Wavelet： 去中心化异步通用 PoS 账本

Kenta Iwasaki
kenta@perlin.net

Heyang Zhou
heyang.zhou@perlin.net

2019 年 8 月 16 日

摘要

所有的分布式系统都受到一个著名问题困扰：伸缩性困境 (译者:the scalability dilemma). 这个困境逼迫系统设计者不得不在性能，一致性和可用性三者之间做取舍；当你想尝试创作一个实用的，自愈的，容错的应用时是最重要的方面。至少可以说，任何一个实用的分布式系统，区块链，都受到上述困境影响。

下面，我们将对经典计算机科学文献的结果进行简单的分析，以描述和分析导致许多现代区块链结构不实用的大量缺陷。

1 简介

回想一下多读者，多写者问题 [4]. 假定，存在 n 个并行处理器都试图读写同一个共享资源 R。系统每增加一个处理器，协同 n 个处理器对资源 R 的访问工作量会成倍（译者：exponential) 增加。

现在，我们把这个问题映射到一个分布式情景中，在每个处理器之间存在通信延迟，那么，对于这个问题可以建立一个等价关系：在不引入指数级通信成本的情况下，有效地协调节点网络来维护分布式账本从根本上来说是困难的。

更窘迫的是，在设计一个协议来协调多个读者和多个写者与资源 R 交互时，如果考虑到某些处理器有问题，又会带来大量额外的处理开销。在文本中，我们把这个问题统一称为可伸缩性困境（*scalability dilemma)*。

1.1 最长链法则不具备伸缩性

考虑一个由 n 个节点组成的网络，一个难度系统，机制上类似于去中心化的随机抽奖。在一个较短的间隔时间 Δ 内， n 个节点中的一个子集 F 提交了一个交易区块，试图加入到这个网络中共享的 DAG 图 G 的最长加权路径上。

我们把这种分布式系统的行为，称之为最长链法则 [10]。比特币、以太坊和其他一些分布式账本都是典型的遵循这个法则的分布式系统。

在这个时间间隔 Δ 内，最多存在 $|F| - 1$ 个冲突的区块，这些区块同时试图追加进 G 的最长加权

路径上，也称为最长链 。要解决掉 $|F|-1$ 个区块之间的冲突，又需要不断有新的区块提交到他们自己已提议的区块中。但难度系统使得 $\in F$ 中的节点，要通过指数级别时间复杂度才能解决冲突，这是一项艰巨的任务。

这种情况放宽来说即：最长链法则使得 n 个节点竞争对某个资源 R 的读写机会。作为最长链法则，对于取值大的 n 显然是不具有伸缩性的。因此，任何一个遵循最长链法则的分布式系统都面临可伸缩性困境。

1.2 委员会模式提高了性能但降低了安全性

考虑一个由 n 个节点组成的网络，其中 C 是 n 个节点的子集，表示一个委员会。委员会有责任验证和处理这个网络中的所有交易。

只要在给定时间能选出委员会，怎么选的不重要，无论是通过权益委托 [9]，可验证随机函数 [6]，还是通过去中心化投票方案 [2]。典型的使用委员会模式的分布式账本有 Algorand, Tendermint, Ouroboros Praos, Dfinity, 和 EOS.

只要这个分布式账本在 $\in C$ 的节点间维护并保持通信，这种形式是具有伸缩性的。但是，当 $|C|$ 很小时，那么任何形式的 DoS 攻击或委员会内部勾结，都会危害全网。

为了把此类攻击风险降到最低，就需要扩大 C 。然而，扩大 C 会成倍地增加通信复杂度，与维持一个小的委员会的好处是矛盾的。因此，任何依赖于节点子集来验证和处理整个网络中所有交易的分布式系统都面临可伸缩性难题。

1.3 只保证交易部分有序的账本并无实用价值

值得注意的是，一些基于 DAG 在节点间复制交易的共识协议，例如 Avalanche[12]，存在以下问题：

1. 只能保证交易的部分有序
2. 无法保证交易的不可逆性，且
3. 当试图使新节点引导并加入网络时，容易受到许多挑战。

无法实现这些保证，就决定了账本无法提供大量的功能。例如，如果缺乏全局交易顺序保证，这个账本除了做一个支付平台之外什么也干不了。

一个图灵完备的智能合约平台也不可能构建在只保证部分交易有序的账本之上。这是因为，智能合约在网络中不同节点的执行顺序是完全不同的。

1.4 网络假设和协议保证

定义 1.1. 敌对模型 (Adversarial model, , 译者：关键术语提供英语原文) 我们假设存在一个强大的，灵活的对手，具有观察网络所有节点内部状态的能力。只要每个节点成功地将 $1/k$ 的消息传递到

目标接收方，并且 k 是一个较小的固定的安全参数，那么这个对手可以均匀的丢弃，重新排序或重定向来自每个节点任意数量的消息。

假设网络中所有的通信都是部分同步的 [5]，在给定的对抗性模型下，对 k 的概率边界是温和的，那么，Wavelet 能同时保证了强安全性和网络活性。在这样的条件下，Wavelet 能够实现不可逆性、全局交易一致性和良性交易的正常采纳。

2 Wavelet 协议

该协议由并行运行的两个进程组成：*gossiping* 和 *querying*

定义 2.1. 身份 (Identity)。网络中的每个节点都假定有一个公钥基础设施 (PKI)，其中节点由它们的关联公钥标识。

定义 2.2. 哈希 (Hash)。定义密码学哈希函数 H, 函数是不可微 (in-differentiable) 随机预言机，我们将哈希函数 H 的所有输出的位长 (bit-length) 表示为 $|H|$。

定义 2.3. 账户 (Account)。帐户由关联的公钥引用，并由虚拟货币的账户余额 **balance**，和一个 **nonce** 两者组成. **nonce** 是一个计数器，它按顺序计数对指定帐户所做的更改，用于防止重放攻击。

定义 2.4. 状态 (State)。网络中的每个节点维护账本状态 [公式] ，代表一个不可变更，默克尔化的键值存储，并且会产生一个默克尔树根 δ。

定义 2.5. 图 (Graph)。网络中的每个节点维护一个有向无环图 G (译者: directed-acyclic-graph G)，其顶点由已接受的或待处理的交易构成，顶点与顶点之间的连接（边）代表了一种因果关系的拓扑顺序。

定义 2.6. 交易 (Transaction)。交易 tx 表示一个原子变动单元，从交易创建者来看，节点会将交易应用于其状态 S . 每一个交易具有唯一 ID，唯一 ID 是所有内容和元数据的校验和。交易是 G 中的顶点，它们通过自己的 ID 随意地引用一组父顶点 tx.parents。 交易将它们相对于图 G 的深度引用为 tx.depth。交易通过有效负载 tx.payload 来传达对状态 S 的变动，有效负载用于调用在 tx.tag 下指定的一些惟一操作。简而言之, 一个交易： $\text{tx} = (\text{id}, \text{depth}, \text{parents}, \text{tag}, \text{payload})$.

定义 2.7. 交易作者 (Transaction authorship)。一个交易 tx 被应用到某个状态 S 的时候，是与交易的创建者有关的。交易的创建者由其公钥作为 tx.creator 引用，对于交易的创建者来说，他可以（可选）把交易提交和中继责任委托给网络中的其他节点，通过 tx.sender 指定代理方公钥。交易必须始终指定其创建者和发送者，即便交易的发送者是创建者本身。交易的创建者会对交易进行签名，并把签名后的 $\textbf{nonce} \| \text{tx.tag} \| \text{tx.payload}$ 这段数据附加到交易，而交易的发送方会对整个交易内容进行签名，并附加到交易 tx，不包括其唯一 ID.

2.1 创建一笔交易

假设 Alice 和 Bob 两方，Alice 持有大量虚拟货币，她希望创建一笔交易从她账户中扣除并添加到 Bob 的账户中。Alice 审视图 G ，并从当前算起节点深度为 DEPTH_DIFF 的所有节点中，选取最多 MAX_NUM_PARENTS 个父交易进行构造，当然，这些选取的节点是没有被任何其他子节点所引用的交易（例如图 G 中的叶子节点)。

定义 2.8. 交易深度 (译者：Transaction depth，避免和交易所的深度概念混淆)。选择了交易 tx 的父节点后，此交易的深度 tx.depth，通过以下公式计算并附加到交易中：

$$\text{tx.depth} = \max\{\text{parent.depth} \mid \text{parent} \in \text{tx.parents}\} + 1 \tag{1}$$

定义 2.9. 交易种子 (Transaction Seed)。计算好交易深度后，我们把经过如下计算的校验和称为交易种子 tx.seed，计算方法为：

$$\text{tx.seed} = H(\text{tx.sender} \parallel \{\text{parent.}\mathit{seed} \mid \text{parent} \in \text{tx.parents}\} \parallel \text{tx.id}[0]) \tag{2}$$

需要注意的是，交易 tx 的 ID 和 seed 都简单的存储在内存中，并且可以容易的被网络中的其他节点计算。在签署交易 tx 时，Alice 在生成并将创建者和发送者的签名附加到 tx 之前，会先排除交易的 ID 和 Seed。

2.2 Gossiping

(译者：以太坊中翻译 Gossip 为“八卦”协议，这里保留原文不译) Alice 接下来执行签名，并准备广播这笔交易给网络，这会通过任意一种具备最终一致性的异步 gossip 协议 $\mathcal{P}_{\text{gossip}}$ 来完成。在经过最终一致性周期时间 Δ 之后，如果没有新交易的发生，所有节点会最终使得自身的图 G 内容保持一致。

所有新创建的交易，新中继的交易，或其他任何被当前 gossiping 进程接收到的数据，都有待被网络接受。gossiping 进程的重要性在于确保节点最终能构建一个等价和一致的图 G 视图。

我们之前已经做了一个假设，承载网 $\mathcal{P}_{\text{overlay}}$ 能够在网络中存在对手的情况下完成通信，这个对手有能力丢弃，重新排序或者无限推迟单个节点发送的大部分消息。

定义 2.10. 深度断言 (Depth assertion)。收到来自其他节点的交易 tx 后，Alice 执行一个断言，之后再将 tx 添加到她的图 G 中，即：交易的深度 tx.depth 与其引用到的父交易深度之差不超过 DEPTH_DIFF。DEPTH_DIFF 是一个固定的系统参数，用于限制无法从图 G 中的任意深度选取父交易 (译者：parent transactions, 类似于 Bitcoin 中的 TxIn)。

定义 2.11. 顺序断言 (Order assertion)。在收到来自其他节点的交易后，Alice 执行一个断言，之后再将 tx 添加到她的图 G 中，即：父交易 tx.parents，是按字典升序顺序排好的 (译者：ascending order)。

定义 2.12. 父交易选择限定 (Parent selection limit)。在收到来自其他节点的交易后，Alice 执行一个断言，之后再将 tx 添加到她的图 G 中，即：最多 MAX_NUM_PARENTS 个父交易被引用。

2.3 Querying

（译者：Querying 进程不局限于字面” 查询” 本意) 除 gossping 进程之外，另一个并行运行的进程用于排序，完成交易确认 (译者：finalizing，根据上下文语义翻译为最终，或交易确认两个词)，并应用交易到节点本地状态 S。

Querying 进程渐进式的在每个节点上构建图 G ，完成从图 G 的根节点 G 开始成批的交易确认，直至图 G 前沿 (frontier)。更具体的说，querying 进程的目标是把每个节点上的图 G 一致的划分为共识轮 (译者：consensus rounds)。

定义 2.13. 共识轮 (Consensus round)。共识轮 R_r 是一个非重叠深度区间，记为：$R_r = (r, \mathrm{tx}_{\mathrm{start}}.\mathrm{depth}, \mathrm{tx}_{\mathrm{end}}.\mathrm{dep}$ ， $r \in [0, \infty)$ ，其中 $\mathrm{tx}_{\mathrm{start}}$ 和 $\mathrm{tx}_{\mathrm{end}}$ 是共识轮所组成的深度区间的起点和终点，一轮共识的起点和终点可以只是被标记为关键 (译者：critical) 的交易。

定义 2.14. 关键交易 (Critical transaction)。 一个交易 tx 只有在其种子 tx.seed 的前面的 0 位的个数 $\geq D_r$ 时才被标记为关键交易，其中 D_r 是一个动态系统参数，也称为难度参数，相对于某个给定的共识轮 r 是唯一的。

假设 Alice 正好处在她的初始共识轮 R_0 ，随着 gossiping 进程的进展，图 G 的构建，在发现一个关键交易之后，Alice 会提交一个属于她当前轮 R_0 的候选终点 (译者：candidate ending point)，这将是下一轮 R_1 的起点。这个提案记作：$R_1 = (1, \mathrm{root}.\mathrm{depth}, \mathrm{tx}_c.\mathrm{depth})$ ，其中 root 是图 G 中的根交易， tx_c 是 Alice 观察到的第一个关键交易。 另外，Alice 除了提交 R_0 轮的终点之外，她也可以从他的对等节点 (译者：peers) 上学习到其他属于 R_0 轮的可选终点。

接下来 Alice 会执行任意二元拜占庭容错协议 (译者:Byzantine fault-tolerant binary consensus protocol) $\mathcal{P}_{\mathrm{query}}$ ，以从网络获得 (译者：garner) 对共识轮 R_1 最终内容的一个无偏 (译者：unbiased) 的意见。

在 R_1 轮的内容完成确认之后，会广度优先遍历一次图 G ，查找图中起点为 root ，终点为 $\mathrm{tx}_{c'}$ 的所有路径中的所有交易，其中 $\mathrm{tx}_{c'}$ 是 R_1 轮确认后的起点，同时也是 R_0 轮完成确认后的终点。

执行广度优先遍历的顺序是所有待应用交易的最终顺序，这些交易在本地应用于 R_0 轮的每个节点状态 S 。所有在第 R_0 轮被遍历到的交易构成的集合，被标记为" 已接受"， R_0 深度区间内的所有其他交易被标记为" 已失败"。

定义 2.15. 状态断言 (State assertion). 用 r 表示所有共识轮 R 的时域序号，附加默克尔数根 δ，树根值是所有已接受交易在某一轮 $R_i \in R$ 对状态 S 的变更计算得出。在每一共识轮 $R_i \in R$ 完成确认之后，节点可以把 R_i 中已接受的交易应用到本地状态 S ，求得新的默克尔树根 δ'。节点此时可以断言 $\delta' = \delta$ 来判断 R_i 相关的新状态 S' 的完整性。

然后，将已接受交易全部应用到每个节点状态 S 上，每一次应用，都会增加交易创建者的 **nonce**。接下来，节点会重复整个 querying 进程过程，再次寻找 R_2 共识轮的下一个终点关键交易，并把 R_1 共识轮的根设置为 $\mathrm{tx}_{c'}$. 这一进程将归纳地推进到下一轮共识，并继续进行下去。

2.4 部分同步

本协议的一个关键点是为系统参数 D_r 计算一个理想的值，使其适用于共识轮 r。描绘一个画面，gossiping 是一个会在全网产生最终一致的图 [G 的进程，因此，对于较大的 n ，其中 n 为网络中节点的个数，n 越大，则 gossiping 协议收敛到使得所有节点具有一致图 G 的目标时间上限 Δ 越大。

大的 n 值也暗示了每时每刻有更多的交易，因为存在更多的源可以在整个网络中生成和分发交易。用 N_i 来表示网络中某个节点，这个节点的交易产生和广播频率定义为 [公式] ，那么任意时刻存在的最大交易量为 nf ，也是网络最大可能负载。

把用于确认单个共识轮所需的时间记作 t' ，把在非重叠深度区间，给定难度系数为 D_r 时的单个共识轮所包含的交易数记作 f'。

猜想 2.1. 部分同步界限 (Partial synchrony bounds)。 描述图 G 中待处理交易数为 $|G_{\text{pending}}|$，描述图 G 在某一时刻 t 已确认交易总数为 $|G_{\text{finalized}}| = |G| - |G_{\text{pending}}|$。在当前节点中，如果位于图 G 中的一个交易，它的深度小于或等于节点所知道的最近一次确认后的共识轮的终点交易的深度，则认为交易已确认。如果 $nf > \frac{f'}{t'}$，则 $\lim_{t\to\infty} \frac{d}{dt}(|G_{\text{pending}}| - |G_{\text{finalized}}|) > 0$。反之，如果 $nf \leq \frac{f'}{t'}$，则 $\lim_{t\to\infty} \frac{d}{dt}(|G_{\text{pending}}| - |G_{\text{finalized}}|) = 0$。

猜想 2.1的结果是，如果随着时间的推移，共识轮无法在图 G 的深度区间足够大的情况下完成确认，那么 $|G_{\text{pending}}|$ 会相对于 $|G_{\text{finalized}}|$ 在时间上严格递增。这是由于协议由于某个节点错误配置难度参数 D_r 而产生了活性故障（liveness fault），在这种情况下，共识轮做不到足够快的去确认交易，以至于过多的待处理交易被堆积在图 G 上。

为了避免这个活性故障，我们必须根据任意给定时刻在网络中的交易创建频率来调整难度参数 D_r 。为了做到这一点，需要一个可验证的，确定性的机制，该机制允许节点独立计算网络总负载的近似值，以调整每一轮共识的区间跨度。

我们提出了一种新的方法来估算网络负载，然后可以对数组合产生一个充分的 D_r ，动态调谐网络，以防止活性故障。

2.5 网络负载近似

回想一下 2.1 节中提到的父节点选择算法，来表示两个诚实节点 N_1 和 N_2 ，这两个节点独立的构建和广播交易，最多每秒产生 [公式]f 个交易，其中全网节点数目为 $|N|$ 。

假设 N_1 和 N_2 在任意时间 t 具有 gossiping 构建的类似的图 G ，那么在任意时刻上的 N_1 和 N_2 必定包含相似的选定父交易集 (译者：selected sets of parent transactions)。

用 Δ 来表示仅有 N_1 和 N_2 严格的以 $2f$ 每秒的速度创建和广播交易的时间间隔。那么节点 N_1 和 N_2 在整个时间间隔 Δ 内创建的交易可能在新创建的图深度内共存的最大交易数为 2。

推导：

猜想 2.2. 网络负载现象（Network load phenomenon）。 用时间间隔 Δ 表示在其间 i 个诚实节点正在创建和广播消息，否则每秒最多 if 条交易，在 Δ 间隔时间内新创建的图深度内可能共存的最大交易数为 i。

如果能利用这个现象，那就可以创建一个确定的，可验证的启发 (译者：verifable heuuristic)，使得节点可以有效逼近所属网络的负载或拥塞。

定义 2.16. 网络负载因子（Network load factor)。设猜想 2.2 中对于一个给定的交易 2.2，他的所有祖先交易的集合为 tx.ancestors。对于某一共识轮 $R_r = (r, \mathrm{tx_{start}}.\mathrm{depth}, \mathrm{tx_{end}}.\mathrm{depth})$ 我们可以估算其用于描述网络 N 中某一共识轮 r 下，节点拥塞负载因子 L_r 的计算公式为：

$$L_r = \frac{|\mathrm{tx_{end}}.\mathrm{ancestors}|}{\mathrm{tx_{end}}.\mathrm{depth} - \mathrm{tx_{start}}.\mathrm{depth}} \tag{3}$$

定理 2.1. 对手阻力 (Adversarial resistance)。 假设在猜想 2.2 中，用 A 表示一个对手，对手可以在某一共识轮 R_r 创建范围允许的任意深度交易，那么对手 A 也无法做到减小负载参数 L_r 的值，或者无法在一次有效交易中且父节点可以任意选取的情况下，使得参数减小超过 1。

证明. 任意一个对手 A 创建的交易 tx'，要么使得图 G 的深度增加 1，要么在 $\mathrm{tx'}.\mathrm{depth} \leq \mathrm{tx_{end}}.\mathrm{depth}$ 的条件下，使得图 G 中的深度为 tx'.depth 的交易个数，增加 1。对于前一情况， $|\mathrm{tx_{end}}.\mathrm{ancestors}|$ 和 $\mathrm{tx_{end}}.\mathrm{depth}$ 会简单加 1，而在后面一种情况会允许 L_r 的任意增加，结果等同于延展了图 G . 然而，如果考虑到在定义 2.1 中的父节点选取算法：一个诚实节点只会引用位于图中叶子节点的交易作为父节点，符合前一情况，那么一个诚实节点只可能选取来自对手 A 的，而且不符合后面一种情况的交易。因此，在一次交易中，不可能使得负载因子 L_r 增加超过 1，两种情况也不允许 L_r 的任意减值。 □

2.6 难度调整

根据定义 2.14. 中标记关键交易的条件，要么简单的调整系统参数 D_r 的值，要么指数级的增加或降低一个关键交易的创建频率。在交易种子的比特长度为 $|H|$ 时，一个关键交易的创建的概率为 $\frac{1}{2^{D_r}}$ ，其中 $D_r \leq |H|$ 。

为了线性调整关键交易的创建频率,我们根据两个系统参数 MIN_DIFFICULTY 和 DIFFICULTY_SCALE 对负载因子 L_r 做对数级的伸缩调整，使得：

$$D_r = \mathrm{MIN\_DIFFICULTY} + \mathrm{DIFFICULTY\_SCALE} * \log_2 L_r \tag{4}$$

对数级伸缩 L_r 能对关键交易的创建频率做线性调整， D_r 的增长率反比于 $\frac{1}{2^{D_r}}$。

为了搞清楚 MIN_DIFFICULTY 和 DIFFICULTY_SCALE 这两个参数的理想固定值，考虑，如果网络通信延迟越低，则网络瞬时最大负载越高。在一个零延迟的假想网络环境下，我们可以通过任意理想方法调整环境参数，使得协议暴露 5 个最小活性故障。

这个新派生的参数集是否允许在网络存在零延迟情况下，并且协议带有最小活性故障情况下，同时这些参数也可以安全地使协议在实际网络下的非零延迟环境中的运行。

这是由于：真实网络和零延迟网络的差别在网络的负载呈现会更多，因此，这些参数需要调整到即使在最坏情况也能满足协议执行。

3 设计

定义 3.1. Wavelet. Wavelet 是一个共识协议族，Wavelet 的各个独立变体通过元组表示为：$W = (\mathcal{P}_{\text{overlay}}, \mathcal{P}_{\text{gossip}}, \mathcal{P}_{\text{query}})$.

迈向实际的第一步，我们考虑一个 Wavelet 变体 W ，其中 $\mathcal{P}_{\text{overlay}}$ 采用 S/Kademlia 承载网络协议 [1]，$\mathcal{P}_{\text{gossip}}$ 是准 gossiping 协议，节点通过自身维护的 Kademlia 路由表进行信息交换， $\mathcal{P}_{\text{query}}$ 是 Snowball 共识协议 [12]

3.1 S/Kademlia

选择 S/Kademlia 承载网是由于他的 De-Brujin 网络拓扑结构，该拓扑结构允许使用 $O(\log n)$ 跳后将消息高效地发送到指定的接收方，其中 n 是网络中的节点数。

S/Kademlia 的静态和动态密码学谜题强制所有节点标识均匀分布，这强制要求每个节点单独维护的路由表都被均匀的分区，因此跨越诚实子网的通信能延续，符合定义 1.1 中的丢包，重定向，延迟发送的对手模型。

基于节点标识的均匀性，一个简单的 gossiping 协议可以构建在 S/Kademlia 之上，节点只需要按照自身的路由表，简单的在邻居节点间传播信息即可。假设用户至少知道一个可靠的引导节点，新用户可以很容易地加入成为网络成员。

3.2 Snowball

Snowball 是一个共识协议，其灵感来自于 Snowball 采样技术 [7] 的启发，该协议保证了在部分同步网络中，存在一个符合定义 1.1 的对手时，具有很强的安全性和活性。

引理 3.1. 强活性 (Strong liveness)。 给定一组具有 t-resilient 的 n 个处理器，Snowball 循环会最终将终止，其中 t 是每个处理器可以处理的可容忍故障数量。存在一个单独的 Snowball 循环终止条件，即我们连续成功的从 k 个对等节点上采样一个颜色 β 次 。基于最多 1/3 个处理器存在故障的假设，我们可以重整 Snowball 的活性刚性，并重新按照以下方式表达：

定义随机数 X ，满足任意离散分布，采样空间由两个元素构成： $\mathbf{R}$ 和 $\mathbf{B}$ （颜色），并且概率不包含零测度。求证一个对 X 的无穷采样会最终产生一个具有相同元素的 β 序列。

证明. 假设在一个具有 t-resilient 的 n 节点网络上，Snowball 的循环是不可能终止的。然而，回想一下“无限猴子” 定理 [8]：给定一个无限字符串序列，其中每个字符都是随机均匀挑选的，则任何给定的有限长度的字符串几乎可以肯定会作为这些字符串之一的前缀出现。

我们可以放宽从 X 中采样的字符服从均匀分布的假设，只要概率测度是非零的，基于第二个玻尔-坎特利引理 [3] 的逆命题结果，且每个采样事件都是独立的。

这样就出现了一个悖论：几乎可以肯定从 X 中无限采样的，会产生一个长度为 β 的具有完全一致元素的字符串。因此，假设网络通信是部分同步的，那么 Snowball 循环会最终肯定会终止。 □

引理 3.2. 强安全性（Strong safety）。 在所有 n 个处理器 Snowball 循环终止时，可以保证所有处理器在某个首选颜色 C 的值上达成一致。

证明. 根据引理 3.1，假设首选颜色 [公式] 在节点之间并不一致，然而，一般来说，首选颜色 C 在 Snowball 循环结束时，相比其他颜色，具有最大的计数值。这就产生了一个矛盾:首选的颜色 C 在所有节点上都是一致的。 □

根据 引理 3.1 和 3.2， Snowball 必须能够在由 n 个节点组成的部分同步网络上，以强安全性和活性实现二元共识。

然而，我们强调，即使从引理 3.1 得到的结果不能证明，在表示为对抗性模型的情况下，Snowball 在一个少量的，实际的迭代次数内达成强活性。尽管如此，考虑到 Snowball 本质上是一种采样机制，可以启用 (译者:bootstrapped) 各种各样的策略，可以让 Snowball 在少量的，有限的迭代次数内完成共识收敛。

例如，有一种策略包含了基于权益托管货币下的节点响应加权，以下详述，同时也能在对手存在的情况下实现动态调整 α 参数 [11]。

3.3 权益证明

节点可以押注 (译者：stake) 一定数量的虚拟货币用于获得更多的权重，辅助在共识轮的结算行为。有了这样一种机制，就可以为协议建立具有针对女巫攻击和日蚀攻击能力的协议，因为一个独立节点的权益是跨共识轮一致的。

将离散随机变量记作 $R_k \sim \mathrm{B}(k,p)$ 来将 k 个随机抽样的对等点的响应，建模为 k 个成功概率 p 的集合；以及对网络中 k 个节点权益分布，建模为离散随机变量 S_k。 按照乘积分布 $R' = S_k R_k$ ，我们可以宣称，成功的色彩采样事件满足 $\mathbf{E}[R' \geq \alpha]$ ， (译者：E 表示 event)。

3.4 交易手续费和奖励

一个拥有更多权重辅助共识轮结算的节点，也叫做*验证者 (译者：validator)*，自然能吸引更多的注意力，无论是来自于诚实节点，还是敌对节点。作为对这种情况的补偿，手续费会从每个已经被接受的交易中，在共识轮结算后进行扣除，并被分到多个辅助共识轮结算的验证者。

然而，许多问题出现了：我们如何能够使验证者在共识结算轮上的的付出和收获相匹配。为了解决这个问题，我们设计了一套激励机制，根据验证者的权益和的附加新交易时的活动，通过可验证的随机投票方案，对引用到尽可能多父节点的图 G 进行奖励。

为了说明这个系统是如何工作的，用 tx 表示一个刚刚被确认的交易，接下来：

1. 使 eligible = {tx'.sender ≠ tx.sender | tx' ∈ tx.ancestors} 。如果 |eligible| = 0 ，步骤终止。
2. 使 validators = {tx'.sender | tx' ∈ eligible} ，将所有 {tx'.id | tx' ∈ eligible} 拼接后进行 H 哈希函数处理，得到一个校验和，用作熵源 E。
3. 使 X' 作为某个阈值， X' 是 E 的最后 16 位，用一个随机变量 X 去模型化一个带权重的候选验证者加权均匀分布情况。我们选定一个满足权重 $X \geq X'$ 的验证者 $v \in$ validators，作为交易费奖励发放目标。
4. 从 tx.sender 转账一定数量的交易费用到预期的奖励发放目标 v 。

这样一个系统激励了验证者去主动的创建和广播新的交易，并从图 G 中选择大量的父交易，但并不简单的是，这个系统并不存在 *nothing-at-stake* 问题，考, 因为任何努力获得回报要求验证的验证器必须协助验证大量的、指数级增长的交易。

奖励发放模式结合了 S/Kademlia 承载协议 $\mathcal{P}_{\text{overlay}}$ ，自然而然的推延了: “凡有的，还要加给他，叫他有余；凡没有的，连他所有的也要夺去。“描述的马太效应困境，因为网络拓扑决定了并不会区分对待节点，因此对等节点必定会和具有高权益的验证者交互。

4 讨论

假设网络上的通信是部分同步的，我们将讨论第 3 节中描述的拟议协议设计中需要考虑的几个关键问题。

定理 4.1. 分叉自愈 (Fork resilience)。 如果在某个时间点上，某一轮 R_i 已经完成确认，但两个节点的视图不同，那么我们说这两个节点产生分叉 *(fork)* 或分离 *(diverge)*。给定两个诚实的节点 N_1 和 N_2，在具有可容忍的对抗性节点数的网络中，节点 N_1 和 N_2 不可能产生分叉。

证明. 根据定理 2.1： 诚实节点只构建这种交易，即交易克制的引用在某个深度 d 上交易数量会有意增加的父交易。因此，诚实节点为结束一轮 R_i 而提议的关键交易，只会引用由其他诚实节点创建的父交易，这些交易是不会延展图 G 的。因此，敌对节点不可能构建由可靠节点维护的交易分叉图 G，这些节点由轮 R_i 的终点交易的父交易引用。

现在，考虑下诚实节点活动中的分叉自愈，假设 N_1 和 N_2 的图 G 并不相等，并且它们中的任何一个提议了一个关键交易用于终止一轮 R_i 。用符号 N_1' 表示没有提交关键交易的其中一个，另一个提交了的记为 N_2' 。按照定义 2.15： N_1' 构建默克尔根会由于 N_2' 提交了一个不相等的图而失效。只有在 N_1' 和 N_2' 图相似的情况下，Snowball 共识协议可以允许 N_1' 对 N_2' 的提议发起一次投票以终结一轮 R_i 。因此，如果两个节点要分叉，就违背了 Snowball 共识协议的安全属性。

因此，既然诚实节点在收到任意节点发出的交易的情况下不可在 R_i 轮产生内容分歧，因此在一个具有可容忍的敌对节点数量的网络中，两个节点 N_1 和 N_2 不可能产生分叉。 □

然而，由于定理 4.1 的结果，诚实节点的良性交易可能不会在其创建的轮内完成。这将导致随后被拒绝的交易必须重新签名，并用新选择的父交易重新提交到网络，这很麻烦。

这是留给未来的工作：描述一个解决方案，使协议针对这样的问题更能自愈。在全网范围内，对构建图 G 形成激励，例如通过虚拟货币或计算资源是解决这个问题的可行方法。

5 Results

第 3 节中描述的 Wavelet 变体 W 在部分同步网络下建立了以下保证:

定义 5.1. 交易的全局顺序（Total ordering of transactions）。何两个具有各自维护的图的节点都将以完全相同的顺序最终确认并应用相同的接受交易集。

定义 5.2. 交易不可逆性（Transaction irreversibility）。所有深度小于或等于任何最终协商一致回合结束点深度的交易都被标记为可接受和可最终确认，并且根据协议，不得撤销其已接受或已最终确认交易。

这些保证使得 Wavelet 变体 W 具有一些有趣的前景和特性。

定义 5.3. 简洁性 (Succinct)。所有最终确定的交易都可以安全地进行修剪，因为根据协议，最终确定的交易是不可逆转的。假设一个支付交易是几百字节级别，那么每个节点维护的完整交易图可以完全存储在内存中。这使得可充分验证的 Wavelet 节点可以在低端硬件上运行，只需要至少 512MB 的动态内存，比如智能手机。

定义 5.4. 全局性 (Global-scale)。当活性故障发生时，协议会根据网络负载自动调整自身的系统参数，以防止活性故障的进一步恶化。基准测试报告，在一个由 240 个节点组成的网络上，5 秒的移动窗口平均（Moving Window Average）为每秒完成 31,240 个交易。节点使用部署在 DigitalOcean 上 Kubernetes 集群上的消费级硬件 (2vCPUs, 4GB RAM)，平均通信延迟为 220 毫秒，同时使用 Linux 工具 **tc**(**1**) 模拟 2%的数据包丢失。每个单独的节点限定最多只能启动一个单独的 Wavelet 实例，封装到一个 Kubernetes pod，并且最多以 890kb/s 的速率产生出站流量。每个 Wavelet 实例包含一个 Snowball 实例，按照 $k = 10$, $\alpha = 0.8$, and $\beta = 150$ 的参数进行共识轮结算，每个传入的交易经过全加密签名验证。每个 Wavelet 实例也被参数化，系统参数设置为：DEPTH_DIFF = 10, MAX_NUM_PARENTS = 32, MIN_DIFFICULTY = 8, and DIFFICULTY_SCALE = 0.5. 基准测试任务是让所有节点创建并向网络中注入由 40 个独立的权益分配交易组成的批量交易，以考虑每个节点在维护其运行状态 S 时，Wavelet 所面临的延迟。基准测试开始时，在收集统计数据之前，会考虑一个 25 分钟的热身时间。在一个通信延迟可以忽略的网络中，仅为了将私有区块链托管在一起，Wavelet 可以用相同的基准测试任务平均每秒完成 18 万多个交易。

定义 5.5. 智能合约 (Smart contracts)。考虑到交易的顺序和应用程序在所有节点上是一致的，图灵完备的智能合约可能会得到安全的支持，并可以确定地执行。作为第一步，基于 WebAssembly 虚拟机的智能合约，需要确定性软件仿真浮点数 (译者：software-emulated-floating-point) 和基于确定性控制流图（译者：control-flow-graph-based）的燃料 (gas) 计数支持（译者：这是智能合约虚拟机的可重现性需求，可以参考作者另一篇关于 Cartesi 可重现虚拟机的内容）。在调用智能合约函数之前，解释器的内存快照存储在内存中。在调用智能合约函数之后，调用后解释器的内存和调用前解释器的内存快照之间的差异将持久化到账本上。

定义 5.6. 权益证明（Proof of stake）回想一下 Snowball 共识协议 $\mathcal{P}_{\text{query}}$ 和 3.3 节中提到的权益证明机制。将 R_k 与 S_k 进行卷积，通过虚拟货币的权益衡量 $\mathcal{P}_{\text{query}}$ 最终终止的进度，从而允许 Wavelet 变体 W 自然地建立起女巫攻击抗性。

定义 5.7. 公平激励（Fair incentives）。回想 3.4 所述的奖励分配计划。协议自然要求验证者必须努力协助达成共识，以防止出现 “nothing-at-stake” 的问题，并延缓富者越富，贫者越贫的马太效应的发生。

定义 5.8. 去中心化治理（Decentralized governance）。该协议形成了一个网络拓扑结构，其中财富非常均匀地分布在节点之间，以鼓励分散的治理形式。历史的不可逆性保证了，协议范围的协商共建能有效进行，如渐进式的协议更新和系统参数调优。

6 结论

自从比特币问世以来，已经过去了 10 年，它的起源催生了越来越多的新技术和应用。修正和推广比特币所依赖的框架的过程，使本文的作者引入了一种新的实用协议家族; 推动中本聪 (Satoshi Nakamoto) 的愿景，迈向去中心化未来的愿景。

Wavelet 强调最小配置、最小资源消耗和最小系统需求，以禁止只有富人才能从这种分散的网络中获益。还将允许建立、采用和探索多种去中心化的应用程序。

本文还向读者介绍了一组小的猜想，这些猜想经过探索很容易有助于开发有前途的、令人兴奋的、新的数学工具和框架。

然而，开发通用账本框架的最后一个小问题尚未得到解决:随着时间的推移，节点状态 S 可能增长得过大，使磁盘空间成为瓶颈。作者想得出的结论是，解决这一问题的办法已经想到，这将在即将发表的一篇题为《**Perlin**》的论文中介绍

致谢

两位作者想感谢中本聪 (Satoshi Nakamoto) 对比特币的研究，他早在本世纪初就对两位作者的生活产生了重大影响。还想感谢 Dimitris Papadopolous、Foteini Baldimtsi 和 Josh Lind 的宝贵意见和反馈。

同时在此感谢译者傅理 (知乎 ID: xtaci)。

参考文献

[1] Ingmar Baumgart and Sebastian Mies. S/kademlia: A practicable approach towards secure key-based routing. In *2007 International Conference on Parallel and Distributed Systems*, pages 1–8. IEEE, 2007.

[2] Ethan Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, 2016.

[3] Kai Lai Chung and Paul Erdös. On the application of the borel-cantelli lemma. *Transactions of the American Mathematical Society*, 72(1):179–186, 1952.

[4] Pierre-Jacques Courtois, Frans Heymans, and David Lorge Parnas. Concurrent control with "readers" and "writers" . *Communications of the ACM*, 14(10):667–668, 1971.

[5] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[6] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

[7] Leo A Goodman. Snowball sampling. *The annals of mathematical statistics*, pages 148–170, 1961.

[8] Richard Isaac. *The pleasures of probability*. Springer Science & Business Media, 2013.

[9] Dan Larimer. Delegated proof-of-stake consensus, 2018.

[10] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[11] Serguei Popov. On fast probabilistic consensus in the byzantine setting, 2019.

[12] Team Rocket. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies, 2018.