# 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开发人员引入进来，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. 利益分配. 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*—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*—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*—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:

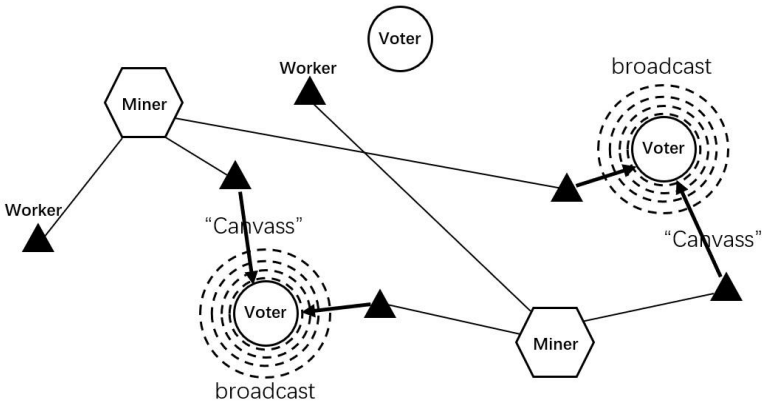


图 1. 节点及网络结构示意图

## 3. 共识过程

(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，投票节点将主链[1]顶端区块地址 b 和工蜂所属矿工节点账户 m 写进交易结构，广播本次交易；
(3) 此交易被某个矿工确认打包到新的区块，并广播到网络；
(4) 矿工节点接收到新块的广播后，验证每个交易的同时，检查交易中的 m 和 b。当 m 指向矿工节点自己时，标记此条交易为 x，当 b 指向当前链头部时，标记为 y；
(5) 矿工节点取得所有投票节点的权益数，并平均分配到它们各自在块中的每条交易；
(6) 逐块累计同时有 xy 标记的交易分得的权益数，记为 X，直到出块为止，最多累计到一个投票周期（假设 6000 个区块）；[2]
(7) 统计当前块中 y 标记的交易分得的权益数，记为 Y；
(8) 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*Y`（其中 `target` 是目标，`d` 是难度调节参数）；[3,4]
(9) 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广播至全网。同时打包的还有所有 xy 标记的交易 id(为了节约空间，可以在 6000 个区块范围内编码)、各个节点收益和其它计算参数，以供验证。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。[5]

为了便于理解，以上步骤可以进行如下简要描述

“用户每次交易都用权益对主链和记账节点分别进行投票，投票结果记录在区块中，通过历史区块统计选票，统计结果用于确定当前的记账权和主链分支。”[6,7]

下图解释了依靠网络延迟竞争出块的原理:

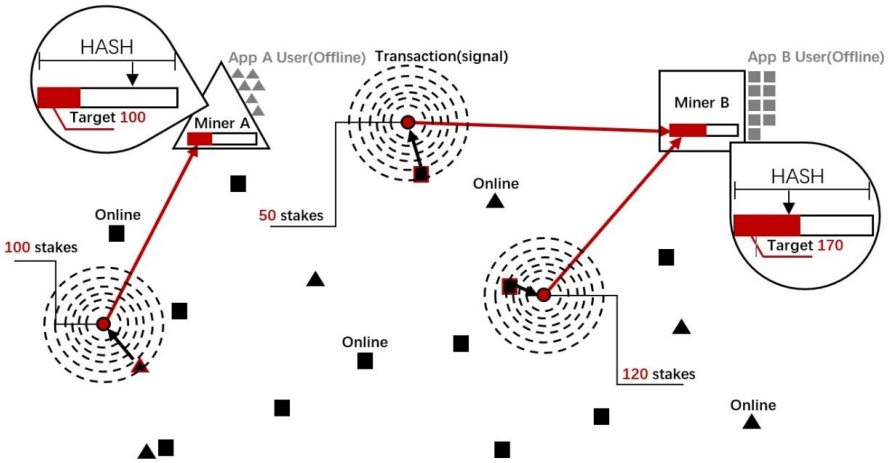


图 2. 争夺投票节点示意图

如上图所示，在线用户更多的 App，有更高的几率获得选票（权益）[8]，当前的选票会有助于在将来的竞争中获胜。

而投票的结果将会记录在已经产生的区块中，下面用两张图分别解释出块权投票（x 投票）和主链投票（y 投票）的工作原理：

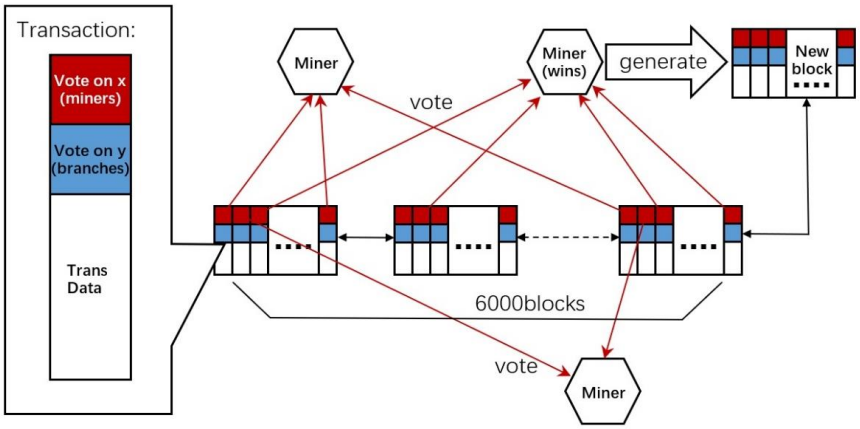


图 3. 出块权投票生效过程

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

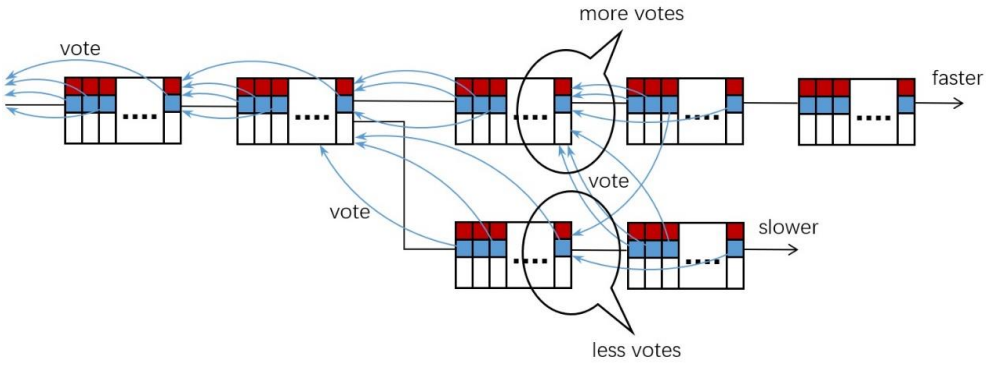


图 4. 主链投票生效过程

如上图所示，每当产生分支时，投票人会对分支进行投票，并记录在下一个区块中，这些记录决定了下次出块时各个分支所获得的选票（权益）多少，也就决定了出块难度参数 Y 的大小。因此获得越多选票的分支，产生下一个区块的速度也会更快，而更快的广播出块会在再下一次的投票中获得比上次更多的选票，因此上述过程将加速拉开两个分支的差距，迅速决定出主链。

把主链的投票与记账权投票分离的意义在于，将主链投票独立于挖矿收入之外，用户就能保持投票的客观性。而且因为主链只需要根据链上数据即可进行判断，能够被操控的空间很小。这样能保证即使在出现用户很集中的情况，也可以效的防止对分支的攻击。

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进行，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，我希望以此能够帮助一些开发者走出只有靠显示广告获得收入的困境，或者能够在使 App 或网页拥有更佳的用户体验上有所贡献。另外大量的网络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，虽然我设计的初衷是大家都用更多工蜂来获得选票，例如在 App 中嵌入工蜂程序，为了有更多的工蜂，就必须不断提高 App 的品质来赢得在线用户。但是不可避免的，会有一些用户相互达成协议，以团队形式合作挖矿（比方说钱包或者矿池）。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，平衡三种方式的矿力，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，把一部分收益分配给钱包应用的运营商，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。如果钱包供应商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物质上激励的基础。

### 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，而是在交易时销毁掉，当节点所投票的矿工成功出块后，再作为收入的一部分进行分配。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 5%，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 20%，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。

### 5.2 挖矿奖励

为了激励更多的高权益用户参与挖矿，系统会在手续费之外额外再增发一小部分（与手续费相当）的货币作为挖矿奖励。PoND 协议中，挖矿奖励由出块矿工和参与出块的投票节点按照分配规则分享。为了说明得更加清晰，我们设定以下变量名：

(1)V.投票节点收入，下标表示类别，三类投票节点收入分别为 $V_a$, $V_b$, $V_c$

(2)M.矿工收入，下标表示类别，只有 A, B 两类有此项，分别为 $M_a$, $M_b$

(3)W.钱包收入，只有 A 类有此项，即 $W_a$

(4) R.挖矿奖励总收入，下标表示类别，三类总收入分别为 $R_a$，$R_b$，$R_c$，满足 $R_a=V_a+M_a+W_a$，$R_b=V_b+M_b$，$R_c=V_c$

考虑以下的问题并根据解决方法推导出的公式为：

(1) 为了减少钱包应用参与挖矿，需要保证钱包的收益始终大于钱包自己挖矿的收益，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，收入通常都会大于 B 类挖矿收入，所以简化为 $R_c=R_a$，同时限制 $R_a>R_b$

在遵循以上公式的前提之下，收益的大小根据当前链上的数据进行动态调整，以平衡矿力。因为有 $R_c=R_a$，所以只需要在 A，B 两类挖矿收入中进行调整即可：

(1) 根据最近参与竞争的总权益数，可以判断当前用户在两类挖矿中的分布情况，如果参与用户过多，则降低此类投票节点收入 V，反之则增加；

(2) 根据最近每次参与出块的权益在参与竞争的总权益中的比例，可以判断当前的矿工的竞争程度，如果竞争程度过高，则降低矿工的收入 M，反之则增加。

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，完全由矿工进行分配，矿工可以让出自己收入 $M_b$ 的一部分以吸引更多用户合作。

## 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，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) 同样是过滤交易，但目的是影响分支的发展速度

因为每次出块包含的交易数量也决定了当前链下次出块速度（影响到 Y 参数），攻击者可以有意减少某条分支的交易数量，影响主链的认定。

解决方法：我们在共识过程第 4 步检查交易内的地址 b 时，除了标记指向当前链顶端的交易为 y 以外，还标记指向当前链第 2、3（数量可调整）块的交易为 y1，计算 Y 值时，y、y1 两种标记的交易同时统计。这样即使某个攻击者削弱了某个块的出块速度，只要他后面的矿工是诚实的，就会把攻击者有意漏掉的交易补上，这些交易会指向当前链第 2+个区块。如果这些交易可以有效的提供给难度参数 Y 所需要的权益值，那这条支链发展的速度也会由后面的矿工弥补上来。

(3) 模拟工蜂，即试图通过模拟创建大量的工蜂节点在 p2p 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数个节点建立连接即可。

(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，使安全性打了很大的折扣。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大大增加，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 的难度是严格根据在线权益的（参考注解 [3]），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) 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；
(5) 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] 认定主链遵循“最重”原则，参考 GHOST 协议；工蜂的请求中会有一个块地址作为参考，但采用与否完全在投票节点。

[2] x 标记的含义为记账权投票结果，而这里还需要检查 y 标记，因为这样做能保证选错分支的节点将不会获得任何收益，促使投票节点在选择分支时更加谨慎。

[3] 由于本协议对在线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，可以直接根据在线权益的大小对参数 d 进行设置，这样可以有效防止各种历史攻击，例如“Long Range Attack”或“Costless Simulation attack”。

[4] 此过程期间如果收到竞争分支的区块，可以并行处理。为了不削弱主链，只要主链重量不超过当前链+8，不要停止在当前分支的尝试，这样也符合自身利益（有可能变成主链）。

[5] 关于验证参数和收益分配数据所占用的额外空间，至多相当于给每条交易增加 4 字节的 id、4 字节的收益，完全在可接受范围之内；验证区块所要做的额外工作则至多是遍历之前 6000 个区块，只要算法得当，根据目前数据查找速度，也不会造成瓶颈。

[6] 为了控制投票频率，需要统计每个账户上一次交易到当前的区块间隔，以一定粒度，比如以 10 个区块为单位作为调节系数，从 0 开始，每过一个单位时间，系数增大一定比例，直到一个投票周期，这里为 6000 个区块，大约 100 个小时之后才能将系数恢复到最大值；同样每一个 UTXO 也增加调节系数，以转账频率调节权益大小，方法和参数都与前者相同。

[7] 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。

[8] 理论上获胜的几率和在线节点数成正比。

## 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/