为他们收集选票，使用选票参加出块权竞争，此外还要验证交易、区块等。任何用户都可以成为矿工节点。

工蜂节点(Worker)—附属于各个矿工节点，负责侦测附近的投票节点，同时尽快发出拉票请求。工蜂节点多数是以嵌入 App 客户端或网页源代码的形式运行于网络终端。

网络结构如下图所示:

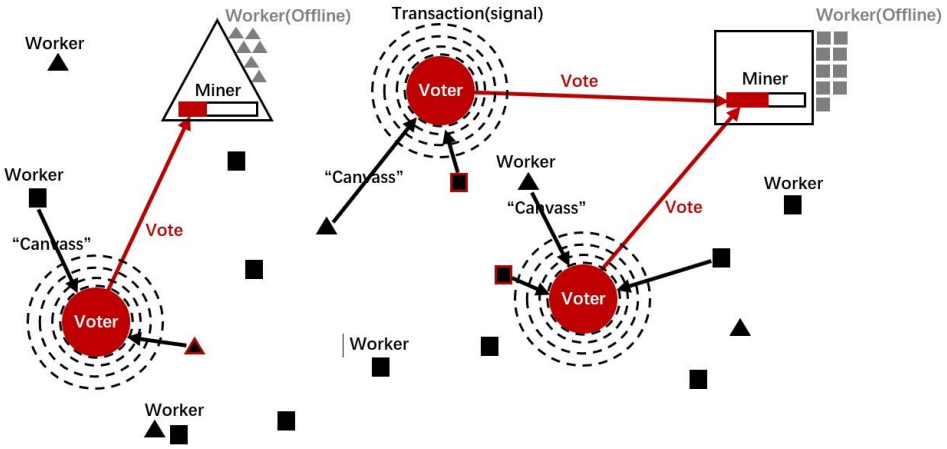


图 1. 节点及网络结构示意图

如上图所示，工蜂节点更多的矿工，有更高的几率获得选票（权益）。[1]

3. 共识过程

共识过程分为以下四个阶段。

3.1 投票阶段

我们依靠两种投票来达成共识，分别为投票节点对矿工出块权的投票（记为 x 投票），和矿工节点代表投票节点对当前主链的投票（记为 y 投票），具体步骤如下：

(1) 每笔交易发布之前，投票节点发出一个广播信号，附近的工蜂节点收到广播信号后向投票节点发出“拉票”请求；

(2) 收到第一个请求后，投票节点会和此工蜂节点进行简短通信，工蜂节点将与此同时向所属的矿工节点（记为 m）处获得的主链顶端区块（记为 b）以及矿工的签名信息发送给投票节点；

(3) 投票节点将区块哈希 b、矿工账户地址 m 以及签名信息写进交易结构，广播本次交易；

(4) 此交易被某个矿工确认打包到新的区块，并发布到网络。

按照以上步骤，每次交易都进行了两种投票[2]。为了控制投票频率，需要对账户所拥有的投票能力（记为 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，大约 100 个小时）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。同样每一个 UTXO 也加入 x、y 两种权益值属性，以转账间隔作为调节系数，调节投票权益大小，方法和参数都与相应的账户权益调节过程相同。

3.2 计票阶段

在计票阶段，我们把两种投票分开统计，计票阶段步骤如下：

(1) 把过去一个投票周期内的区块中每个投票节点的 x 权益和 y 权益平均分配到它们各自在块中的每条交易；

(2) 把过去一个投票周期内的所有交易的集合记为 T，检查集合 T 中交易的 b 字段，统计所有 `b == pre_hash`（即所在区块头中的“前一区块”字段）的交易[3]，记为集合 T'，$T' \subseteq T$；

(3) 从 T' 中去除每个矿工最后一次出块之前的相关交易，记为集合 T''，$T'' \subseteq T'$；[4]

(4) 通过统计集合 T'' 中交易的 m 字段和在它们在第(1)步中分配到的 x 权益，得出各个矿工所获得的 x 投票权益数，记为集合 X；

(5) 通过统计集合 T 中交易的 m 字段和在它们在第(1)步中分配到的 y 权益，得出各个矿工所获得的 y 投票权益数，记为集合 Y。

按照以上步骤，通过对每个块中的交易逐一统计，能够得到所有参与竞争的矿工获得的 x 权益数，以及他们所代表的投票节点的 y 权益数，完成计票，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。

下面两张图分别描述了两种投票的工作原理：

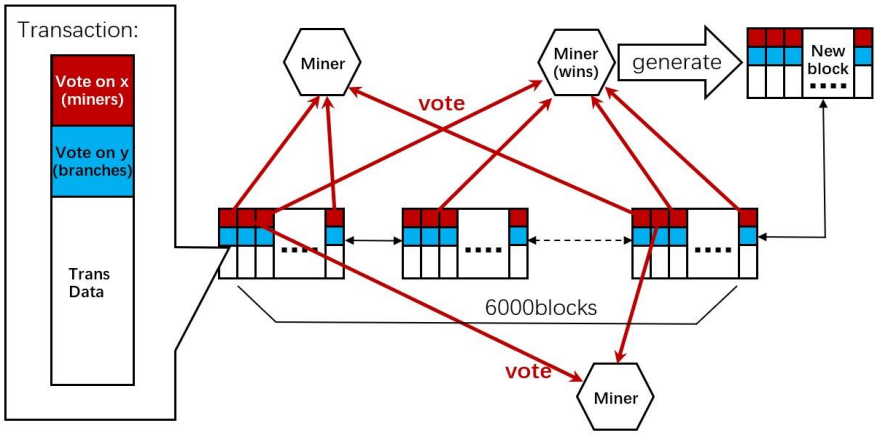


图 2. 出块权投票生效过程

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。

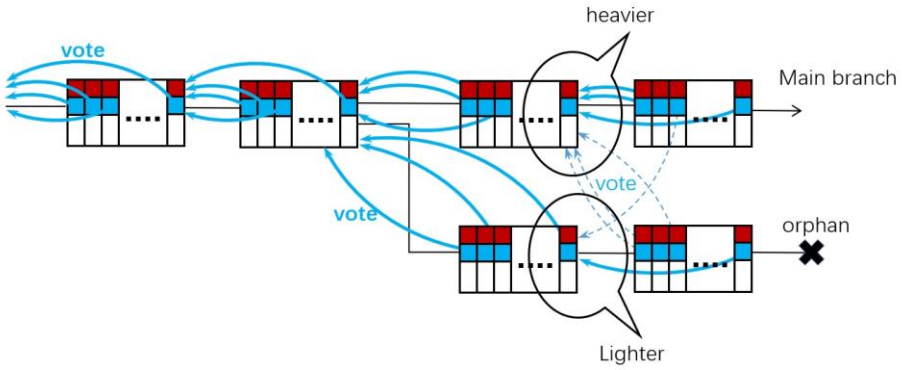


图 3. 主链投票生效过程

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。

3.3 竞争阶段

竞争阶段的主要任务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*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（$x \in X$）；

(2) 当达到出块要求后，矿工节点将这段时间收到的交易打包并生成区块，发布到网上。同时打包的还有各个节点收益和其它计算参数(为了节约空间，交易 id 可以在 6000 个区块范围内编码)。挖矿所得的收益由矿工和所有参与的投票节点按比例分配。

3.4 确认阶段

在 PoND 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”[6]，一段没有分叉且长度小于一个投票周期的分支重量计算方法如下：

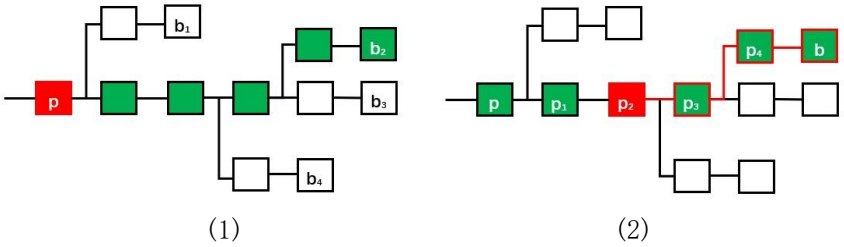
(1) 假设要计算的分支段为 c→d，首先统计 Y_d，然后找到每个矿工在此区间内首次投票的区块，将 Y_d 中各个矿工的 y 权益平均分配到他们在相应区块中的每条相关交易；

(2) 统计其中 b 字段指向 c→d 区间内的交易分得的 y 权益，最后将结果求和，即为 c→d 段分支的重量。

如果分支长度超过一个投票周期，需要分成以投票周期为单位的小段，再分段计算。根据以上计算方法，每个矿工所代表的权益持有者对任意一段小于投票周期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，根据 p→b_i 的重量来确定当前最重的分支为主链；

(2) 假设新的主链头部区块为 b，p_j 为 p 的子孙且 $p_j \rightarrow b$ 的长度小于一个投票周期，依次计算 $p_j \rightarrow b$ 的重量，当 $p_j \rightarrow b$ 的重量超过总权益的一半时，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)

3.5 优点

按照以上共识流程，矿工要想在竞争中获得更大的优势，就必须在物理网络上拥有更多而且更分散的工蜂，以响应随机出现在网络中的投票节点信号。所以此竞争机制无法在单机上进行模拟，避免了无限提高机器性能的装备竞赛，保证了挖矿的公平性。

我们没有在共识过程中借助任何外界的资源，这一点和 PoS 相同。不同的是，由于有矿工的存在，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安全性，换句话说，用冷钱包也能挖矿。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。

矿工节点加工蜂的结构可以在普通 App 和网页开发中进行嵌入，大量的网络程序开发者和应用程序可提供足够的“矿力”储备，转型非常简单，降低了初期的进入门槛，保证了整个系统的可持续性。

4. 争夺投票节点

为了拥有更强的出块能力，矿工会尽可能地争取到更多的投票节点，虽然我们设计的初衷是大家都用更多工蜂来获得选票，但是不可避免的，会有一些用户相互达成协议，以团队的形式合作挖矿。既然如此，我们就在交易结构中提供一个分类选项，你可以选择 A.以广播的形式接受工蜂拉票；或者 B.指定一个矿工，直接让他赢得选票；或者 C.指定一个矿工地址，自己获得挖矿的全部收入。这样做看似影响了公平性，但实际上这三类矿工节点在竞争时，普通矿工节点和带有合作投票节点的 B 类矿工节之间是在聚集用户能力上进行竞赛，而 C 类矿工和前两类之间又是在权益大小上进行竞赛。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得到降低。这样反而对提高公平性和安全性起到了积极的作用。

4.1 钱包应用

因为特殊的用户群体，钱包应用可以引导甚至直接决定交易的类型和投票的目标，基于自身利益的考虑，大部分钱包应用都会选择 B 类方式为自己挖矿，或者优先响应自己的矿工节点，这样的话普通矿工就基本没有机会参加竞争了。因此我们需要有一个机制来鼓励钱包应用多选用 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 类和 C 类则没有这种奖励。由于给钱包应用提供了正当的收入途径，所有影响 A 类竞争公平性的行为都被视为恶意的，因为利益相关，所有的用户和开发者会自发的对此类行为起到监督作用。如果钱包开发者能从系统得到回报的话，会有一个额外的好处：激励开发团队开发优秀的钱包程序，或者更重要的，激励他们开发侧链项目，这样一来为整个系统成为开放式的平台提供了协议层面的激励基础。

4.2 PoS 变种

由此还可以导出一个此共识的变种方式，即只存在 C 类矿工。这种情况下每个人都凭借自身的权益单独挖矿，变成了纯粹的 PoS 模式。这样的变种体系也能保证公平性，而且解决了 PoS 协议下的一些常见的缺陷，当然，除了财富集中化的问题以外。

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。为了激励更多的矿工使用网络分散能力挖矿，PoND 中的手续费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，而是当节点所投票的矿工成功出块后，作为收入对其进行奖励。但是为了激励矿工打包手续费更高的交易，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，比如各 15%，剩余部分由参与出块的所有投票节点分享。考虑到使用网络分散能力挖矿时，投票节点可能数量很多的情况，我们将收入分为几份，分多次在投票节点中进行抽奖，投票节点的权益占比与赢得抽奖的几率相同。

为了保证有足够的手续费作为激励，我们应该确定一个最低手续费标准，但是并非强制收取，而是在矿工打包低于缴费标准的交易时根据常量进行一个结果是 0/1 的随机运算，只有当结果是 1 时才允许包含此交易，计算结果是 1 的几率根据交易的手续费的提高而提高。这样一方面能促使用户支付足够的手续费，另一方面也给予用户自由选择的空间。

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小部分的货币作为挖矿奖励。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，因为 C 类挖矿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等参数动态计算得出，比如在矿工和钱包各分成 15%，且分配比较均衡的情况下，A 类和 C 类挖矿奖励为手续费 2 倍左右。[7]

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。

此外，我们在设定各类挖矿角色之间的分配比例时需要考虑以下的问题：

设以下变量

V.投票节点收入，三类分别为 V_a, V_b, V_c

M.矿工收入，只有 A，B 两类有此项，分别为 M_a, M_b

W.钱包收入，只有 A 类有此项，即 W_a

R.挖矿奖励总收入，即前面提到的挖矿奖励数值，三类挖矿总收入分别为 $R_a=V_a+M_a+W_a$，$R_b=V_b+M_b$，$R_c=V_c$

(1)为了减少钱包应用参与挖矿，需要保证钱包的收益大于钱包参与挖矿的收益，即 $W_a>M_b$

(2)为了防止 B 类矿工和合作投票节点一起伪装成 A 类挖矿的作弊行为，需要将 B 类投票节点的收入始终保持在 A 类投票节点收入之上，即 $V_b>V_a$

(3)为了防止 C 类挖矿伪装成其它两类，C 类的总收入不应小于其它两类收入，即 $R_c=\max\{R_a,R_b\}$。由于 A 类挖矿多了 W_a 一项，收入会大于 B 类挖矿收入，所以简化为 $R_c=R_a$，$R_a>R_b$

总结起来，我们设定的收入分成比例遵循以下原则：$W_a>M_b$，$V_b>V_a$，$R_a>R_b$，$R_c=R_a$。

对于 V 的分配，A 类挖矿中，和手续费的分配一样，按照权益占比在投票节点中进行抽奖；B 类挖矿中，由矿工进行分配。

5.3 侧链中的挖矿奖励

在单个主链-多个侧链的开放系统结构中，增发货币只能在主链中进行，那么侧链中怎样使用相同的奖励规则呢？解决这个问题可以用负利率的方法，来抵消增发的挖矿奖励，即随时间的推移侧链中的每个账户都会按相同比例减少货币，保证侧链中的总货币数量不变。

6. 作弊和攻击

(1)过滤交易，矿工只选取对其有利的交易打包

因为每次出块时包含的交易可以影响之后的竞争环境和主链的选择，所以矿工有可能通过挑选打包的交易试图获得更有利的位置。

首先每个块包含的交易只占统计总量的一小部分，改变一个块的内容并不能对统计结果造成很大的影响。要更好的解决此类问题，需要将块中所有投票节点的总权益作为参数，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，可以减少矿工作恶的动机。该参数每隔一段时间应该根据平均值做一次调整。

(2)模拟工蜂，或者叫“女巫攻击”，即试图通过模拟创建大量的工蜂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。

(3)超级工蜂，如果有工蜂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，如果大量的矿工都开辟自己的超级节点，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，比如限制每个 ip 地址的连接数或者加入人机验证等等，但是在利益的驱使下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。假如不加任何防范措施，到最后要想争取更多的选票，矿工们就需

要在世界各地布置更多的超级工蜂节点，而这也不会消耗过多的社会资源，并没有违背 PoND 的设计初衷。

(4) 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 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 PoND 系统中，用户只需要在规定投票周期内有过一次投票或者交易就能加入竞争，降低了参与的成本，使得在线权益的比例增加，提高了系统的安全性。

(5) NaS(Nothing at Stake)问题

用权益取代高成本的算力之后带来的一个问题就是无代价(Nothing at Stake)问题，因为参与竞争不需要任何代价，用户可能在多个分支同时进行竞争，导致主链安全性降低，即多重投票问题；也可能在历史区块中的某个时间点重新构造出全新的分支，用以替换原有的分支，即历史攻击，比如 Long Range Attack 和 Costless Simulation Attack 都属于历史攻击。

对于多重投票的问题，PoND 的投票都在统计之前已经保存在了区块中，用户做过的选择不能更改，也不会有隐藏的竞争分支，因而无法进行多重投票。

对于历史攻击，PoND 提供了两层保护。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。但是假如攻击是在最新的保存点之后发起的，还有第二层保护：难度调节机制。PoND 的出块难度是严格根据在线权益来计算的（参考注解[5]），在线权益较高的分支出块速度一定较快，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。

7. 结论

本协议相对于 PoS 和 PoW，具有以下优点:

(1) 不存在算力竞赛，节约能耗；
(2) 不存在 NaS(Nothing at Stake)导致的多重投票和历史攻击等缺陷；
(3) 激励足够的竞争强度来维护网络的安全，同时不会陷入财富集中化和通货膨胀的问题；
(4) 具备最终性，而且不需要任何特殊节点和额外开销；
(5) 激励开发者开发扩展项目，为系统的可持续性和可扩展性打下基础；
(6) 新的挖矿方式有丰富的潜在矿工资源作为储备。

利益冲突

本设计已申请专利，专利号：CN2018102289423.

注解

[1] 理论上获胜的几率和在线节点数成正比。

[2] 为了提高效率，可以增加纯投票类型的交易，用户在投票周期之内也可以自己决定当前交易是否参加投票。

[3] 排除选错分支的投票，这样做会使选错分支的投票不能获得出块权竞争上的任何收益，促使矿工在 y 投票时更加谨慎。

[4] 矿工出块后，获得的 x 投票权益清空，但 y 投票权益不变，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以往的出块速度，可以直接根据活动权益的大小对参数 d 进行设置，这样可以有效防止各种历史攻击。

[6] 根据“GHOST”协议，有分叉的分支的重量等于其所有子分支的重量之和，但在实现的时候，可以加以简化，使用类似于 Ethereum 的实现方法：每当有孤块产生后，为了不浪费已经投票给了这些孤块的权益，后续的块会引用这些孤块以增加分支的重量。与以太坊中不同的是，我们并不需要奖励孤块的创建者，因为孤块本身并没有消耗资源，也不需要奖励他们的引用者，因为引用孤块会增加他们自己分支的重量。

[7] 用户在单独挖矿时不会被矿工和钱包提取分成，但会失去获得大量手续费机会，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。假设矿工和钱包各分成 15%，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%+F)*(1-30%)=F,得出 F/f=2.1，即 2 倍左右。

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