一种利用周期投票的链式共识算法

摘要.此算法实现的是一种区块链共识协议，本文通过对投票、主链判定和终结等过程的描述介绍了这种链式权益证明机制的主要算法。

关键词

1.主链(Canonical Chain).
2. 终结(Finality).
3.反悔期(Hesitation period)

1. 引言

本文的初衷是希望在链式共识的基础上设计一种安全可用的共识协议，为公链系统提供更好的底层服务。因为链式共识相对于 BFT(拜占庭容错)体系的共识来说，它对验证节点的开放策略和利用竞争的增长逻辑更加符合公链——主要是加密货币系统——的核心定位，工作原理更加直观、简洁和高效，应用也更加成熟，但到目前为止的链式共识方案都还没有完全发挥出这些优势。同时，BFT 类的共识相对于传统的链式共识也有一些非常优秀的特性，在本设计中也尝试加以实现。

在加密货币系统中，我们最基本的需求只是能用的“账本”，至于所有节点是否随时保持同步并不是我们真正关心的问题，从这点上来说很多分布式系统的要求都过于严格，比如 PBFT(实用拜占庭容错)协议，它虽然实现了记账功能，但是其节点之间复杂的交互都过于繁琐，消息复杂度太高，不能支撑较多的验证节点，而且在公共网络当中遇到网络状况不好的情况时，反而会降低效率。但是此类系统也有很多优点，比如低能耗、高通量、明确的“终结”(finalization)点等，这些特性是最初的链式共识所不具备的，但是经过不断的改进，也已经有了对应的解决方案。

对于这些特性，能耗的问题最早就有了解决方案，最初的 PoS(权益证明)机制就是其中之一。而 BFT 共识下吞吐量高也只是因为其验证节点数量少而带来的错觉，同样节点数的情况下消息复杂度低的链式共识显然会更加高效，只要适当地控制验证节点的数量，链式共识可以适应不同流量要求的更多场景。剩下的“最终性”(finality)的问题是最难解决的，也是本文的关注重点。权益证明机制的出现，实际上已经为最终性的实现打开了大门，因为权益的总量是可以统计的，那么我们根据权益进行的集体决策就有机会取得一个占绝对优势的结果，理论上它就可以被“终结”。事实上也已经有了具备最终性的共识协议出现，但是这些终结方案都显得不太完美，比如 Casper 的方案虽然具备了最终性，但实现方式还是过于复杂了：“投票”这件事如此简单，没有必要把确认主链和终结过程分离，既增加了代码复杂度又降低了效率。而且 Casper 参照 PBFT 的原理使用了两段式终结，主动延长了终结时间，并没有很好地利用链式共识的优势。

本文介绍了一种新的设计，以下是基本算法的介绍。

2. 共识过程

共识过程依次分为以下几个步骤：

(1)生成选票，权益人依照所持权益多少按几率生成选票，获得投票和出块资格，成为矿工；
(2)竞争出块，矿工们依照拥有的选票数量按几率生成区块，获得奖励；
(3)确定主链，矿工周期性地对区块投票，然后将获得选票最多的分支认定为主链；
(4)区块终结，当计票结果满足一定条件，可以确定此结果在特定假设(2/3 节点诚实)前提下无法再改变，此时称为区块终结，已终结的分支为最终主链，已终结的区块数据就是最终数据。

2.1 生成选票

权益人根据当前常量(区块哈希值，时间戳等)，进行一个随机数的运算，再与账户掌握的权益数进行比较，达到要求者获得投票资格。此条件记为：*VRF_VoteProof<Account_stakes*[1]，其中 *Account_stakes* 表示账户权益，其越大，满足条件的可能性越高，由此达到根据权益生成选票的目的。取得投票资格的权益人即可生成一张选票，发布到网络上，登记成为矿工(或称投票者)，登记记录和交易记录一样需要被打包进区块。每张选票的生效期为当前(哈希值常量取值的)区块至其之后的若干个区块之间的时间段(通常以小时计，以不至于使用户操作得过于频繁，例如“600 个区块”或“2 小时”)。

投票资格的引入是为了能够降低并灵活控制验证节点的数量，以便不同的项目在安全性和响应速度之间做出平衡。通过控制获取选票的难度，我们可以把系统总选票数控制在需要的数目左右。根据概率，设定的总票数越多，实际产生的选票数量越接近于理想数量，我们根据总选票数来进行的各种投票结果就越可靠，但是选票越多，投票节点就越多，共识速度就越慢，这需要根据不同项目需求进行调整。

2.2 竞争出块

权益人成为矿工后开始参与区块链的创建和维护工作，为了获得奖励，矿工们需要互相竞争创建区块的权力：

(1) 矿工节点基于常量(时间戳、区块哈希值等)进行一个随机运算，期望结果达到某个目标，满足出块要求，记为：*Block_Proof<target*votes*efficiency*。其中 *target* 是难度目标，用以控制出块速度；*votes* 表示矿工当前拥有的选票数，可以看到选票数与出块几率成正比；*efficiency* 表示出块效率，受到多种因素的影响，将在后面详细介绍；

(2) 当达到出块要求后，矿工节点将他收到的交易和投票信息等打包生成区块并发布到网络。

2.3 确定主链

跟所有链式共识一样，决定哪个分支成为主链是共识过程中最重要的一步。从创世块开始，矿工节点周期性地(例如每“10 个区块”或“120 秒”)基于先前的主链选择一个分支，然后对自己选择的分支顶端区块地址签名并发布到网络，称为投票，投票记或录将和交易一样被打包进区块。确定当前主链的过程就是根据这些投票记录来比较分支优先级，最后选择优先级最高的一个分支(也是链)的过程。

2.3.1 分支的优先级

在我们的协议中，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”。“重量”在这里也并不等于区块个数，而是等于分支所获得的选票数。定义：一个分支的重量等于它的根节点和其所有子孙节点所获得的选票总数。在此计算方式下，两条链在比较优先级时应该从它们共同的祖先出发，分别计算它们各自所属的两个分支的重量，较重者优先。为了找到主链，我们需要逐个比较当前存在的竞争分支，找出优先级最高的链。(如图 1)

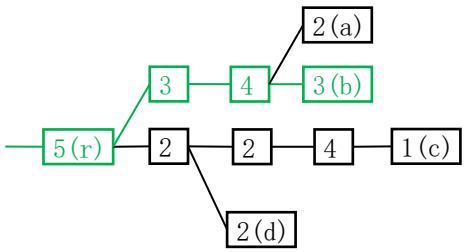

图 1. 分支重量计算示意图
如图中所示的分支结构中，优先级最高的链为 r→b

2.3.2 激励措施

投票过程中还有一些问题需要解决，首先是要保证矿工有足够的积极性进行投票，这一点可以利用使投票“正确”性与出块效率挂钩的方法来激励矿工：根据“上周期投票”(大于等于一个周期，小于两个周期以前的投票)到当前链的距离远近(后面会解释)来确定当前出块效率。这又带来下面两个问题：在产生分叉时，如果矿工都为了追求投票的正确性而选择投向分叉之前较稳定的区块的话，就会影响系统稳定的速度；或者矿工选择推迟投票，等分叉已有明显胜负之后再投票的策略，同样也会影响系统稳定的速度。对这两个问题，我们首先在原有周期投票的规则上稍加修改，之前的规则是“投票间隔不能小于一个周期”，现在增加“记录投票的区块与下一个投票之间的间隔不能小于一个周期-1 的长度”。按照新的规则，每当有矿工投票给较早的区块，或者推迟投票，此次投票与上次投票之间的间隔就会大于一个周期，而超过一个周期的地方出现一个空窗，每当区块的“前 2 周期-1”处位于这个空窗内，这个矿工在此区块位置的“上周期投票”都不存在，因此其正确性为零，无法出块(如图 2)。

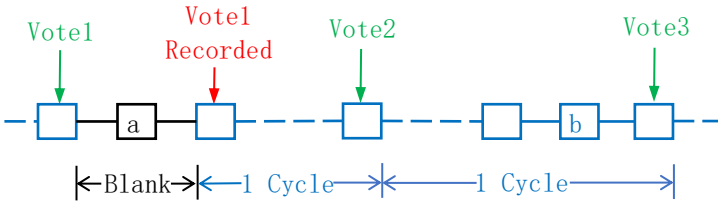

图 2. 出块空窗示意图
如上图所示，当前矿工在 a 区块位置有一个空窗，导致 a 之后 2 周期-1 处的 b 区块位置无法出块

所以为了尽量少出现这种空窗，矿工应该选择尽早投票，且投给最顶端的区块。其次，我们利用使投票的“重要性”也和出块效率挂钩的方法来解决，用“正确投票总数越少，投票的重要性越高”的判定，来使矿工在分叉之后投票分散的时候的投票具有更高的价值。

2.3.3 投票规则

综上各个因素，有如下投票规则：

(1) 矿工每两次投票的目标区块之间的间隔必须大于等于一个投票周期，否则会被惩罚；

(2) 矿工在当前链上的投票(目标区块在当前链上)被记录的区块与下次投票的目标区块之间的间隔必须大于等于一个投票周期-1 的长度，否则会被惩罚；

(3) 矿工在当前链的出块效率参数(*efficiency*)受到他的上周期(1 周期≤距离当前位置<2 周期)投票的“正确性”影响，具体为：该投票的目标区块距离当前链的距离(分叉之后的子孙代数)越小，正确性越高，出块效率越高，反之越低，目标区块在当前链上的投票称为“正确”投票；

(4) 矿工在当前链的出块效率参数(*efficiency*)受到他的上周期(同上)投票的“重要性”影响，具体为：投票目标区块位于该高度的正确投票总数越少，单个的投票重要性越高，出块效率越高，反之越低；

2.3.4 信息完整性

这里还存在一个信息的完整性问题，即在一条链中存储的数据能不能包含完整的投票和分支信息。如果可以的话不仅能简化比较优先级的算法，还能为之后的“最终化”提供支撑，更重要的是让共识过程有完整的记录，这保证了系统的客观性和可验证性。要讨论这一点，需要看矿工们有没有足够的理由去打包所有的投票信息，毕竟投票是不提供手续费的。

我们分两种情况来看：

对于指向本链的投票，矿工自然会尽量地多保存到块，以此来保证自身链的竞争力；

对于指向其它链的投票，首先，记录它们并不会增加对方链条的竞争力，因为其指向的链本身也会记录，在比较两链优先级时取的是各自链中的数据。相反这些投票很可能在更大的分支之间做比较时转而帮助己方获胜。其次对出块矿工而言，如果记录了某个投票者的外链投票，就降低了这个投票者在本链出块的效率，从而增加了一些矿工自己出块成功的几率，虽然只是小小的激励，但也有助于促使矿工去记录这些本就不复杂的投票信息了。与此同时，每个区块也会尽量引用这些其它分支的区块(块头)，用来支撑记录的投票信息，从而完善了区块树。

以上论证了我们能够保存完整的投票和分支信息的合理性，实现中还需要设计相应的数据结构，本文不再赘述。

2.4 区块终结

当达成的共识形成绝对优势，主链再也不可能被替代后，区块就可以被固定下来，区块中的数据成为最终数据，这个过程就是“终结”。链式共识的终结过程不像 BFT 类型的协议那样，需要用多轮投票来保持节点之间严格的同步状态。当第一轮投票达成共识后，我们只需要知道这个共识是绝大多数人的决定，而并不需要关心是否所有节点都接收了投票结果，因此在理想情况下只需要一轮投票即可完成终结。但为了保证被终结的链不产生冲突(安全性)，同时还要让投票者有机会修正自己投票，以防止陷入永远无法终结的境地(活性)，情况就变得更加复杂了，需要增加新的投票规则并且推迟终结的时间。

2.4.1 新的投票规则

根据周期投票的机制和非诚实用户不超过 1/3 的前提，我们可以确定：只要一个分支在一个周期之内获得了 2/3 以上的选票，那么这个分支就不应该有竞争者，可以被终结。但是要保证所有被终结的分支都不相互冲突，我们需要有一条从创世区块开始的连续的终结链。这需要投票者的投票也能够前后连贯，为此我们修改了刚才的投票规则，在原有的规则上增加：

(5) 在每次投票时需要向前连接他上一次的投票，称为“来源”。不连接来源的投票称为“无源投票”，是作为对错误投票的修正，没有权重，但是可以被后面的投票用作来源。无源投票可以与有源投票在同一个周期内同时出现，但自身也需遵守之前的投票规则(1)(2)，否则也会被惩罚；

(6) 创世块是第一个终结(finalized)块，在不违反其它规则的情况下创世块可被后面的投票用作来源；

(7) 来源位于终结块的投票称为“有根投票”，只有“有根投票”才具有权重和正确性，从而递进地产生后续的终结块，终结块之前的所有祖先块同时也视为已终结。“有根投票”可以向前传递；(如图 3 和图 4)

(传递是指假如 c 是有根投票，那么连接 c 作为来源的投票同样也是有根投票)

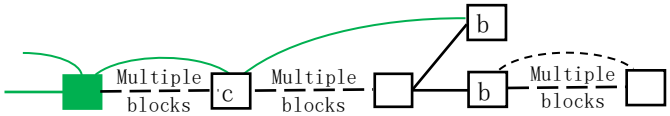

图 3. 矿工投票链示意图

如上图所示，绿色实心区块表示已终结块，绿色实线表示有根投票，黑色虚线表示无根投票。图中投票者因为前后冲突，他的投票链就断了

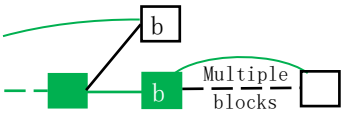

图 4. 有根投票示意图

如上图所示，继续图 3 的情况，当节点 b 成为终结区块之后此投票者后面的投票又变成了有根投票，投票链重新连接了起来

2.4.2 安全性和活性

保证安全性的原理很简单：因为投票者每次投票都只能与他上一次投票相连，如果存在冲突的投票，就不能保持为一条首尾相连的链条(如图 3)，所以他的投票链就不会出现冲突。既然单个投票者的投票链不会冲突，那么基于连续投票链来生成的终结点也不会存在冲突，由此保证了安全性。(如图 5)

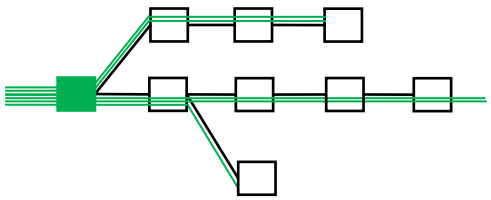

图 5. 安全性示意图

如上图所示，只有连续的投票链才能终结区块

但这显然还不满足活性的要求，因为出现分叉后，很可能有超过 1/3 的选票被投向错误的分支，一旦出现这种情况，终结链就断了而且无法恢复。所以要保证活性就比较复杂：首先我们要对这种情况——我们称为“关键分叉”——能够察觉；然后允许下次投票的来源从本次投票位置倒退回分叉的地方，给投票者们一个反悔的空间，使他们能够及时改正之前的决定，回到主链上来(如图 6)；最后把终结位置也推迟到分叉的地方。

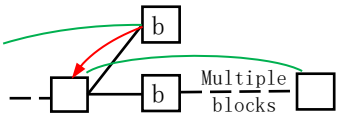

图 6. 投票源移动示意图
如上图所示，继续图 3 的情况，当此投票者的下一个投票源倒退回分叉位置后，它就成了有根投票，投票链又回到了当前分支上。

2.4.3 关键分叉的察觉和定位

为了能够察觉关键分叉，我们要对终结条件做一些调整，之前的条件是一个分支“在一个投票周期之内获得超过总数 2/3 的选票”，改为“在一个投票周期之内获得超过总数 2/3 的选票，且这些选票的投票目标距离来源都小于两个周期”，换句话说，必须是由连续投票(间隔不到两周期)选出来的分支才能被终结。此终结条件把能产生终结的所有投票的来源都集中在终结点之前的两个周期范围之内(如图 7)，这样我们只需要在这个范围进行检查，看是否有足够的投票就可以了。

图 7. 终结条件示意图
如图所示，满足区块 b 的终结条件的投票目标集中在 b 为根节点的 1 周期长的分支上，而来源集中在 a 之前 2 周期长的链上

察觉到关键分叉后，我们使用一种回溯算法来找到分叉点和它对应的投票反悔期长度：

当生成一个高度为 *h* 的块 *a* 时，我们使用一个自然数 *n* 来表示 *a* 位置的临时“反悔期”，高度 *h-n* 的区块称为临时“回撤点”，回撤点通常也等同于分叉点。*n* 的取值基于于以下计算结果：

(i) *a* 之前(包括 *a*)两个投票周期内的链上获得的选票数超过总选票的 *2/3* 时，*n=0*.
**(同一投票人只计最后一次投票)*

(ii) 上述票数小于等于总选票的 *2/3* 时，将目标是 *h* 高度的所有投票回溯到高度 *h-1*，再和目标高度 *h-1* 的所有投票一起回溯到 *h-2*，以此类推，当回溯到高度 *h-i* 时，满足(i)的条件，则 *n=i*.
**(由于对高度为 h 的区块的绝大部分投票都保存在高度 h+1 的区块中，所以上面这个工作实际上是在生成 a 的下一个区块 a’ 的时候来做的，对应的 n 值也保存在 a’ 中，但这并不影响算法实现，因为 n 在被使用的时候 a’必定早就生成了。更进一步地，可以把“目标是 h 高度的所有投票”改进为“保存在 h+1 高度的所有投票”并以此类推，这样能够把那些没有在第一时间被记录或者推迟发布的投票都计算进来。)*

(iii)*a* 位置的临时回撤点高度不能低于 *a* 前 *2* 周期处的实际回撤点高度；

*(换个角度看：回撤点可以被后出现的但高度更低的回撤点取代，但不能超过 2 个周期，也即第(iv)条规则。2 周期是在最坏情况下察觉到关键分叉的距离，如图 8)

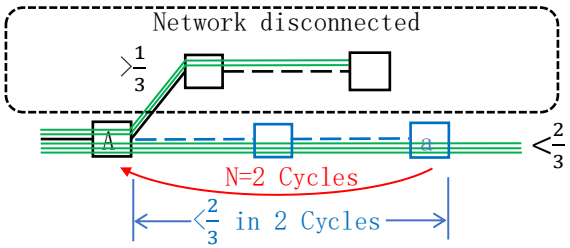

图 8. 最坏情况下察觉到分叉示意图
如图所示，当大于 1/3 的选票在 A 处分叉，而且网络刚好在此时出现故障，导致这些投票信息全都丢失时为最坏情况，此时需要一个检测长度的时间才能察觉到分叉

(iv)*a* 位置的实际回撤点取自 *a* 位置之后(包括 *a*)*2* 周期内最早的那个临时回撤点，实际反悔期 *N* 也由此而来；
*(因此，在最坏的情况下一个区块在其生成后需要等 2 个周期才能被终结。但通常情况下，只要区块之后的一条链在这 2 个周期之内获得了 2/3 的选票，就不会再有回撤点早于它了，此时就能够得到实际回撤点，可以立即被终结。)

2.4.4 投票回撤和延迟终结

取得反悔期之后，投票回撤和区块终结过程如下：
(1) 矿工的每次投票只能使用他上次投票目标或其回溯 *N* 步(包括 *N*)以内的区块作为来源， *N* 值取自于此矿工上次投票目标的高度上，相关链的实际反悔期 *N*，违反此规则的矿工将被惩罚；(如图 8)
*(“相关链”是指上次投票的目标所在的链和当前链，如果上次投票目标不在当前链上，则需要分别取得两条链的实际反悔期，然后取其中较短的一个。[2,3])
*(“上次投票目标的高度上”可改进为“上次投票在本链保存的高度-1 的位置上”，与回溯算法第(ii)条后面的改进相对应。)

(2) 从根部算起，当一个分支在一个投票周期之内获得的选票超过系统总选票的 *2/3*，而且这些选票的投票记录中来源和目标之间的距离都小于 *2* 个投票周期时，满足终结条件，但终结区块不是分支的根节点 *b*，而是 *b* 处的实际回撤点 *B*。(如图 9)

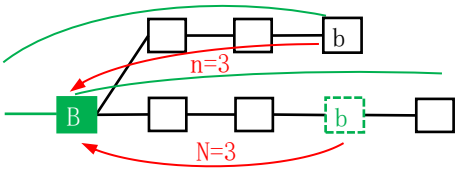

图 9. 反悔期、投票回撤和终结点后移示意图

依照这种方法，在通常(较理想)情况下，N 值都为 0，我们可以在生成区块后大约 2/3 个投票周期完成终结，偶尔会有一些分歧比较严重的分叉可能会延续几个周期的长

度，但总体终结时间一定远小于两个周期。在一些网络状况很差或遭受攻击导致投票特别分散的情况下，可以预先察觉到分叉并且延迟终结，放宽投票要求，提高系统容错率，可以防止被比如“左右摇摆以延长分叉长度”式地恶意攻击而导致失去活性。顺便提到我们的分支重量计算方式——不是像 LMD(Latest Message Driven)方案那样只统计最后一次投票，而是相对麻烦地把每次投票都统计在内，并且按周期投票，也是因为这样做对此类攻击具有较强的抵抗力[4]。

2.5 惩罚、奖励和押金

对于用户的违规行为，比如前面提到的违反投票规则和下一节要介绍的违反弃权规则的用户，我们会对其进行惩罚，也会对提交违规证据的用户进行奖励。这需要用户在做相应操作之前存入(或称锁定)一部分“押金”，作为惩罚和奖励的资金，这部分资金会在锁定一段时间之后解押。用户所交押金的金额与他拥有的选票数或权益量成正比。

2.6 优化

(1) 计算投票资格时用到的“区块哈希值”可以做一个优化：因为权益小的用户有可能很长时间都不能获得投票资格，如果还要让他们每时每刻根据当前区块的信息作为常量来尝试生成选票的话会增加没必要的开销。所以我们将用当前区块之前的第 m 个区块信息来代替(若干个小时以前)，这样用户就能提前若干个小时知晓自己取得了投票资格，以后只需要每隔若干个小时集中计算一次就可以了。但是为了防止通过转移资金来获得竞争优势，我们必须规定：经过转账的货币必须在 m 个区块之后才能作为权益来生成选票。

(2) 对于权益小而且不希望花太多的代价参加挖矿的用户来说，他们也可以发布“弃权”登记，宣布在一段时间内不生成选票、不参与出块和投票。这样做虽然不能直接参与系统维护，但是可以为我们提供更准确的活跃权益数量，进而得出更准确的总选票数，加快最终化进程，为系统的稳定做出一定的贡献。对于这样的弃权登记，我们可以给予一小部分奖励(与权益数量成正比)，但是这些弃权的权益人如果被证明参与到了投票或出块活动中的话，则会受到惩罚。

(3) 由于矿工登记记录没有交易费奖励给区块生成者，为了激励矿工将它们打包进块中，我们要根据生成块中登记记录的数量来调整出块效率参数 $efficiency$，从而影响该矿工下一次出块的速度，使得打包的登记记录越多越好。该数量的标准值每隔一段时间应该做一次调整。

(4) 如果出块速率设置得较快，导致短分叉频繁出现，不妨把默认的反悔期直接设为 1 或 2，默认终结点也相应延迟 1-2 个区块，可以减少长分叉的出现次数，使系统运行得更加顺畅。

3. 结论

以上章节介绍了本协议的基本算法，由于它是基于链式共识和权益证明所设计，所以自然延续了这两者的部分优点，比如权益证明的低能耗、链式共识对网络环境的高容错、验证节点更多、加入要求更宽松等。除此以外，算法在保证了安全性和活性的同时，还能提供延迟在一个投票周期以内的终结能力。同当前已有的一些共识方案相比，此算法具有较为独特的优势，能更好地满足部分场景的要求，将来可以考虑加入可选方案之列作为补充。

利益冲突

本设计已申请专利，专利号：202110495622.6

注解

[1] VRF,“Verifiable Random Function”，即“可验证随机函数”，一种使用私钥加密公钥验证的随机函数，使用 VRF 可以避免生成的选票被攻击者提前预测。↩

[2] 为了实现此方法，需要在生成区块时把临时反悔期 n 存入区块头，以便在其它链的视角也能够获取。↩

[3] 如果上次投票位置的实际反悔期还没有产生，就使用当前能获得的最长的反悔期来替代。这样做有可能使当前反悔期偏短，但因为回撤点一定在实际回撤点之后，所以不会影响安全性；也不会影响活性，因为实际回撤点会回到关键分叉处，投票者可以在后面的投票中再回到主链。↩

[4] 当只计最后一次投票且不受投票周期限制时，攻击者对一个分叉的投票不仅会增加当前叉的权重，而且会在原先分叉扣掉先前的投票权重，此消彼长的过程导致这种攻击的效果是他本身选票权重的两倍。↩