电报开放网络 *

尼古拉·杜罗夫博士

2018年1月18日

v1.0

提要

本文的目的是提供电报开放网络(TON)和相关区块链，点对点，分布式存储和服务托管技术的第一个描述。 为了将本文章缩小到合理的长度，我们主要关注电报开放网络(TON)平台的独特和定义功能，这些功能对于实现其既定目标非常重要。

导言

电报开放网络(TON)是一个快速，安全且可扩展的区块链和网络项目，如果需要，每秒能够处理数百万次交易，并且对用户友好且对服务提供商友好。 我们的目标是让它能够容纳目前提出和构思的所有合理应用程序。 人们可能会认为电报开放网络(TON)是一个巨大的分布式超级计算机，或者更确切地说是一个巨大的《超级服务器》，为了托管和提供各种服务。

本文无意作为所有实施细节的最终参考。 在开发和测试阶段，一些细节可能会发生变化。

* 本文的此版本作为附录A附加于入门书。

内容

1 电报开放网络(TON)组件简介

电报开放网络(*TON*)是以下组件的组合：

- 灵 灵活的多区块链平台（电报开放网络(*TON*)区块链;参见第 **2** 章），能够每秒处理数百万笔交易，具有图灵完备智能合约，可升级的正式区块链规范，多加密货币价值转移，支持微支付渠道和脱链付款网络。 电报开放网络(*TON*)区块链提供了一些新的独特功能，例如《自我修复》垂直区块链机制（参见 **2.1.17** ）和即时超立方体路由（参见 **2.4.20** ），使其快速，可靠，可扩展， 同时自洽。

- 点对点网络（*电报开放网络*(*TON*)*P2P*网络，或仅电报开放网络(*TON*);参见第 **3** 章），用于进接电报开放网络(*TON*)区块链，发送交易候选者，以及仅接收客戶感兴趣的区块链部分的 更新（ 例如，与客戶账戶和智能合约相关的那些部分），但也能够支持任意分布式服务，不管是不是与区块链相关。

- 分布式文件存储技术（电报开放网络(*TON*)存储;参见 **4.1.8** ），可通过电报开放网络(*TON*)进接，由电报开放网络(*TON*)区块链用于存储块和状态数据（快照）的存档副本，但也可用于为用戶或为其他在平台上运行的服务存储任意文件，具有类似洪流的进接技术。

- 网络代理/匿名者层（电报开放网络(*TON*)代理;参见 **4.1.11** 和 **3.1.6** ），类似于I^2P（隐形网计划），用于在必要时隐藏电报开放网络(*TON*)节点的身份和IP地址（例如，有节点从具有大量加密货币的账戶提交交易，或者高权益区块链验证者节点，其希望隐藏其确切的IP地址和地理位置作为针对DDoS攻击的措施）。

- 类似Kademlia的分布式哈希表（电报开放网络(*TON*)*DHT*;参见 **3.2** ），用作电报开放网络(*TON*)*存储的*《洪流跟踪器》（参见 **3.2.10** ），作为电报开放网络(*TON*)代理的《输入隧道定位器》（参见 **3.2.14** ），以及作为电报开放网络(*TON*)服务的服务定位器（参见 **3.2.12** ）。

- 任意服务的平台（电报开放网络(*TON*)服务;参见第 **4** 章），可以通过电报开放

网络(TON)和电报开放网络(TON)代理得到，并可通过形式化接口（参见 **4.3.14** ）实现类似浏览器或智能手机应用程序的交接。 这些正式接口和持久服务入口点可以在电报开放网络(TON)区块链中发布（参见 **4.3.17** ）；在电报开放网络(TON)区块链中发布的信息（参见 **3.2.12** ），可以通过电报开放网络(TON)分布式哈希表查找在任何给定时刻提供服务的实际节点。 服务可以在电报开放网络(TON)区块链中创建智能合约，为其客戶提供一些保证（参见 **4.1.7** ）。

- 电报开放网络(TON)域名系统（参见 **4.3.1** ），一种为（各种）帐戶，智能合约，服务和网络节点分配人们可读名称的服务。

- 电报开放网络(TON)付款（参见第 **5** 章），小额支付，微支付渠道和微支付渠道网络的平台。它可用于快速的脱链价值转移，以及支付由电报开放网络(TON)服务提供支持的服务。

- 电报开放网络(TON)将允许与第三方信息传递和社交网络应用程序轻松集成，从而使区块链技术和分布式服务最终可供普通用戶进接（参见 **4.3.24** ），而不仅是少数早期的加密货币采用者。 我们将在另一个项目（电报信使）中提供这种集成的示例（参见 **4.3.19** ）。

虽然电报开放网络(TON)区块链是电报开放网络(TON)项目的核心，而其他组件可能被视为区块链的支持角色，但它们本身就具有有用也有趣的功能。 结合起来，它们允许平台承载比仅使用TON区块链（参见 **2.9.13** 和 **4.1** ）更多样化的应用程序。

2 电报开放网络(TON)区块链

我们首先描述电报开放网络(TON)区块链，这是项目的核心组成部分。 我们的方法是《从上到下》：我们首先给出整体的一般描述，然后提供每个组件的更多细节。

为简单起见，我们在这里谈论电报开放网络(TON)区块链，尽管原则上该区块链协议的几个实例可以独立运行（例如，由于硬分叉）。 我们只考虑其中一个。

2.1 电报开放网络(TON)区块链作为两区块链的集合

电报开放网络(TON)区块链实际上是区块链的集合（甚至是区块链的区块链的集合，或者是两区块链——这一点将在后面的 **2.1.17** 中说明），因为没有一个区块链项目能够实现我们的目标：每秒处理数百万次交易，而不是现在的标准，每秒数十次交易。

2.1.1 区块链类型列表。 这个系列中的区块链是：

- 独特的主区块链，或简称主链，包含一般信息，关于：协议及其参数的当前值，验证者及其权益的集合，当前活动的工作链集及其《分片》，最重要的是， 所有工作链和分片链的最新块的哈希集。

- 几个（最多2^{32}个）运转的（或正在工作的）区块链，或简称工作链，实际上是《主力》，包含价值转移和智能合约交易。 不同的工作链可能具有不同的《规则》，意味着不同格式的帐户地址，不同的交易格式，用于智能合约的不同虚拟机（VMs），不同的基本加密货币等。 但是，它们都必须满足某些基本的互操作性标准，以使不同工作链之间的交互成为可能并且相对简单。 在这方面，电报开放网络(TON)区块链是异构的（参见 **2.8.8** ），类似于EOS（参见 **2.9.7** ）和波卡（参见 **2.9.8** ）项目。

- 每个工作链又被细分为多达2^{60}个分片区块链，或简称为分片链，具有与工作链本身相同的规则和块格式，但仅负责一个帐户子集，取决于帐户地址的几个前面的（最重要的）位。 换句话说，系统中内置了一种分片形式（参见 **2.8.12** ）。 因为所有这些分片链共享一个共同的块格式和规则，所以电报开放网络(TON)区块链在这方面是同构的（参见 **2.8.8** ），类似于以太坊扩展提议中讨论的内容。[1]

[1] https://github.com/ethereum/wiki/wiki/Sharding-FAQ

- 分片链（和主链）中的每个块实际上不仅仅是一个块，而是一个小区块链。 通常，这个《块 区块链》或《垂直区块链》只包含一个块，然后我们可能会认为这只是分片链的相应块（在这种情况下也称为《横向区块链》）。但是，如果有必要修复不正确的分片链块，则会将新块提交到《垂直区块链》中，其中包含无效《横向区块链》块的替换或《块区别》，仅包含需要更改的此块的先前版本的那些部分的描述。这是一种特定于电报开放网络(TON)的机制，用于替换检测到的无效块，而不会作出所有被涉及的分片链的真正分叉; 它将在 **2.1.17** 中更详细地解释。 现在，我们只是说每个分片链（和主链）不是传统的区块链，而是区块链的区块链，或两维区块链，或者只是一个两区块链。

2.1.2. 无限分片范式。几乎所有的区块链分片提议都是《从上到下》的：首先想象一个区块链，然后讨论如何将其拆分成几个交互的分片链，以提高性能并实现可扩展性。

电报开放网络(TON)的分片方法是《从下到上》，解释如下。

想象一下，分片已经发挥到极端，因此每个分片链中只保留一个帐户或智能合约。 然后我们有大量的《账户链》，每个账户链描述一个账户的状态和状态转换，并相互发送价值信息以传输价值和信息。当然，拥有数亿个区块链是不切实际的，其中每个区块链通常很少出现更新（即新区块）。为了更有效地实现它们，我们将这些《帐户链》分组为《分片链》，以便分片链的每个块本质上是已分配给此分片的帐户链块的集合。 因此，《帐户链》在《分片链》中仅具有纯粹的虚拟或逻辑存在。

我们将这种观点称为无限分片范式。 它解释了电报开放网络(TON)区块链的许多设计决策。

2.1.3. 信息。 即时超立方体路由。 无限分片范式指示我们将每个帐户（或智能合约）视为自己的分片链。 然后，一个帐户可能影响另一个帐户状态的唯一方法是向其发送信息（这是所谓的参与者模型的特殊实例，帐户为参与者;参见 **2.4.2** ）。 因此，帐户（和分片链之间的信息系统，因为来源账户和目标帐户通常位于不同的分片链中）对于诸如电报开放网络(TON)区块链的可扩展系统是至关重要的。 实际上，电报开放网络**(TON)**区块链的一个新功能，称为即时超立方体路由（参见 **2.4.20** ），使其能够将在一个分片链块中创建的信息传递和处理到目标分片链的下一个块中，而不管系统中分片链的总数。

2.1.4. 主链，工作链和分片链的数量。 电报开放网络(TON)区块链只包含一个主链。 但是，该系统最多可容纳2^{32}个工作链，每个工作链最多可分为2^{60}个分片链。

2.1.5. 工作链可以是虚拟区块链，而不是真正的区块链。 因为工作链通常被细分为分片链，所以工作链的存在是《虚拟的》，这意味着它不是真正的区块链，如在下面的 **2.2.1** 中提供的一般定义所指示的那样， 它只是一组分片链。 当只有一个分片链对应一个工作链时，这个独特的分片链可以用工作链识别，在这种情况下，工作链至少在一段时间内成为《真正的》区块链，从而获得与传统单区块链设计的表面相似性。 然而，无限分片范式（参见 **2.1.2** ）告诉我们，这种相似性确实是表面的：只是巧合的是，潜在的大量《帐户链》可以暂时归为一个区块链。

2.1.6. 工作链的识别。 每个工作链由其编号或工作链标识符（工作链_id：uint_{32} ）标识，它只是一个无符号的32位整数。 工作链由主链中的特殊交易创建，定义（以前未使用的）工作链标识符和工作链的正式描述，至少足以使该工作链与其他工作链的交互以及对该工作链块的表面验证。

2.1.7. 创建和激活新的工作链。 基本上任何社区成员都可以创建新的工作链，准备支付发布新工作链正式规范所需的（高）主链交易费用。 但是，为了使新的工作链变得活跃，需要三分之二的验证者共识，因为他们需要升级他们的软件来处理新工作链的块，并表示他们准备好以特殊的主链交易使用新工作链。对激活新工作链感兴趣的一方可能会为验证者提供一些激励，通过智能合约分发的一些奖励来支持新的工作链。

2.1.8. 鉴定分片链。 每个分片链由一对（*w, s*）=（工作链_*id*, 分片_前缀）标识，其中工作链_*id*：$uint_{32}$ 标识相应的工作链，分片_前缀：$2^{0..60}$ 是一个长度最多为60的位串，定义了这个分片链负责的子集帐户。 即，所有具有以分片_前缀开头的账户_*id*的帐户（即，将分片_前缀作为最高有效位）将被分配给该分片链。

2.1.9. 识别帐户链。 回想一下，帐户链只有虚拟存在（参见 **2.1.2** ）。 但是，它们有一个自然的标识符——即（工作链_*id*, 账户_*id*）——因为任何帐户链都包含有关状态的信息和一个帐户的更新（简单帐户或智能合约——这里的区别并不重要）。

2.1.10. 分片链的动态分裂和合并;参见 2.7 。 不太复杂的系统可能使用静态分片 - 例如，通过使用账户_*id* 的前八位来选择256个预定义分片中的一个。

电报开放网络(TON)区块链的一个重要特征是它实现了动态分片，这意味着分片数量不固定。 相反，如果满足某些形式条件，分片 (*w, s*) 可以自动细分为分片 (*w, s.0*) 和 (*w, s.1*) （基本上，如果始发分片上的交易负载在很长一段时间内足够高）。 相反，如果负载在一段时间内保持太低，则分片 (*w, s.0*) 和 (*w, s.1*) 可以自动合并为分片 (*w, s*) 。

最初，只为工作链w创建一个分片 $(w, \emptyset)$。 之后，如果有必要，它会细分为更多分片（参见 **2.7.6** 和 **2.7.8** ）。

2.1.11. 基本工作链或零工作链。 虽然可以使用特定的规则和交易定义多达 2^{32} 个工作链，但我们最初只定义了一个工作链，其中工作链 $id = 0$。 此工作链称为零工作链或基本工作链，是用于处理电报开放网络(TON)智能合约和转移电报开放网络(TON)代币的工作链。这些货币也称为*Gram*（参见附录 **A** ）。 大多数应用程序可能只需要零工作链。基本工作链的分片链将被称为基本的分片链。

2.1.12. 块生成间隔。 我们期望在每个分片链和主链中生成一个新块，大约每五秒生成一次。 这将导致相当小的交易确认时间。 所有分片链的新块几乎同时生成; 大约一秒后生成一个新的主链块，因为它必须包含所有分片链的最新块的哈希值。

2.1.13. 使用主链使工作链和分片链紧密耦合。 一旦分片链块的哈希被合并到主链的块中，该分片链块及其所有祖先被认为是《规范的》，这意味着它们可以从所有分片链的后续块中被引用为固定和不可变的东西。 实际上，每个新的分片链块包含最新主链块的哈希，并且从该主链块引用的所有分片链块被新块视为不可变的。

本质上，这意味着在分片链块中提交的交易或信息可以安全地用在其他分片链的下一个块中，而不需要等待，例如，二十个确认（即，在相同的区块链里的始发块之后生成的二十个块） 之后而转发信息或基于先前的交易采取其他措施，这在大多数提议的《松散耦合》系统（参见 **2.8.14** ）中是常见的，例如EOS。 这种在其他分片链中提交后仅仅五秒钟就使用交易和信息的能力就是为什么我们认为我们的《紧密耦合》系统（这样的系统中的首次第一）能够提供前所未有的性能（参见 **2.8.12** 和 **2.8.14** ）。

2.1.14. 主链块哈希作为全局状态。 根据 **2.1.13** ，最后一个主链块的哈希完全从外部观察者的角度确定系统的整体状态。 不需要分别监控所有分片链的状态。

2.1.15. 验证者生成新块;参见 2.6 。 电报开放网络(TON)区块链使用权益证明方法在分片链和主链中生成新块。 这意味着有一组验证者，比方说，最多几百个验证者——特殊节点通过特殊的主链交易存入权益（大量电报开放网络(TON)货币），以便有资格进行新的块生成和验证。

然后，以确定性伪随机方式将较小的验证者子集分配给每个分片 (w, s) ，大约每1024个块改变一次。 这个验证者子集通过从客户端收集合适的建议交易给新的有效块候选者，建议并就下一个分片链块的内容达成共识。 对于每个块，在验证者上存在伪随机

选择的顺序，以确定哪个块候选者具有在每个转弯处提交的最高优先级。

验证者和其他节点检查提议的块候选者的有效性; 如果验证者签署了无效区块候选者，则可能会因丢失其部分或全部权益而被自动处罚，或者在一段时间内从该验证者组暂停。 在此之后，验证者应该就下一个区块的选择达成共识，主要是通过BFT（拜占庭容错;参见 **2.8.4** ）共识协议的有效变体，类似于PBFT [7]或 蜜獾 BFT [18] [2] 。如果达成共识，则创建一个新块，验证者在它们之间划分所包含交易的交易费用，以及一些新创建的（《铸造的》）代币。

可以选择每个验证者参与几个验证者子集; 在这种情况下，期望每个验证者并行运行所有验证和一致性算法。

生成所有新的分片链块或传递超时后，将生成一个新的主链块，包括所有分片链的最新块的哈希值。 这是由所有验证者的拜占庭容错共识完成的。 [3]

有关电报开放网络(TON)权益证明方法及其经济模型的更多详细信息，请参见第 **2.6** 节。

2.1.16. 主链的分叉。 由我们的紧耦合方法产生的一个复杂结果是，在主链中切换到不同的分叉几乎必然需要在至少一些分片链中切换到另一个分叉。 另一方面，只要主链中没有分叉，分片链中的分叉甚至都不可能，因为分片链的替代叉中的块不会通过将它们的哈希结合到主链块中而变成《规范》。

一般规则是，如果主链块 B' 是 B 的前身，则 B' 包括 (w, s) 分片链块 $B'_{w,s}$ 的哈希 HASH($B'_{w,s}$) ，并且 B 包括哈希HASH($B_{w,s}$)，那么 $B'_{w,s}$ 必须是 $B_{w,s}$ 的前身; 否则，主链块 B 无效。

我们期望主链分叉很少见，几乎不存在，因为在有关电报开放网络(TON)区块链采用的拜占庭容错范例中，它们只能在大多数验证者有错误行为的情况下发生（参见 **2.6.1** 和 **2.6.15** ） ，这将意味着违规者的重大权益损失。因此，应该期望分片链中没有真正的分叉。相反，如果检测到无效的分片链块，它将通过两区块链（参见 **2.1.17** ）的《垂直区块链》机制进行纠正，可以不需要分叉《横向区块链》（即 ，分片链）而实现这一目标。 同样的机制也可用于修复主链块中的非致命错误。

2.1.17. 更正无效的分片链块。 通常，只提交有效的分片链块，因为分配给分片链的验证者在提交新块之前必须达到三分之二的拜占庭共识。 但是，系统必须允许检测先前提交的无效块及其校正。

当然，一旦找到无效的分片链块——由验证者（不一定分配给此分片链的验证者）或《渔民》（系统里进行某种存款以便能够提出关于块有效性的问题的任何节点; 参

[2] 基于流言的拜占庭共识协议，类似于哈希图[1]，也可能与此目的相关。
[3] 实际上，三分之二的权益足以达成共识，但努力收集尽可能多的签名。

见 **2.6.4** ）——断言无效及其证据被提交到主链中，签署无效区块的验证者将受到惩罚: 失去部分权益和（或）暂时停止使用验证者（后措施对于攻击者窃取良性的验证者——验证者除此之外是良性——的私人签名密钥的情况，很重要）。

但是，这还不够，因为系统的整体状态（有关电报开放网络(TON)区块链）由于先前提交的分片链块无效而证明是无效的。 必须使用较新的有效版本替换此无效块。

大多数系统通过《回滚》到此分片链中的无效块之前的最后一个块来实现这一点，并且最后一个块不受从其他每个分片链中的无效块传播的信息的影响，并从这些块创建新的分叉。 这种方法的缺点是突然回滚大量除此之外正确和已提交的交易，并且不清楚它们是否稍后将被包括在内。

电报开放网络(TON)区块链通过使每个分片链和主链（《横向区块链》）的每个《块》自身成为一个小区块链（《垂直区块链》）来解决这个问题，包含这《块》的不同版本或它们的《差异》。通常，垂直区块链只包含一个块，而分片链看起来像是一个经典的区块链。 然而，一旦确认块的无效并将其提交到主链块中，则允许无效块的《垂直区块链》在垂直方向上由新块增长，并且替换或编辑无效块。 新块由相关分片链的当前验证者子集生成。

新《垂直》区块的有效规则非常严格。 特别是，如果无效块中包含的虚拟《帐户链块》（参见 **2.1.2** ）本身有效，则必须保持它不被新的垂直块改变。

一旦在无效块之上提交了新的《垂直》块，其哈希就会发布在新的主链块中（或者更确切地说是在新的《垂直》块中，位于始发主链块上方，在无效的分片链块的哈希值最初发布的地方），并且更改被进一步传播到任何描述该块的先前版本的分片链块（例如，那些已从不正确的块接收信息的那些分片链块）。这是通过在垂直区块链中为以前引用“不正确”区块的所有区块提交新的《垂直》区块来解决的; 新的垂直块将引用最新（已更正）的版本。 同样，严格的规则禁止更改未受影响的帐户链（即，接收与先前版本相同的信息的帐户链）。 通过这种方式，修复不正确的块会产生《细浪》，最终传播到所有受影响的分片链的最新块; 这些变化也反映在新的《垂直》主链块中。

一旦《历史重写》细浪到达最近的块，新的分片链块仅在一个版本中生成，仅作为最新块版本的后继者。 这意味着它们将从一开始就包含对正确（最近）垂直块的引用。

主链状态隐含地定义了将每个《垂直》区块链的第一个块的哈希转换为其最新版本的哈希的映射。 这使客户端能够通过其第一个（通常是唯一的）块的哈希来识别和定位任何垂直区块链。

2.1.18. 电报开放网络(TON)代币和多货币工作链。 电报开放网络(TON)区块链支持多达2^{32}种不同的《加密货币》，或《代币》，由有32位数的货币_*id*区分。 可以通过主链中的

特殊交易添加新的加密货币。 每个工作链都有一个基本的加密货币，并且可以有几个额外的加密货币。

有一种特殊的加密货币，货币_*id* = 0，即TON代币，也称为*Gram*（参见附录 **A** ）。 它是零工作链的基本加密货币。 它还用于交易费和验证者权益。

原则上，其他工作链可能会以其他代币收取交易费用。 在这种情况下，应提供一些智能合约，将这些交易费自动转换为Gram。

2.1.19. 信息传递和价值转移。 属于相同或不同工作链的分片链可以相互发送信息。 虽然允许的信息的确切形式取决于接收工作链和接收帐户（智能合约），但是有一些共同的字段使得工作链间信息传递可能。 特别地，每个信息可以以一定量的Gram（电报开放网络(TON)代币）和（或）其他注册的加密货币的形式附加一些值，只要它们被接收工作链声明为可接受的加密货币。

这种信息传递的最简单形式是从一个（通常不是智能合约）账户到另一个账户的价值转移。

2.1.20. 电报开放网络(TON)虚拟机。 电报开放网络(TON)虚拟机（也称为*TON VM*或*TVM*）是用于在主链和基本工作链中执行智能合约代码的虚拟机。 其他工作链可以与*TVM*一起使用其他虚拟机或代替TVM使用其他虚拟机。在这里，我们列出了它的一些功能。 它们将在 **2.3.12** ， **2.3.14** 和其他地方进一步讨论。

- TVM将所有数据表示为(TVM)细胞的集合（参见 **2.3.14** ）。 每个细胞最多包含一百二十八个数据字节，最多包含四个对其他细胞的引用。 由于《一切都是一袋细胞》的理念（参见 **2.5.14** ），这使TVM能够处理与电报开放网络(TON)区块链相关的所有数据，如有必要包括块和区块链全局状态。

- TVM可以处理任意代数数据类型的值（参见 **2.3.12** ），表示为TVM细胞的树或有向非循环图。 但是，它对代数数据类型的存在是不可知的; 它只适用于细胞。

- TVM内置了对哈希图的支持（参见 **2.3.7** ）。

- TVM是一台堆栈机器。 它的堆栈保持64位整数或细胞引用。

- 支持64位，128位和256位算术。 所有 *n* 位算术运算有三种形式：无符号整数，

有符号整数和模 2^n 整数（后一种情况下不进行自动溢出检查）。

- *TVM*具有从 n 位到 m 位的无符号和有符号整数转换，对于所有$0 \leq m, n \leq 256$，具有溢出检查。
- 默认情况下，所有算术运算都执行溢出检查，大大简化了智能合约的开发。
- TVM具有《乘后移位》和《移位后除》算术运算，其中间值以较大的整数类型计算; 这简化了实现定点算术的过程。
- TVM支持位串和字节串。
- 支持256位椭圆曲线加密(ECC)用于某些预定义曲线，包括Curve25519和Ed25519。
- 在一些椭圆曲线上支持Weil，Tate或相关配对，对于基于配对的加密和zk-SNARK的快速验证[9]也很有用。[4]
- 支持流行的哈希函数，包括SHA256。
- TVM可以使用默克尔树证明（参见 **5.1.9** ）。
- TVM为《大型》或《全局》智能合约提供支持。 这种智能合约必须意识到分片（参见 **2.3.18** 和 **2.3.16** ）。 通常（本地）智能合约可能与分片无关。
- TVM支持结算。
- 可以在TVM内轻松实现《无骨架无标签G机》[20]。

除了《TVM组件》之外，还可以为TVM设计几种高级语言。 所有这些语言都将具有静态类型，并将支持代数数据类型。 我们设想了以下可能性：

- 一种类似Java的命令式语言，每个智能合约类似于一个单独的类。

[4] 更具体地说，这种支持可能包括针对预定义的一组参数值计算Barreto-Naehrig曲线[2]上的Ate配对[12]。

- 一种懒惰的函数式语言（想想Haskell）。
- 一种热切的功能语言（想想ML）。

2.1.21. 可配置的参数。 电报开放网络(TON)区块链的一个重要特征是它的许多参数都是可配置的。 这意味着它们是主链状态的一部分，并且可以通过主链中的某些特殊提议或投票或结成的交易进行更改，而无需任何硬分叉。 改变这些参数将需要收集三分之二的验证者投票和所有其他参与投票过程以支持该提议的参与者的一半以上的投票。

2.2 区块链的一般性

2.2.1. 一般区块链定义。 通常，任何（真实的）区块链是一系列块，每个块 B 包含前一个块的参考 blk-prev(B)（通常通过将前一个块的哈希包括在当前块的标头中），以及 交易清单。 每个交易描述了全局区块链状态的一些转换; 按顺序应用块中列出的交易以计算从旧状态开始的新状态，这是在评估前一个块之后的结果状态。

2.2.2. TON区块链的相关性。 回想一下，电报开放网络(TON)区块链不是真正的区块链，而是两区块链的集合（即区块链的区块链;参见 **2.1.1** ），因此上述内容并不直接适用于它。 但是，我们从真正的区块链的这些一般性开始，将它们用作我们更复杂结构的构建块。

2.2.3. 区块链实例和区块链类型。 人们常常使用区块链这个词来表示一般的区块链类型及其特定的区块链实例，它们被定义为满足某些条件的区块序列。 例如， **2.2.1** 指的是区块链实例。

以这种方式，区块链类型通常是块的列表（即，有限序列）的块*的类型的《子类型》，由满足某些兼容性和有效性条件的那些块序列组成：

$$\text{区块链} \subset \text{块}^* \tag{1}$$

定义**Blockchain**的更好方法是说**Blockchain**是一对依赖的耦合类型，由夫妻($\mathbb{B}$, v)组成，第一组成部分$\mathbb{B}$：块*是块*的类型（即块的列表），第二个组件 v：$isValidBc(\mathbb{B})$是$\mathbb{B}$的有效性的证据或见证。这样，

$$\text{区块链} = \Sigma_{(\mathbb{B}:\text{块}^*)} isValidBc(\mathbb{B}) \tag{2}$$

我们在这里使用从[23]借来的依赖类型的表示法。

2.2.4. 依赖型理论，Coq和TL。 请注意，我们在这里使用（马丁-洛夫）依赖型理论，类似于Coq [5]校对助手中使用的理论。 依赖类型理论的简化版本也用于TL（类型语言）[6]，将用于电报开放网络(TON)区块链的形式规范，以描述所有数据结构的序列化以及块，交易等的布局。

实际上，依赖类型理论给出了证据的有用形式化，例如，当需要为某些块提供无效证明时，这种形式证明（或它们的序列化）可能变得很方便。

2.2.5. TL，或类型语言。 由于TL（类型语言）将用于电报开放网络(TON)块，交易和网络数据报的正式规范，因此需要进行简短的讨论。TL是一种适用于描述从属代数类型的语言，允许使用数字（自然数）和类型参数。 通过几个构造函数描述每种类型。 每个构造函数都有一个（人可读的）标识符和一个名称，它是一个位串（默认情况下为32位整数）。 除此之外，构造函数的定义包含字段列表及其类型。构造函数和类型定义的集合称为TL方案。 它通常保存在一个或多个带有后缀 .tl 的文件中。

TL方案的一个重要特征是它们确定了一种明确的方式来序列化和反序列化定义的代数类型的值（或对象）。 也就是说，当需要将值序列化为字节流时，首先序列化用于此值的构造函数的名称。 然后递归计算每个字段的序列化。

适用于将任意对象序列化为32位整数序列的TL的先前版本的描述可在 https://core.telegram.org/mtproto/TL 获得。 正在开发一种名为TL-B的TL新版本，用于描述电报开放网络(TON)项目使用的对象的序列化。 这个新版本可以将对象序列化为字节流甚至比特流（不仅仅是32位整数），并且支持将序列化到TVM细胞树中（参见 **2.3.14** ）。 TL-B的描述将是电报开放网络(TON)区块链的正式规范的一部分。

2.2.6. 块和交易作为状态转换运算符。 通常，任何区块链（类型）区块链都具有关联的全局状态（类型）状态和交易（类型）交易。 区块链的语义在很大程度上由交易应用程序函数决定：

$$ev\_trans' : \text{交易} \times \text{状态} \to \text{状态}? \tag{3}$$

[5] https://coq.inria.fr

[6] https://core.telegram.org/mtproto/TL

这里$X^?$表示 MAYBE X，即将 MAYBE 单子应用于 X 的结果。这类似于我们对 LIST X 使用 X^*. 基本上，$X^?$ 的值是 X 类型的值或特殊值 $\perp$，表示缺少实际值（考虑空指针）。 在我们的例子中，我们使用$State^?$而不是状态作为结果类型，因为如果从某些始发状态调用交易可能是无效的（考虑尝试从帐户中提取比实际存在更多的钱）。

我们可能更喜欢一个咖喱版的ev_trans'：

$$ev\_trans : 交易 \to 状态 \to 状态^? \tag{4}$$

因为块本质上是一个交易列表，所以块评估函数

$$ev\_trans : 交易 \to 状态 \to 状态^? \tag{5}$$

可以从 ev_trans 派生。 它需要一个块 B：块和前一个区块链状态 s：状态（可能包括前一个块的哈希值）并计算下一个区块链状态 $s' = ev\_block\ (B)(s)$：$State$，它是一个真状态或者表示不能计算下一个状态的特殊值 $\perp$（即，如果从给定的起始状态进行评估，则该块是无效的——例如，该块包括试图从空账户借记的交易。）

2.2.7. 块序列号。 区块链中的每个区块 B 可以通过其序列号 BLK-SEQNO(B) 来引用，从第一个区块的零开始，并且每当传递到下一个区块时增加一。 更正式的方程：

$$\text{BLK-SEQNO}(B) = \text{BLK-SEQNO}(\text{BLK-PREV}(B)) + 1 \tag{6}$$

请注意，序列号在存在分叉时不会唯一地标识块。

2.2.8. 块哈希。 引用块 B 的另一种方式是通过其哈希 BLK-HASH(B)，其实际上是块 B 的标头的HASH(然而，块的标头通常包含依赖于块 B 的所有内容的哈希）。假设所使用的哈希函数没有冲突（或者至少它们是非常不可能的），这块通过其唯一的哈希被标识。

2.2.9. 哈希假设。 在区块链算法的形式分析期间，我们假设使用的 k 位哈希函数哈希：字节* $\to 2^k$ 没有冲突：

$$\text{HASH}(s) = \text{HASH}(s') \Rightarrow s = s' \text{ 对于任何 } s, s' \in \text{字节}^* \tag{7}$$

这里$Bytes = \{0 \ldots 255\} = \mathbf{2}^8$ 是字节的类型，或者是所有字节的集合值，$Bytes$*是任意（有

限）字节列表的类型或集合; 而 **2** ={0,1} 是位类型，$\boldsymbol{2}^k$是所有 k 位的集合（或实际上是类型）序列（即k位数）。

当然，(7)在数学上是不可能的，因为从无限集到有限集的映射不能是单射的。更严格的假设是

$$\forall s, s' : s \neq s', \mathrm{P}(\mathrm{HASH}(s) = \mathrm{HASH}(s')) = 2^{-k} \tag{8}$$

但是，这对于校样来说并不方便。 如果对于某些小 ϵ 例如，$\epsilon = 10^{-18}$），如果(8)在$2^{-k}N < \epsilon$的证明中最多使用 N 次，我们可以推理(7)，如果我们接受失败概率ϵ（即最终结论将是 是真实的，概率至少为1 - ϵ）。

最后评论：为了使(8)的概率陈述真正严格，必须在所有字节序列的集合字节*上引入概率分布。 这样做的一种方式是假设所有相同长度 l 的字节序列是同等概率的，并且设置观察长度 l 的序列等于 $p^l - p^{l-1}$ 的概率用于某些 p → 1- 。 然后(8)应该被理解为条件概率 P(HASH(s) = HASH(s') | $s \neq s'$)的极限，当 p 从下面倾向于一时。

2.2.10. 哈希用于电报开放网络(TON)区块链。 我们暂时把256位SHA256哈希用于电报开放网络(TON)区块链。 如果事实证明它比预期的更弱，那么将来可以用另一个哈希函数替换它。 哈希函数的选择是协议的可配置参数，因此可以在没有硬分叉的情况下进行更改，如 **2.1.2** 中所述。

2.3 区块链状态，账户和Hashmap。

我们已经注意到，任何区块链都定义了某个全局状态，每个块和每个交易都定义了这个全局状态的转换。 这里我们描述电报开放网络(TON)区块链使用的全局状态。

2.3.1. 帐户ID。 电报开放网络(TON)区块链使用的基本帐户ID—或至少由其主链和零工作链——使用的基本帐户ID是256位整数，假定为特定椭圆曲线的256位椭圆曲线密码(ECC)的公钥。 通过这种方式，

$$\text{帐户\_}id : \text{帐户} = uint^{256} = \boldsymbol{2}^{256} \tag{9}$$

此处*Account*是帐户类型，而*Account_id*：*Account*是*Account*类型的特定变量。

其他工作链可以使用其他帐户ID格式，256位或其他。 例如，可以使用比特币风格的帐户ID，等于ECC公钥的SHA256 。

但是，在创建工作链（在主链中）期间必须固定帐户ID的位长度l，并且它必须至少为l六十四，因为账户_*id*的前六十四位用于分片和信息路由。

2.3.2. 主要组成部分：*Hashmap*。 电报开放网络(TON)区块链状态的主要组成部分是一个Hashmap。在某些情况下，我们考虑（部分定义）《映射》$h\colon \mathbf{2}^n \dashrightarrow \mathbf{2}^m$。更一般地，我们可能对复合类型 X 的*Hashmap* $h\colon \mathbf{2}^n \dashrightarrow X$ 感兴趣。但是，来源（或索引）类型几乎总是 $\mathbf{2}^n$ 。

有时，我们将"默认值"设为空：X，并且Hashmap $h\colon 2^n \to X$ 由其"默认值" $i \mapsto$ 空"初始化"。

2.3.3. 示例：电报开放网络(TON)帐户余额。电报开放网络(TON)账户余额给出了一个重要的例子。这是一个*Hashmap*

$$\text{余额}：\text{账户} \to uint_{128} \tag{10}$$

将$Account = \mathbf{2}^{256}$ 映射到 $uint_{128} = \mathbf{2}^{128}$ 类型的Gram（电报开放网络(TON)代币）余额。此Hashmap的默认值为零，这意味着最初（在处理第一个块之前）所有帐户的余额为零。

2.3.4. 示例：智能合约持久存储。 另一个例子是智能合约持久存储，它可以（非常近似地）表示为*Hashmap*

$$\text{存储}：\mathbf{2}^{256} \dashrightarrow \mathbf{2}^{256} \tag{11}$$

该哈希图也具有默认值零，这意味着假定持久存储的未初始化的细胞为零。

2.3.5. 示例：持续存储所有智能合约。 因为我们有多个智能合约，由账户_id区分，每个都有独立的持久存储，我们实际上必须有一个*Hashmap*

$$\text{存储}：\text{账户} \dashrightarrow (\mathbf{2}^{256} \dashrightarrow \mathbf{2}^{256}) \tag{12}$$

将智能合约的账户_*id*映射到其持久存储中。

2.3.6. *Hashmap*类型。 *Hashmap*不仅仅是一个抽象（部分定义的）函数 $\mathbf{2}^n \dashrightarrow X$; 它有一个特定的表示。因此，我们假设我们有一个特殊的*Hashmap*类型

$$Hashmap(n, X) : 类型 \tag{13}$$

对应于编码（部分）映射 $2^n \dashrightarrow X$ 的数据结构。我们也可以写

$$Hashmap(n, nat)(X, 类型) : 类型 \tag{14}$$

或者

$$Hashmap : nat \to 类型 \to 类型 \tag{15}$$

我们总是可以将 $h : Hashmap(n, X)$ 转换成映射 $hget(h) : 2^n \to X^?$。 从此以后，我们通常写 $h[i]$ 而不是 $hget(h)(i)$。

$$h[i] : \equiv hget(h)(i) : X^? \text{ 对于任何 } i : \mathbf{2}^n \text{ , } h : Hashmap(n, X) \tag{16}$$

2.3.7. 将Hashmap类型定义为帕特里夏树。 从逻辑上讲，可以将*Hashmap(n, X)*定义为深度为 n 的（不完整）二叉树，边缘标签为0和1，叶片中的值为X。 描述相同结构的另一种方式是对于长度等于 n 的二进制串的 (bitwise)trie 。

在实践中，我们更喜欢使用此 trie 的紧凑表示，通过压缩每个与其父节点只有一个子节点的顶点。 结果表示称为帕特里夏树或二进制基数树。 现在每个中间顶点都有两个子节点，用两个非空二进制字符串标记，左边的子节点为零，右边的子节点为一。

换句话说，帕特里夏树中有两种类型的（非根）节点：

- 叶片(x)，包含 X 类型的值 x。
- 节点(l, sl , r, sr)，其中 l 是（引用）左子节点或子树， s_l 是标记连接此顶点到其左子节点的边的位串（始终以零开头）， r 是右边的子树， s_r 是标记往右子节点的边的位串（始终以一开头）。

第三种类型的节点，仅在帕特里夏树的根处使用一次，也是必要的：

- 根(n, s_0, t)，其中 n 是*Hashmap*(n, X)的索引位串的公共长度，s_0 是所有索引位串的公共前缀，t 是对叶或节点的引用。

如果我们想让帕特里夏树为空，将使用第四种类型的（根）节点：

- 空根(*n*), 其中 *n* 是所有索引位串的公共长度。我们如此定义帕特里夏树的高度:

$$\text{高度}(\text{叶}(x)) = 0 \tag{17}$$

$$\text{高度}(\text{节点}(l, s_l, r, s_r)) = \text{高度}(l) + \text{长度}(s_l) = \text{高度}(r) + \text{长度}(s_r) \tag{18}$$

$$\text{高度}(\text{根}(n, s_0, t)) = \text{长度}(s_0) + \text{高度}(t) = n \tag{19}$$

最后两个公式中的最后两个表达式必须相等。 我们使用高度为 n 的帕特里夏树来表示 *Hashmap*(*n, X*)类型的值。

如果树中有 N 个叶子（即，我们的*Hashmap*包含 N 个值），则恰好有 N - 1个中间顶点。 插入新值总是涉及通过在中间插入新顶点并添加新叶作为此新顶点的另一个子来分割现有边。 从*Hashmap*中删除值则相反：删除一个叶子及其父级，然后父级的父级和另一个子级直接链接。

2.3.8. 默克尔-帕特里夏树。 使用区块链时，我们希望能够比较帕特里夏树（即*Hashmap*）及其子树，方法是将它们减少为单个哈希值。默克尔树给出了实现这一目标的经典方法。 本质上，我们想要描述一种哈希*Hashmap(n, X)*类型的对象 h 的方法，借助为二进制字符串定义的哈希函数哈希，前提是我们知道如何计算对象 $x : X$ 的哈希值HASH(x)（例如，通过将哈希函数哈希应用于对象 x 的二进制序列化）。

可以递归地定义HASH(h)，如下：

$$\text{哈希}(\text{叶}(x)) := \text{哈希}(x) \tag{20}$$

$$\text{哈希}(\text{节点}(l, s_l, r, s_r)) := \text{哈希}(\text{哈希}(l).\text{哈希}(r).\text{码}(s_l).\text{码}(s_r)) \tag{21}$$

$$\text{哈希}(\text{根}(n, s_0, t)) := \text{哈希}(\text{码}(n).\text{码}(s_0).\text{哈希}(t)) \tag{22}$$

这里 $s.t$ 表示（位）字符串 s 和 t 的串联，而码 CODE(s) 是所有位串 s 的前缀代码。 例如，可以将0编码为10, 1编码为11，并将字符串的结尾编码为0。[7]

稍后我们将看到（参见 **2.3.12** 和 **2.3.14**）这是一个（略微调整的）递归定义的哈

[7] 可以证明，这种编码对于具有随机或连续索引的帕特里夏树的所有边缘标签的大约一半是最佳的。 剩余的边缘标签可能很长（即，几乎256位长）。 因此，边缘标签的差不多最佳编码是使用上述代码，前缀为0表示《短》位串，并编码1，然后是九位，其中包含长度为 $l = |s|$ 的位串 s，然后是 l 位 s 为表示《长》位串（$l \geq 10$）。

2.3.9. 重新计算默克尔树的哈希值。 这种递归定义HASH(h)的方式，称为默克尔树哈希，具有以下优点：如果有人明确地将HASH(h')与每个节点 h' 一起存储（导致称为默克尔树的结构，或者在我们的情况下， 一个默克尔-帕特里夏树），当在*Hashmap*中添加，删除或更改元素时，只需要重新计算最多 n 个哈希值。

这样，如果通过合适的默克尔树哈希表示全局区块链状态，则在每次交易之后很容易再计算该状态哈希。

2.3.10. 默克尔树证明。 在所选哈希函数哈希的《注入性》假设(7)下，我们可以构造一个证明，对于HASH(h), h : *Hashmap*(n, X)的给定值 z，我们有一些 $i : 2^n$ 和 $x : X$ 的 $hget(h)(i) = x$。 这样的证明将包括默克尔-帕特里夏树中从对应于 i 的叶到根的路径，由在该路径上发生的所有节点的所有兄弟的哈希增强。

以这种方式，仅知道某些*Hashmap* h 的*HASH*(h)的值的轻节点[8]（例如，智能合约持久存储或全局区块链状态）可以从完整节点[9] 不仅请求值 $x = h[i] = hget(h)(i)$，而且还请求这样的值再加从已知值HASH(h)开始的默克尔树证明。 然后，在假设(7)下，轻节点可以自己检查 x 确实是 $h[i]$ 的正确值。

在某些情况下，客户端可能想要获得值 y = HASH(x) = HASH($h[i]$)——例如，如果 x 本身非常大（例如，*Hashmap*本身）。 那么就可以提供(i, y)的默克尔树证明。 如果 x 也是*Hashmap*，则可以从完整节点获得从 y = HASH(y) 开始的第二个默克尔树证明，以提供值 $x[j] = h[i][j]$ 或仅其哈希。

2.3.11. 对于诸如电报开放网络(TON)的多链系统，默克尔树证明的重要性。 请注意，节点通常不能是电报开放网络(TON)环境中存在的所有分片链的完整节点。 它通常只是一些分片链的完整节点——例如，那些包含自己的帐户，它感兴趣的智能合约，或者已经分配了该节点作为验证者的那些分片链。 对于其他分片链，它必须是一个轻节点——否则存储，计算和网络带宽要求将是令人望而却步的。 这意味着这样的节点不能直接检查关于其他分片链状态的断言; 它必须依赖于从那些分片链的完整节点获得的默克尔树证明，这与自身检查一样安全，除非(7)失败（即，发现了哈希冲突）。

2.3.12. 电报开放网络(TON)VM的特点。 用于在主链和零工作链中运行智能合约的TON VM或TVM（电报虚拟机）与受EVM（以太坊虚拟机）启发的习惯设计有很大不同：它不仅适用于256位整数，而且实际上有（几乎）任意《记录》，《结构》或《和积类型》，使其更适合执行用高级（特别是功能）语言编写的代码。从本质上讲，TVM使用标记数据

[8] 轻节点是不跟踪分片链的完整状态的节点; 相反，它保留最少的信息，例如几个最近的块的哈希值，并且当需要检查完整状态的某些部分时，依赖于从完整节点获得的信息。

[9] 完整节点是一个用于跟踪所讨论的分片链的完整最新状态的节点

类型，与Prolog或Erlang实现中使用的数据类型相似。

人们可能首先想到，TVM智能合约的状态不仅仅是一个*Hashmap* $\mathbf{2}^{256} \to \mathbf{2}^{256}$，或*Hashmap*(256, $\mathbf{2}^{256}$)，而是（作为第一步）*Hashmap*(256, *X*)，其中X是具有多个构造函数的类型，使其能够存储除256位整数之外的其他数据结构，包括其他*Hashmap*，特别是*Hashmap*(256, *X*)。 这将意味着TVM（持久或临时）存储的细胞——或TVM智能合约代码中的变量或阵列元素——不仅可以包含整数，而且可以包含全新的*Hashmap*。 当然，这意味着一个细胞不仅包含256位，而且还包含八位的标签，为了描述如何理解这256位。实际上，值不需要精确为256位。 TVM使用的值格式由一系列始发字节和对其他结构的引用组成，以任意顺序混合，一些描述符字节插入合适的位置，以便能够区分指针和原始数据（例如，字符串或整数）;参见 **2.3.14** 。

该原始值格式可用于实现任意和积代数类型。 在这种情况下，该值首先包含一个原始字节，描述正在使用的《构造函数》（从高级语言的角度来看），然后是其他《字段》或《构造函数参数》，由原始字节和对其他结构的引用组成，具体取决于所选的构造函数（参见 **2.2.5** ）。 但是，TVM对构造函数与其参数之间的对应关系一无所知; 字节和引用的混合由某些描述符字节显式描述。 [10]

默克尔树哈希扩展到任意这样的结构：为了计算这种结构的哈希，所有引用都被递归的对象哈希替换，然后计算得到的字节串（包括描述符字节）的哈希。

通过这种方式， **2.3.8** 中描述的用于哈希图的默克尔树哈希，只是对任意（从属）代数数据类型进行哈希的特殊情况，应用于具有两个构造函数的类型*Hashmap*(*n*, *X*)。 [11]

2.3.13. 持久存储TON智能合约。 持久存储电报开放网络(TON)智能合约基本上由其《全局变量》组成，在智能合约的调用之间保留。 因此，它只是一个《产品》，《元组》或《记录》类型，由正确类型的字段组成，每个字段对应一个全局变量。

如果全局变量太多，则由于对电报开放网络(TON)细胞大小的全局限制，它们不能适合一个TON细胞。 在这种情况下，它们被分成若干记录并组织成树，基本上成为《产品的产品》或《产品的产品的产品》类型而不仅仅是产品类型。

2.3.14. TVM细胞。 最终，电报开放网络(TON) VM将所有数据保存在(TVM)细胞的集合中。 每个细胞首先包含两个描述符字节，指示该细胞中存在多少字节的原始数据（最多128个）以及存在多少个其他细胞的参考（最多四个）。 然后是这些原始数据字节

[10] 这两个描述符字节，存在于任何TVM细胞中，仅描述参考总数和原始字节总数; 引用在所有原始字节之前或之后保存在一起。

[11] 实际上，叶和节点是辅助类型的构造函数，*Hashmap*辅助(*n*, *X*)。 类型*Hashmap*(*n*, *X*)具有构造函数根和空根，其中根包含类型为*Hashmap*辅助(*n*, *X*)的值。

和引用。 每个细胞只被引用一次，因此我们可能在每个细胞中包含对其《父级》的引用（引用该父级细胞的唯一细胞）。 但是，此引用不必明确。

以这种方式, 电报开放网络(TON)智能合约的持久数据存储细胞被组织成树 [12] ，引用智能合约描述中保存的此树的根。 如有必要，从叶子开始递归计算整个持久存储的默克尔树哈希值，然后简单地用生成的细胞的递归计算哈希值替换细胞中的所有引用，并随后计算由此获得的字节串的哈希值。

2.3.15. 任意代数类型值的广义默克尔树证明。 因为电报开放网络(TON) VM通过由(TVM)细胞组成的树表示任意代数类型的值，并且每个细胞具有明确定义的（递归计算的）默克尔哈希，实际上取决于根据该细胞生成的整个子树， 我们可以为任意代数类型的（部分）值提供《广义默克尔树证明》，旨在证明具有已知默克尔哈希的树的某个子树采用特定值或具有特定哈希的值。 这概括了 **2.3.10** 的方法，其中只考虑了$x[i] = y$ 的默克尔树证明。

2.3.16. 支持TON VM数据结构中的分片。 我们刚刚概述了电报开放网络(TON) VM如何在不过度复杂的情况下支持高级智能合约语言中的任意（相关）代数数据类型。 但是，大型（或全局）智能合约的分片需要在TON VM级别上提供特殊支持。 为此，系统中添加了特殊版本的*Hashmap*类型，相当于《映射》账户 $\dashrightarrow X$。 这个《映射》似乎等同于$Hashmap(m, X)$，其中账户 $= 2^m$ 。 但是，当分片分为两个或两个分片合并时，这些*Hashmap*会自动分成两部分或合并回来，以便只保留属于相应分片的那些钥匙。

2.3.17. 支付持久存储。 电报开放网络(TON) 区块链的一个值得注意的特征是从智能合约中提取的用于存储其持久数据的付款（即，用于扩大区块链的总状态）。 它的工作原理如下：

每个块声明两个速率，以区块链的主要货币（通常是Gram）提名：将一个细胞保留在持久存储中的价格，以及在持久存储的某个细胞中保留一个原始字节的价格。 每个帐户使用的细胞和字节总数的统计信息存储为其状态的一部分，因此通过将这些数字乘以块标头中声明的两个费率，我们可以计算要从帐户余额中扣除的付款以便保留它在前一个块和当前块之间的数据。

但是，对于每个帐户和每个块中的智能合约，不会为持久存储使用付款; 相反，该支付最后被执行的块的序列号存储在帐户数据中，并且当对该帐户进行任何措施时（例如，通过智能合约接收和处理价值转移或信息）， 在任何措施被采取之前从帐户余额中扣除自从上一次此类付款以来所有块的存储使用付款。 如果此后帐户的余额将变为

[12] 逻辑上; **2.5.5** 中描述的"细胞袋"表示识别所有重复细胞，在序列化时将该树转化为有向无环图 (dag)。

负数，则该帐户将被销毁。

工作链可以声明每个帐户的一些始发数据字节是《免费》的（即，不参与持久性存储支付），以便制作《简单》帐户，这些帐户的余额仅以一种或两种加密货币保存，免除这样的持续付款。

请注意，如果没有人向帐户发送任何信息，则不会收集其持久存储付款，并且它可以无限期地存在。 但是，任何人都可以发送一条空信息来销毁这样的帐户。 从要销毁的帐户的始发余额的一部分收集的小激励可以被提供给这样的信息的发送者。 但是，我们期望验证者可以免费销毁此类无力偿债的账户，只是为了减少全局区块链的规模，避免在没有补偿的情况下保留大量数据。

为保持持久数据而收集的付款分布在分片链或主链的验证者中（与后者的权益成比例）。

2.3.18. 本地和全局智能合约; 智能合约实例。 智能合约通常只存在于一个分片中，根据智能合约的账户_*id*选择，类似于《普通》账户。 对于大多数应用程序来说，这通常就足够。但是，一些《高负荷》的智能合约可能希望在某些工作链的每个分片链中都有一个《实例》。 为了实现这一点，他们必须将他们的创建交易传播到所有分片链中，例如，通过将此交易提交到工作链 w 的《根》分片链 $(w, \emptyset)$ [13] 并支付大额佣金。 [14]

此操作有效地在每个分片中创建智能合约的实例，并具有单独的余额。 最初，在创建交易中传输的余额仅通过在分片(w, s)中为实例提供总余额的$2^{-|s|}$ 部分来分配。 当分片分成两个子分片时，所有全局智能合约实例的余额分成两半; 当两个分片合并时，余额会加在一起。

在某些情况下，分割或合并全局智能合约的实例可能涉及（延迟）执行这些智能合约的特殊方法。 默认情况下，如上所述拆分和合并余额，并且一些特殊的《帐户索引》Hashmap也会自动拆分和合并（参见 **2.3.16** ）。

2.3.19. 限制智能合约的分割。 全局智能合约可能会在创建时限制其拆分割深度 d，以便使持久存储费用更具可预测性。 这意味着，如果分片链 (w, s) 与 $|s| \geq d$ 分成两部分，两个新的分片链中只有一个继承了智能合约的一个实例。 确定性地选择此分片链：每个全局智能合约都有一些《账户_*id*》，它本质上是其创建交易的哈希值，并且其实例具有相同的账户_*id*，其中第一个 $\leq d$ 位替换为落入正确分片所需的适当值。 此账户_*id*选择在拆分后将哪个分片继承智能合约实例。

2.3.20. 账户或智能合约状态。 我们可以总结以上所有内容，得出结论：帐户或智能合约状态包含以下内容：

[13] 更昂贵的替代方案是在主链中发布这样一个《全局》智能合约。

[14] 这是所有分片的一种《广播》功能，因此它必定很昂贵。

- 区块链的主要货币余额
- 区块链其他货币的余额
- 智能合约代码（或其哈希）
- 智能合约持久性数据（或其默克尔哈希）
- 统计持久存储细胞的数量和使用的原始字节
- 收集智能合约持久存储的付款时的最后一次（实际上是主链块号）
- 传输货币并从此帐户发送信息所需的公钥（可选;默认情况下等于账户 *_id*本身）。 在某 些情况下，可以在此处找到更复杂的签名检查代码，类似于比特币交易输出所做的;然后账户 *_id* 将等于此代码的哈希值。

我们还需要在帐户状态或其他一些帐户索引的Hashmap中保留以下数据：

- 帐户的输出信息队列（参见 **2.4.17** ）
- 最近发送的信息（哈希）的集合（参见 **2.4.23** ）

并非所有这些都是每个帐户真正需要的; 例如，智能合约代码仅适用于智能合约，但不适用于《简单》帐户。 此外，虽然任何账户必须具有主要货币的非零余额（例如，基本工作链的主链和分片链的Gram），但其他货币的余额可能为零。 为了避免保留未使用的数据，（在工作链创建期间）定义了和-积类型（取决于工作链），它使用不同的标记字节（例如，TL 构造函数;参见 **2.2.5** ）来区分不同的被使用的《建设者》。 最终，帐户状态本身被保存为TVM持久存储的细胞的集合。

2.4 分片链之间的信息

电报开放网络(TON) 区块链的一个重要组成部分是区块链之间的信息传递系统。 这些区块链可以是相同工作链的分片链，也可以是不同工作链的分片链。

2.4.1. 信息，帐户和交易：鸟瞰系统。 信息从一个帐戶发送到另一个帐戶。 每次交易包括一个帐戶接收一条信息，根据某些规则更改其状态，以及生成几个（可能是一个或零个）新信息到其他帐戶。 每条信息都会生成并接收（传递）一次。这意味着信息在系统中起着基本作用，与账戶（智能合约）相当。 从无限分片范例（参见 **2.1.2** ）的角度来看，每个帐戶都位于其独立的《帐戶链》中，并且影响其他帐戶状态的唯一方法是发送信息。

2.4.2. 作为进程或参与者的帐户; 参与者模式。 人们可能会将帐戶（和智能合约）视为《进程》或《参与者》，它们能够处理传入的信息，更改其内部状态并生成一些出站信息。 这与所谓的参与者模式密切相关，在Erlang等语言中使用（但是，Erlang中的参与者通常称为《进程》）。 由于处理入站信息的结果也允许现有参与者创建新参与者（即智能合约），因此与参与者模式的对应基本完成。

2.4.3. 信息接收者。 任何信息都有其接收者，其特征是目标工作链标识符 *w*（默认情况下假定它与发起的分片链相同）和接收者帐戶 账戶*_id*。 账戶*_id* 的确切格式（位数）取决于 *w*; 但是，分片始终由其前（最重要的）64位确定。

2.4.4. 信息发送者。 在大多数情况下，信息具有发送者，再次由 (*w′*, 账戶*_id′*) 这一对（值）来代表特征。 如果存在，则它位于信息接收者和信息值之后。 有时，发送者不重要或者是区块链之外的（即，不是智能合约），在这种情况下，该字段不存在。

请注意，参与者模型不要求信息具有隐式发送方。 相反，信息可以包含对应该发送请求的答案的参与者的引用; 通常它与发送者一致。 但是，在加密货币（拜占庭）环境中的信息中具有显式的不可伪造的发送者字段是有用的。

2.4.5. 信息值。 信息的另一个重要特征是其附加值，由源和目标工作链支持的一个或多个加密货币。 信息的值在信息接收者之后立即显示; 它本质上是（货币*_id* , *value*）这一对的列表。

请注意，《简单》帐戶之间的《简单》值转移只是空（无操作）信息，并附加了一些值。 另一方面，稍微复杂的信息体可能包含简单的文本或二进制注释（例如，关于支付的目的）。

2.4.6. 外部信息，或《来自无处的信息》。 有些信息《从无处》进入系统 - 也就是说，它们不是由位于区块链中的账戶（智能合约或非智能合约）生成的。 当用戶想要将一些资金从她控制的账戶转移到其他账戶时，就会出现最典型的例子。 在这种情况下，用戶将

《来自无处的信息》发送到她自己的帐户，请求它生成带有指定值的接收帐户的信息。如果此信息已正确签名，则她的帐户会收到该信息并生成所需的出站信息。

事实上，人们可能会将《简单》帐户视为具有预定义代码的智能合约的特例。这个智能合约只收到一种信息。这样的入站信息必须包含由于传送（处理）入站信息而生成的出站信息列表以及签名。智能合约检查签名，如果正确，则生成所需的信息。

当然，《来自无处的信息》与正常信息之间存在差异，因为《来自无处的信息》无法承受价值，因此他们无法自己支付《汽油》（即他们的处理）。相反，他们暂时执行了一个小的汽油限制，甚至建议包括在一个新的分片链块; 如果执行失败（签名不正确），则《来自无处的信息》被认为是不正确的并被丢弃。如果执行没有在小汽油限制内失败，则该信息可以包括在新的分片链块中并完全处理，从接收方的账户中消耗的汽油（处理能力）的支付。《来自无处的信息》还可以定义一些交易费用，该费用从接收者的账户中扣除，用于再分配给验证者的汽油支付。

在这个意义上，《来自无处的信息》或《外部信息》承担在其他区块链系统（例如，比特币和以太坊）中使用的交易候选者的角色。

2.4.7. 记录信息，或《发往无处的信息》。 类似地，有时可以生成特殊信息并将其路由到特定的分片链，而不是将其传递给其接收者，而是记录以便任何接收有关该分片的更新的接收者都能够容易地观察到。这些记录的信息可以在用户的控制台中输出，或者在离线服务器上触发某些脚本的执行。从这个意义上说，它们代表了《区块链超级计算机》的外部《输出》，就像《来自无处的信息》代表《区块链超级计算机》的外部《输入》一样。

2.4.8. 与脱链服务和外部区块链的互动。 这些外部输入和输出信息可用于与脱链服务和其他（外部）区块链（如比特币或以太坊）进行交互。有人可能会在电报开放网络(TON)区块链内部创建代币或加密货币，这些区块链与比特币，以太币或以太坊区块链中定义的任何ERC-20代币挂钩，并使用《来自无处的信息》和《发往无处的信息》，由驻留在某些第三方脱链服务器上的脚本生成和处理，用于实现电报开放网络(TON)区块链与这些外部区块链之间的必要交互。

2.4.9. 信息正文。 信息体只是一个字节序列，其含义仅由接收工作链和（或）智能合约确定。对于使用电报开放网络(TON) VM的区块链，这可以是通过 Send( )操作自动生成的任何TVM细胞的序列化。简单地通过用所引用的细胞递归地替换TON VM细胞中的所有引用来获得这种序列化。最终，出现一串原始字节，通常由四字节《信息类型》或《信息构造函数》作为前缀，用于选择接收智能合约的正确方法。

另一种选择是使用TL序列化对象（参见 **2.2.5** ）作为信息体。 这对于不同工作链之间的通信尤其有用，其中一个或两个工作链不一定使用TON VM。

2.4.10. 汽油限制和其他工作链/ VM特定参数。 有时，信息需要携带有关汽油限制，汽油价格，交易费用和依赖于接收工作链的类似值的信息，并且仅与接收工作链相关，但不一定与始发工作链相关。 这些参数包含在信息正文中或之前，有时（取决于工作链）具有特殊的4字节前缀，表示它们的存在（可以通过TL方案定义;参见 **2.2.5** ）。

2.4.11. 创建信息：智能合约和交易。 有两个新信息来源。 大多数信息是在智能合约执行期间（通过TON VM中的Send( )操作）创建的，此时调用某个智能合约来处理传入信息。 或者，信息可以来自外部，作为《外部信息》或《来自无处的信息》（参见 **2.4.6** ）。[15]

2.4.12. 传递信息。 当信息到达包含其目标帐户的分片链时，[16] 它将《传递》到其目标帐户。 接下来会发生什么取决于工作链; 从外部的角度来看，重要的是这样的信息永远不能从这个分片链进一步转发。

对于基本工作链的分支链，交付包括将信息值（减去任何汽油支付）添加到接收帐户的余额，并且可能在之后调用接收智能合约的信息相关方法，如果接收帐户是智能合约。 实际上，智能合约只有一个用于处理所有传入信息的入口点，它必须通过查看它们的前几个字节来区分不同类型的信息（例如，包含TL构造函数的前四个字节;参见 **2.2.5** ）。

2.4.13. 传递信息是一种交易。 因为信息的传递改变了帐户或智能合约的状态，所以它是接收分片链中的特殊交易，并且明确地如此注册。 从本质上讲，所有电报开放网络(TON)区块链交易都包括向其接收账户（智能合约）交付一个入站信息，忽略一些小的技术细节。

2.4.14. 同一智能合约的实例之间的信息。 回想一下智能合约可能是本地的（即，像任何普通账户一样驻留在一个分片链中）或全局的（即，在所有分片中具有实例，或者至少在所有分片中具有某个已知深度 *d*;参见 **2.3.18** ）。 如果需要，全局智能合约的实例可以交换特殊信息以在彼此之间传递信息和价值。 在这种情况下，（不可伪造的）发送者 账户_*id* 变得很重要（参见 **2.4.4** ）。

[15] 以上只需要对基本的工作链及其分片链算是对的; 其他工作链可能提供其他创建信息的方法。

[16] 作为一个退化的案例，这个分片链可能与始发的分片链重合——例如，如果我们在一个尚未拆分的工作链内工作。

2.4.15. 发送给智能合约任何实例的信息；通配符地址。 有时，信息（例如，客户端请求）需要被传递到全局智能合约的任何实例，通常是最接近的一个（如果存在与发送者相同的分片链中的一个实例，则它是明显的候选者）。 一种方法是使用《通配符收件人地址》，允许目标 账户_*id* 的前 *id* 位采用任意值。 在实践中，通常会将这些 *d* 位设置为与发送方的 账户_*id* 中相同的值。

2.4.16. 输入队列不存在。 区块链（通常是分片链;有时是主链）接收的所有信息——或者基本上由驻留在某个分片链内的"帐户链"——立即传递（即，由接收帐户处理）。 因此，没有《输入队列》。 相反，如果不是因为块的总大小和汽油使用的限制而可以处理发往特定分片链的所有信息，则一些信息只是留在始发分片链的输出队列中累积。

2.4.17. 输出队列。 从无限分片范例（参见 **2.1.2** ）的角度来看，每个帐户链（即每个帐户）都有自己的输出队列，包括它已生成但尚未传递给其接收者的所有信息。 当然，账户链只有虚拟存在; 它们被分组为分片链，而分片链有一个输出《队列》，它由属于分片链的所有帐户的输出队列的并集组成。

此分片链输出《队列》仅对其成员信息强加部分顺序。 也就是说，必须在后续块中生成的任何信息之前传递前一个块中生成的信息，并且必须按照它们生成的顺序传递由同一帐户生成并具有相同目的地的任何信息。

2.4.18. 可靠，快速的链间信息传递。 对于像电报开放网络(TON)这样的可扩展多区块链项目来说，能够在不同的分片链之间转发和传递信息（参见 **2.1.3** ）是至关重要的，即使系统中有数百万个分片链也是如此。 信息应该可靠地并且很快地传送（即，信息不应丢失或传送多于一次）。 电报开放网络(TON)区块链通过结合使用两个《信息路由》机制实现了这一目标。

2.4.19. 超立方体路由：确保传递的信息的《慢速路径》。 电报开放网络(TON)区块链使用《超立方体路由》作为一种缓慢但安全可靠的方式，将信息从一个分片链传递到另一个分片链，如有必要，可使用多个中间分片链进行传输。 否则，任何给定分片链的验证者将需要跟踪所有其他分片链的状态（输出队列），这将需要大量的计算能力和网络带宽，因为分片链的总数量增加，从而限制了系统的可伸缩性。 因此，无法直接从任何分片向其他分片传递信息。 相反，每个分片仅《连接》到不同于其(*w*, *s*)分片标识符的一个十六进制数字的分片（参见 **2.1.8** ）。 这样，所有的分片链都构成了一个《超立方体》图形，并且信息沿着这个超立方体的边缘传播。

如果将信息发送到与当前分片不同的分片，则当前分片标识符的十六进制数字之

一（确定性地选择）将被目标分片的相应数字替换，并将生成的标识符用作转发信息的邻近目标。[17]

超立方体路由的主要优点是块有效性条件意味着创建分片链块的验证者必须收集并处理来自《相邻》分片链的输出队列的信息，以免丢失其权益。通过这种方式，可以预期任何信息迟早会到达其最终目的地;信息不能在传输过程中丢失或交付两次。

请注意，超立方体路由会引入一些额外的延迟和费用，因为必须通过几个中间分片链转发信息。然而，这些中间分片链的数量增长非常缓慢，是分片链 N 的总数的对数 $\log N$ (更确切地说，$\lfloor \log_{16} N \rfloor - 1$)。例如，如果 $N \approx 250$，最多只有一个中间跃点;对于 $N \approx 4000$ 个分片链，最多两个。有四个中间跃点，我们可以支持多达一百万个分片链。我们认为这对于系统的基本无限可扩展性来说是一个非常小的代价。事实上，没有必要支付这个代价：

2.4.20. 即时超立方体路由：信息的《快速路径》。 TON区块链的一个新特点是它引入了一条《快速路径》，用于将信息从一个分片链转发到任何其他分片链，允许在大多数情况下完全绕过 **2.4.19** 的《慢速》超立方体路由并将信息传递到最后一个目的地分片链的下一个区块。

这个想法如下。在《慢速》超立方体路由期间，信息沿着超立方体的边缘行进（在网络中），但是在每个中间顶点处被延迟（大约五秒）以在继续其航行之前被提交到相应的分片链中。

为了避免不必要的延迟，可以改为将信息与沿着超立方体边缘的合适的默克尔树证明一起中继，而不必等待将其提交到中间的分片链中。实际上，网络信息应该从始发分片的《任务组》（参见 **2.6.8** ）的验证者转发到目标分片的《任务组》的指定块生成者（参见 **2.6.9** ）;这可以直接完成，而不必沿着超立方体的边缘。当带有默克尔树证明的信息到达目的地分片链的验证者（更准确地说，是对照者;参见 **2.6.5** ）时，他们可以立即将其提交到新的块中，而无需等待信息完成其沿着《慢路径》的行进。然后沿着超立方体边缘发回确认交付以及合适的默克尔树证明，并且可以通过提交特殊交易来使其沿着《慢路径》停止信息的行进。

请注意，这种《即时交付》机制并不能取代 **2.4.19** 中描述的《慢速》而防止故障的机制。仍然需要《慢速路径》[18]，因为验证者不能因丢失或仅仅决定不将《快速路径》信息提交到其区块链的新块而受到惩罚。

因此，两个信息转发方法并行运行[19]，并且只有在将《快速》机制的成功证明提

[17] 这不一定是用于计算超立方体路由的下一跳的算法的最终版本。特别地，十六进制数字可以由 r 位组代替，其中 r 是可配置参数，不一定等于四。

[18] 然而，验证者有一些动机尽快这样做，因为他们将能够收集与慢速路径上尚未消耗的信息相关的所有转发费用。

[19] 事实上，人们可能暂时或永久地禁用《即时交付》机制；系统将继续工作，尽管速度较慢。

交到中间分片链中时才会中止《慢速》机制。

2.4.21. 从邻近的分片链的输出队列收集输入信息。 当提出用于分片链的新块时，邻近的分片链的一些输出信息（在 **2.4.19** 的路由超立方体的意义上）被包括在新块中作为《输入》信息并立即传递（即，处理）。 关于必须处理这些邻居的输出信息的顺序，存在某些规则。 基本上，必须在任何《较新》信息之前传递《较旧》信息（来自指向较旧主链块的分片链块）；对于来自相同邻近分片链的信息，必须遵守 **2.4.17** 中描述的输出队列的部分顺序。

2.4.22. 从输出队列中删除信息。 一旦观察到输出队列信息已经被相邻的分片链传递，它就会被特殊交易从输出队列中显式删除。

2.4.23. 防止双重传递信息。 为了防止从相邻分片链的输出队列中获取的信息的双重传递，每个分片链（更确切地说，其中的每个帐户链）将最近传递的信息（或仅仅是它们的哈希）的集合保持为其状态的一部分。 当观察到传递的信息由其始发的相邻分片链从输出队列中删除时（参见 **2.4.22** ），这信息也会从最近传递的信息的集合中被删除。

2.4.24. 转发用于其他分片链的信息。 超立方体路由（参见 **2.4.19** ）意味着有时出站信息不会传递到包含预期接收者的分片链，而是传递到位于到目的地的超立方体路径上的相邻分片链。 在这种情况下，《传递》包括将入站信息移动到出站队列。 这在块中显式地反映为包含信息本身的特殊转发交易。 从本质上讲，这看起来好像是分片链中的某者收到了信息，并且结果生成了一条相同的信息。

2.4.25. 支付转发和保留信息。 转发交易实际上花费一些汽油（取决于转发的信息的大小），因此从代表该分片链的验证者转发的信息的价值中扣除汽油支付。 这种转发支付通常远小于当信息最终传递给其接收者时所提供的汽油支付，即使该信息由于超立方体路由而被多次转发。 此外，只要信息保存在某个分片链的输出队列中，它就是分片链全局状态的一部分，因此特殊交易也可以收集长时间保存全局数据的支付。

2.4.26. 发送到主链与来自主链的信息。 信息可以直接从任何分片链发送到主链，反之亦然。 但是，在主链中发送信息和处理信息的汽油代价非常高，因此只有在真正需要时才会使用此功能——例如，验证者可以存入他们的权益。 在某些情况下，可以定义发送到主链的信息的最小存款（附加值），仅当信息被接收方视为《有效》时才被送回。

信息无法通过主链自动路由。 含有工作链_$id \neq -1$的信息（-1是表示主链的特殊

工作链_*id*）无法传递给主链。

原则上，可以在主链内创建信息转发智能合约，但使用它的代价将是令人望而却步的。

2.4.27. 同一个分片链中的帐户之间的信息。 在某些情况下，信息由属于某个分片链的帐户生成，该帐户将发往同一个分片链中的另一个帐户。 例如，这发生在一个尚未拆分为多个分片链的新工作链中，因为负载是可应付的。

此类信息可能会累积在分片链的输出队列中，然后作为后续块中的传入信息进行处理（为此目的，任何分片都被视为其自身的邻居）。 但是，在大多数情况下，可以在始发块本身内传递这些信息。

为了实现这一点，对分片链块中包括的所有交易施加部分顺序，并且以该部分顺序处理交易（每个交易包括向某个帐户传递信息）。 特别地，允许交易以该部分顺序处理先前交易的一些输出信息。

在这种情况下，信息正文不会被复制两次。 相反，始发和处理交易涉及的是信息的共享副本。

2.5 全局分片链状态。《细胞袋子》的哲学

现在我们准备描述电报开放网络(TON)区块链的全局状态，或者至少是基本工作链的分片链。

我们从《高级》或《逻辑》描述开始，其中包括说全局状态是代数类型分片链状态的值。

2.5.1. 分片链状态作为帐户链状态的集合。 根据无限分片范例（参见 **2.1.2** ），任何分片链只是一个（临时）虚拟《帐户链》集合，每个只包含一个帐户。 这意味着，本质上，全局分片链状态必须是一个*Hashmap*：

$$\text{分片链状态} \quad (\text{账户} \dashrightarrow \text{账户状态}) \tag{23}$$

所有出现在这个*Hashmap*的索引中的 账户_*id* 必须以前缀 s 开头，如果我们正在讨论分片(w, s)的状态（参见 **2.1.8** ）。

在实践中，我们可能希望将*AccountState*分成几个部分（例如，将帐户输出消息队列分开以简化其相邻分片链的检查），并在*ShardchainState*内部有几个*Hashmap*（账户 $\dashrightarrow$ 账户状态部分）。 我们还可以向*ShardchainState*添加少量《全局》或《整数》参数

（例如，属于该分片的所有帐户的总余额，或所有输出队列中的消息总数）。

然而，(23) 是分片链全局状态看起来的良好的第一近似，至少从《逻辑》（《高级》）的角度来看。 代数类型*AccountState*和*ShardchainState*的形式描述可以借助TL方案（参见**2.2.5** ）完成，在其他地方提供。

2.5.2. 拆分和合并分片链状态。 请注意，分片链状态(23)的无限分片范例描述显示了在分割或合并分片时应如何处理此状态。 实际上，这些状态转换结果是使用哈希图的非常简单的操作。

2.5.3. 账户链状态。 （虚拟）帐户链状态只是一个帐户的状态，由账户状态类型描述。 通常它具有 **2.3.20** 中列出的全部或部分字段，具体取决于所使用的具体构造函数。

2.5.4. 全局工作链状态。 与(23)类似，我们可以通过相同的公式定义全局工作链状态，但允许使用账户*_id*获取任何值，而不仅仅是属于一个分片的值。 类似于 **2.5.1** 中的注释也适用于这种情况：我们可能希望将此*Hashmap*拆分为好几个*Hashmap*，我们可能希望添加一些《整数》参数，例如总余额。

本质上，全局工作链状态必须由与分片链状态相同的分片链状态类型给出，因为如果此工作链的所有现有分片链突然合并为一个，我们将获得此分片链状态。

2.5.5. 低级视角：《细胞袋》。 对于帐户链或者分片链状态存在《低级》描述，与上面给出的《高级》描述互补。 这个描述非常重要，因为它非常普遍，为网络表示，存储，序列化和传输几乎所有电报开放网络(TON)区块链使用的数据提供了通用基础（块，分片链状态，智能合约存储，默克尔树证明，等）。 同时，这种普遍的《低级》描述一旦被理解和实施，就可以使我们只关注《高级》考虑。

回想一下，TVM通过TVM细胞树或简称细胞（参见 **2.3.14** 和 **2.2.5** ）表示任意代数类型的值（例如，包括 (23) 的分片链状态）。

任何这样的细胞由两个描述符字节组成，定义了某些标志和值 $0 \le b \le 128$，原始字节的数量，以及 $0 \le c \le 4$，即对其他细胞的引用数量。 然后是 b 个原始字节和 c 个细胞引用。 [20]

细胞参考的确切格式取决于实施方式以及细胞是不是位于内存中，磁盘上，网络包中，块中等等。 一个有用的抽象模型在于想象所有细胞都保存在内容可寻址的记忆中，细胞的地址等于其(SHA256)哈希。 回想一下，细胞的（默克尔）哈希是通过用它们（递归计算的）哈希取代对其子细胞的参考并对所得到的字节串进行哈希来精确计算的。

[20] 可以证明，如果同样经常需要存储在细胞树中的所有数据的默克尔树证明，则应使用 $b + ch \approx 2(h + r)$ 的细胞来最小化平均默克尔树证明大小，其中 $h = 32$ 是以字节为单位的哈希大小，$r \approx 4$是生物小区参考的《字节大小》。 换句话说，细胞应包含两个引用和几个始发字节，或一个引用和大约三十六个始发字节，或者根本不包含七十二个始发字节的引用。

以这种方式，如果我们使用细胞哈希来参考细胞（例如，其他细胞的内部描述），则系统稍微简化，并且细胞的哈希开始与表示它的字节串的哈希一致。

现在我们看到TVM表示的任何对象，包括全局分片链状态，可以表示为《细胞袋》——即细胞的集合以及对其中一个的《根》引用（例如，哈希值）。 注意，从该描述中移除了重复的细胞（《细胞袋》是一组细胞，而不是多组细胞），因此抽象树表示实际上可能变成有向无环图(dag)表示。

有人甚至可能将这种状态保存在 *B-* 或 *B+* 树的磁盘上，包含所讨论的所有细胞（可能带有一些额外的数据，如子树高度或参考计数器），由细胞哈希索引。 然而，这种想法的幼稚的实现会导致一个智能合约的状态分散在磁盘文件的远端部分，我们宁愿避免这种情况。[21]

现在我们将详细解释电报开放网络(TON)区块链使用的几乎所有物体如何都可以表示为《细胞袋》，从而证明这种方法的普遍性。

2.5.6. 分片链块作为“细胞袋”。 分片链块本身也可以用代数类型描述，并存储为《细胞袋》。 然后，可以简单地通过以任意顺序连接表示《细胞袋》中的每个细胞的字节串来获得块的幼稚的二进制表示。 例如，通过在块的开始处提供所有细胞的偏移列表，并且尽可能用该列表中的三十二位索引替换对其他细胞的哈希引用，可以改进和优化该表示。 然而，人们应该想象一个块本质上是一个《细胞袋》，所有其他技术细节只是次要的优化和实施问题。

2.5.7. 将对象更新为《细胞袋》。 想象一下，我们有一个旧版本的某个对象表示为《细胞袋》，我们想要代表同一个对象的新版本，据说与前一个对象没有太大的不同。 人们可能只是简单地将新状态表示为具有其自己的根的另一个《细胞袋》，并从中移除旧版本中出现的所有细胞。 剩下的《细胞袋》基本上是对象的更新。 拥有该对象的旧版本和更新的每个人都可以计算新版本，只需将两袋细胞联合起来，然后移除旧根（如果参考计数器变为零，则减小其参考计数器并解除分配细胞）。

2.5.8. 更新帐户状态。 特别是，可以使用 **2.5.7** 中描述的思想来表示对帐户状态或分片链的全局状态或任何*Hashmap*的更新。 这意味着当我们收到一个新的分片链块（它是一个《细胞袋》）时，我们不仅仅是单独解释这个《细胞袋》，而是将它首先与代表分片链的前一状态的《细胞袋》结合起来。在这个意义上，每个块可以《包含》区块链的整个状态。

[21] 更好的实现方法是将智能合约的状态保持为序列化字符串（如果它很小），或者保存在单独的 *B* 树中（如果它很大）；那么代表区块链状态的顶级结构将是一个 *B* 树，其叶子被允许包含对其他*B*树的引用。

2.5.9. 更新块。 回想一下块本身就是一个《细胞袋》，所以，如果有必要编辑一个块，可以类似地将《块更新》定义为《细胞袋》，在《细胞袋》的存在的情况下进行解释，这是该块的先前版本。这大致是 **2.1.17** 中讨论的《垂直块》背后的想法。

2.5.10. 以默克尔树证明为一个《细胞袋》。 请注意，(广义) 默克尔树证明——例如，一个断言 $x[i]=y$ 从已知值HASH(x) $=h$（参见 **2.3.10** 和 **2.3.15** ）开始——也可以表示为《一袋细胞》。 也就是说，只需要提供一个细胞子集，该子集对应于从 x：Hashmap(n, X)的根到其所需叶子的路径，其索引为 $i:2^n$，值为 $y:X$。这些不是在这条路径上的细胞的子代的引用将在此证明中《未解决》，由细胞哈希表示。 也可以通过在《细胞袋》中包括位于 x 的根部的两条路径的并集上的细胞，来提供例如 $x[i]=y$ 和 $x[i']=y'$ 的同时默克尔树证明。 对应于索引 i 和 i' 的叶子。

2.5.11. 默克尔树证明作为来自完整节点的查询响应。本质上，具有分片链（或帐户链）状态的完整副本的完整节点可以在轻节点（例如，运行电报开放网络(TON)区块链客户端的轻型版本的网络节点）请求时提供默克尔树证明，从而启用接收者在没有外部帮助的情况下执行一些简单的查询，仅使用此默克尔树证明中提供的细胞。 轻节点可以将序列化格式的查询发送到整个节点，并接收正确的答案再加默克尔树证明，或只接收默克尔树证明，因为请求者应该只能使用默克尔树证明中包含的细胞来计算答案。这个默克尔树证明只包含一个《细胞袋》，只包含那些属于分片链状态的细胞，这些细胞在执行轻节点的查询时已被完整节点进接过。这种方法尤其可用于执行智能合约的《获取查询》（参见 **4.3.12** ）。

2.5.12. 使用默克尔有效性证明进行增强更新或状态更新。 回想一下（参见 **2.5.7** ）我们可以通过《更新》来描述对象状态从旧值 $x:X$ 到新值 $x':X$ 的变化，这只是一个《细胞袋》， 包含代表新值 x' 的子树中的那些细胞，但不包含代表旧值 x 的子树中的细胞，因为假定接收者具有旧值 x 及其所有细胞的副本。

但是，如果接收者没有 x 的完整副本，但只知道它的（默克尔）哈希 $h=$ HASH(x)，它将无法检查更新的有效性（即，更新中的所有《悬空》细胞引用都指的是 x 树中存在的细胞）。 人们希望得到《可验证的》更新，并通过默克尔树证明旧状态中存在所有被引用的细胞。 那么任何只知道 $h=$ HASH(x)的人都能够检查更新的有效性并自己计算新的 $h'=$ HASH(x')。

因为我们的默克尔树证明本身就是《细胞袋》（参见 **2.5.10** ），所以可以将这样的增强更新构建为《细胞袋》，其中包含 x 的旧根，其中的一些后代以及来自 x 的根到它们，以及 x' 的新根和它的不属于 x 的所有后代。

2.5.13. 分片链块中的帐户状态更新。 特别是，应该扩充分片链块中的帐户状态更新，如 **2.5.12** 中所述。 否则，有人可能会提交一个包含无效状态更新的块，指的是旧状态中缺少的细胞; 证明这样一个区块无效将是有问题的（挑战者如何证明一个细胞不是以前状态的一部分?）。

如果块中包含的所有状态更新都得到了增强，则可以轻松检查其有效性，并且它们的无效性也很容易显示为违反（广义）默克尔哈希的递归定义属性。

2.5.14. 《一切都是一袋细胞》的哲学。 先前的考虑表明，我们需要在电报开放网络(TON)区块链或网络中存储或传输的所有东西都可以表示为《细胞袋》。 这是电报开放网络(TON)区块链设计理念的重要组成部分。 一旦解释了《细胞袋》方法并定义了《细胞袋》的一些《低级》序列化，就可以简单地在高级抽象（依赖于高级抽象的）代数数据类型上定义所有内容（块格式，分片链和帐户状态等）。

《一切都是一袋细胞》的理念的统一效果大大简化了看似无关的服务的实施; 参见 **5.1.9** 举例涉及支付频道。

2.5.15. 阻止电报开放网络(TON)区块链的块《标头》。 通常，区块链中的块以小标头开始，包含前一个块的哈希，其创建时间，块中包含的所有交易的树的默克尔哈希，等等。 然后将块哈希定义为该小块标头的哈希。 因为块标头最终取决于块中包含的所有数据，所以不能在不改变其哈希的情况下改变块。

在电报开放网络(TON)区块链块使用的《细胞袋》方法中，没有指定的块标头。 相反，块哈希被定义为块的根细胞的（默克尔）哈希。 因此，块的顶部（根）细胞可能被认为是该块的小《标头》。

但是，根细胞可能不包含通常从这种标头中预期的所有数据。 实质上，人们希望标头包含块数据类型中定义的一些字段。 通常，这些字段将包含在几个细胞中，包括根目录。 这些是共同构成所讨论的字段值的《默克尔树证明》的细胞。 有人可能会在任何其他细胞之前的一开始就坚持要求一个块包含这些《标头细胞》。 然后，只需要下载块序列化的前几个字节，以获得所有《标头细胞》，并得知所有预期的字段。

2.6 创建和验证新块

电报开放网络(TON)区块链最终由分片链和主链块组成。 必须通过网络创建，验证这些块并将其传播给所有相关方，以使系统顺利，正确地运行。

2.6.1. 验证者。 新块由特殊的指定节点（称为验证者）创建和验证。 基本上，任何希望

成为验证者的节点都可以成为一个验证者，只要它能够将足够大的权益（用TON代币，即Gram;参见附录 **A** ）存入主链。 验证者获得了一些好的工作的《奖励》，即从新生成的块中提交的所有交易（消息）中的交易费，存储费和汽油费，以及一些新铸造的代币，反映了整个社区对验证者的《感激》。 保持电报开放网络(TON)区块链正常工作。该收入按比例分配给所有参与的验证者。

然而，作为验证者是一项高度责任。 如果验证者签署了无效区块，则可以通过丢失其部分或全部股权来惩罚，并暂时或永久地从验证者集中排除。 如果验证者不参与创建块，则它不会收到与该块相关的奖励的份额。 如果验证者长期放弃创建新块，它可能会丢失部分权益，并被暂停或永久排除在验证者集之外。

所有这些意味着验证者不会《无所事事》获得金钱。 实际上，它必须跟踪所有或一些分片链的状态（每个验证者负责验证和创建分片链的某个子集中的新块），执行这些分片链中智能合约请求的所有计算，接收有关其他分片链的更新，等等。 此活动需要相当大的磁盘空间，计算能力和网络带宽。

2.6.2. 验证者而不是矿工。 回想一下，电报开放网络(TON)区块链使用了权益证明方法，而不是比特币和当前版本的以太坊以及大多数其他加密货币所采用的工作量证明方法。这意味着人们不能通过提出一些工作量证明（计算许多其他无用的哈希）来《挖掘》一个新的块，并因此获得一些新的代币。 相反，必须成为验证者并花费计算资源来存储和处理电报开放网络(TON)区块链请求和数据。 简而言之，开采新代币的必须是验证者。在这方面，验证者是新的矿工。

然而，除了作为验证者之外，还有其他一些赚代币的方法。

2.6.3. 提名者和《共享挖掘》。 要成为验证者，通常需要购买并安装多个高性能服务器并为他们获得良好的电网连接。 这并不像目前开发比特币所需的ASIC设备那么昂贵。 然而，绝对不能在家用电脑上挖掘新的电报开放网络(TON)代币，更不用说智能手机了。

在比特币，以太坊和其他工作证明加密货币挖掘社区中，有一个共享挖掘的概念，其中许多节点，没有足够的计算能力来自己挖掘新块，结合他们的努力并在之后分享奖励。 利益证明世界中的相应概念是提名者的概念。 从本质上讲，这是一个节点贷款，以帮助验证者增加其权益; 验证者然后将其奖励的相应份额（或之前商定的一部分——比如百分之五十）分配给提名者。

通过这种方式，提名者也可以参与《采矿》并获得与其愿意为此目的存入的金额成比例的一些奖励。 它仅获得验证者奖励的相应份额的一小部分，因为它仅提供《资本》，但不需要购买计算能力，存储和网络带宽。

但是，如果验证者因无效行为而失去其权益，则提名者也会失去其权益份额。

从这个意义上说，提名者分担风险。 它必须明智地选择其指定的验证者，否则它可能会赔钱。 从这个意义上说，提名者做出有倾向的决定，并用他们的资金对某些验证者进行《投票》。

另一方面，这个提名或借出系统使人们能够成为验证者，而无需先向Gram（电报开放网络代币）投入大量资金。 换句话说，它可以防止那些保留大量Gram的人垄断了验证者的供应。

2.6.4. 渔民：通过指出别人的错误来获取金钱。 获得一些奖励而不做验证者的另一种方法是成为一名渔民。 基本上，任何节点都可以通过在主链中存入少量存款而成为渔民。 然后，它可以使用特殊的主链事务来发布先前由验证者签名和发布的一些（通常是分片链）块的（默克尔）无效证明。 如果其他验证者同意此无效证明，则违规验证者将受到惩罚（通过丢失部分权益），并且渔民获得一些奖励（从违规验证者处没收的一小部分代币）。 之后，必须按照 **2.1.17** 中的描述更正无效（分片链）块。 纠正无效的主链块可能涉及在先前提交的主链块之上创建《垂直》块（参见 **2.1.17** ）；没有必要创建主链的分叉。

通常，渔民需要成为至少一些分片链的完整节点，并通过运行至少一些智能合约的代码来花费一些计算资源。 虽然渔民不需要具有与验证者一样多的计算能力，但我们认为成为渔民的自然候选者是准备处理新区块的潜在验证者，但尚未被选为验证者（ 例如，由于未能存放足够大的权益）。

2.6.5. 对照者：通过向验证者建议新块来获取金钱。 另一种获得一些奖励而不是验证者的方法是成为一名对照者。 这是一个节点，它准备并向验证者建议新的分片链块候选者，补充（对照）从该分片链和其他（通常是相邻的）分片链的状态获取的数据，以及合适的默克尔树证明。 （例如，当某些消息需要从相邻的分片链转发时，这是必要的。）然后验证者可以轻松地检查所建议的块候选者的有效性，而无需下载此或其他分片链的完整状态。

因为验证者需要提交新的（对照的）区块候选者以获得一些（《挖掘》）奖励，所以将一部分奖励支付给愿意提供合适的区块候选者的对照者是合理的。 通过这种方式，验证者可以通过将其外包给对照者来摆脱观察相邻分片链状态的必要性。

但是，我们希望在系统初始部署阶段不会有单独的指定对照者，因为所有验证者都可以自己充当对照者。

2.6.6. 对照者或验证者：获取包括用户交易的金钱。 用戶可以向一些对照者或验证者开

放微支付频道，并支付少量代币以换取在分片链中包含他们的交易。

2.6.7. 全局验证者集选举。 每个月选出一集《全局》验证者（实际上，每2^{19} 个主链块）。 该集是提前一个月确定并普遍知晓的。

为了成为验证者，节点必须将一些电报开放网络(TON)代币(Gram)转移到主链中，然后将它们发送到特殊的智能合约作为其建议的权益 s。 与权益一起发送的另一个参数是 $l \geq 1$，该节点愿意接受的最大验证负载相对于最小可能。 l 上还有一个全局上界（另一个可配置参数）L，等于10。

然后，通过这个智能合约选择全局验证者集，只需选择最多具有最大建议权益的 T 候选者并发布他们的身份。 最初，验证者的总数是 $T = 100$; 我们预计随着负载的增加它会增长到 1000。 它是一个可配置的参数（参见 **2.1.21** ）。

每个验证者的实际利益计算如下：如果最高 T 提议的权益是 $s_1 \geq s_2 \geq \ldots \geq s_T$，则第 i 个验证者的实际权益设置为 $s_i' \ min(s_i, l_i, s_T)$。 这样，$s_i' / s_T' \leq l_i$，所以第 i 个验证者获得的不是 $l_i \leq L$ 倍最弱的验证者的负载（因为负载最终与权益成比例）。

然后当选的验证者可以撤回其未使用的部分，即 $s_i - s_i'$。 不成功的验证者候选者可以撤回他们所有的提议的权益。每个验证者发布其公共签名密钥，不一定等于权益来源的帐户的公钥。[22]

验证者的权益被冻结，直到他们被选出的期间结束，并且如果出现新的争议（即，找到由这些验证者之一签署的无效区块），则会延长一个月。 在此之后，将返还权益，以及验证者的代币份额和在此期间处理的交易的费用。

2.6.8. 选举验证者《任务组》。 整个全局验证者集（其中每个验证者被认为具有等于其权益的多重性——否则验证者可能倾向于假设几个身份并将其中的利益分开）仅用于验证新的主链块。 分片链块仅通过特别选择的验证者子集进行验证，这些验证者子集取自 **2.6.7** 中所述选择的全局验证者集。

为每个分片定义的这些验证者《子集》或《任务组》每小时轮换一次（实际上，每2^{10}个主链块），并且它们是提前一小时知道的，这样每个验证者都知道验证需要哪些分片，并且可以为此做好准备（例如，通过下载丢失的分片链数据）。

用于为每个分片(w, s)选择验证者任务组的算法是确定性伪随机数。 它使用验证者嵌入到每个主链块中的伪随机数（例如，通过使用阈值签名[4] [10]的共识生成）来创建随机种子，然后计算例如HASH(代码(w).代码(s).验证者_*id.rand*_种子) 给每个验证者。 然后验证者按此哈希的值进行排序，并选择前几个验证者，以便至少有 $20/T$ 的验证者总权益，并且至少包含五个验证者。

[22] 为每个验证者选举生成和使用新密钥对是行得通的。

这种选择可以通过特殊的智能合约来完成。 在这种情况下，选择算法很容易升级，而不需要 **2.1.21** 中提到的投票机制的硬分叉。 到目前为止提到的所有其他《常数》（例如 2^{19} ，2^{10} ，T，20 和 5）也是可配置的参数。

2.6.9. 在每个任务组上轮换优先级顺序。 根据先前主链块和（分片链）块序列号的哈希，对分片任务组的成员施加了某个《优先级》顺序。 如上所述，通过生成和排序一些哈希来确定该顺序。

当需要生成新的分片链块时，选择创建此块的分片任务组验证者通常是关于此旋转《优先级》顺序的第一个。 如果它无法创建块，则第二个或第三个验证者可以执行此操作。 基本上，所有这些都可以建议他们的块候选者，但具有最高优先级的验证者建议的候选者应该作为拜占庭容错(BFT)共识协议的结果而获胜。

2.6.10. 分片链块候选者的传播。 由于分片链任务组成员资格是提前一小时知道的，因此他们的成员可以使用电报开放网络(TON)的一般机制（参见 **3.3** ），利用该时间构建专用的《分片验证者多播覆盖网络》。当需要生成新的分片链块时——通常在最近的主链块传播后一两秒钟——每个人都知道谁具有最高优先级来生成下一个块（参见 **2.6.9** ）。该验证者将自行创建一个新的对照块候选者，或者在对照者的帮助下（参见 **2.6.5** ）。验证者必须检查（验证）此块候选者（特别是如果它已由某个对照者准备好）并使用其（验证者）私钥对其进行签名。然后使用预先安排的多播覆盖网络将块候选者传播到任务组的其余部分（任务组创建其自己的专用覆盖网络，如 **3.3** 中所述，然后使用 **3.3.15** 中描述的流式多播协议的版本来传播块候选者）。

真正的拜占庭容错方式是使用拜占庭组播协议，例如蜜獾拜占庭容错 [18]中使用的协议：用 (N, $2N/3$)-擦除代码对块候选者进行编码，发送 $1/N$ 结果数据直接发送给组的每个成员，并期望它们将其部分数据直接组播到组的所有其他成员。

然而，更快更直接的方法（参见 **3.3.15** ）是将块候选者分成一系列有符号的一千字节块（《块》），用里德-所罗门码或者一个喷泉码（例如RaptorQ代码[15] [21]或在线代码[17]）来增加它们的序列，并开始向《多播网格》（即覆盖网络）中的邻居发送块，期望它们进一步传播这些块 。 一旦验证者获得足够的块以从它们重建块候选者，它就签署确认收据并通过其邻居将其传播到整个组。 然后它的邻居停止向它发送新的块，但是可以继续发送这些块的（始发）签名，相信该节点可以通过自己应用里德-所罗门码或喷泉码（具有所有必要的数据）来生成后续块，将它们与签名结合起来，并传播到尚未准备好的邻居。

如果在删除所有《坏》节点之后《多播网格》（覆盖网络）保持连接（回想一下，以拜占庭方式允许多达三分之一的节点是坏的，即，以任意恶意方式行事），此算

法将尽快传播块候选者。 不仅指定的高优先级块创建者可以将其块候选者多播到整个组。优先级的第二和第三验证者可以立即或者在未能从最高优先级验证者接收块候选者之后开始多播它们的块候选者。 但是，通常只有具有最大优先级的块候选者将由所有（实际上，至少为任务组的三分之二）验证者签名并作为新的分片链块提交。

2.6.11. 验证块候选者。 一旦验证者接收到块候选者并且检查了其始发验证者的签名，则接收验证者通过执行其中的所有交易并检查它们的结果是否与所声明的那个结果一致来检查该块候选者的有效性。 从其他区块链导入的所有消息必须由对照数据中的合适默克尔树证明支持，否则块候选者被视为无效（并且，如果将此证明提交给主链，则已经签署此块候选者的验证者可能会受到惩罚）。 另一方面，如果发现块候选者有效，则接收验证者对其进行签名，并通过《网状多播网络》或直接网络消息将其签名传播到该组中的其他验证者。

我们想强调一个验证者不需要进接这个或相邻的分片链的状态，以便检查（已对照的）块候选者的有效性。[23] 这允许验证非常快速地进行（没有磁盘进接），并且减轻验证者的计算和存储负担（特别是如果他们愿意接受外部对照者的服务来创建块候选者）。

2.6.12. 选举下一个候选块。 一旦块候选者收集任务组中验证者的有效性签名的至少三分之二（按比例），它就有资格被提交为下一个分片链块。 拜占庭容错协议运行为了对所选择的块候选者（可能存在多于一个）实现一致，所有《好》验证者优先选该轮的具有最高优先级的块候选者。 作为运行该协议的结果，该块通过至少三个验证者（通过权益）的签名来增强。 这些签名不仅证明了所讨论的块的有效性，而且证明了它是由拜占庭容错协议选出的。 之后，块（没有对照的数据）与这些签名组合，以确定的方式序列化，并通过网络传播给所有相关方。

2.6.13. 验证者必须保留他们签名的块。 在他们加入任务组期间以及之后至少一个小时（或更确切地说是2^{10}个块）之前，验证者应该保留他们签署和提交的块。 未能向其他验证者提供签名块可能会受到惩罚。

2.6.14. 将新的分片链块的标头和签名传播到所有验证者。 验证者使用类似于为每个任务组创建的组播网状网络，将新生成的分片链块的标头和签名传播到全局验证者集。

2.6.15. 生成新的主链块。 在生成所有（或几乎所有）新的分片链块之后，可以生成新的

[23] 可能的例外是相邻分片链的输出队列状态，这是保证 **2.4.21** 中描述的消息排序要求所必需的，因为在这种情况下默克尔树证明的大小可能变得过高。

主链块。 该过程与分片链块（参见 **2.6.12** ）基本相同，不同之处在于所有验证者（或至少三分之二）必须参与此过程。 因为新的分片链块的标头和签名被传播到所有验证者，所以每个分片链中最新块的哈希值可以且必须包含在新的主链块中。 一旦将这些哈希值提交到主链块中，外部观察者和其他分片链就可以认为新的分片链块已提交且不可变（参见 **2.1.13** ）。

2.6.16. 验证者必须保持主链的状态。 主链和分片链之间的一个值得注意的区别是，所有验证者都需要跟踪主链状态，而不依赖于对照数据。 这很重要，因为验证者任务组的知识来自主链状态。

2.6.17. 分片链块是并行生成和传播的。 通常，每个验证者都是几个分片链任务组的成员；它们的数量（因此验证者上的负载）大约与验证者的权益成比例。 这意味着验证者并行运行新的分片链生成块协议的几个实例。

2.6.18. 缓解块保留攻击。 因为验证者的总集合在仅看到其标题和签名之后将新的分片链块的散列插入到主链中，所以生成此块的验证者很可能会合谋并试图避免完整地发布新块。这将导致相邻分片链的验证者无法创建新块，因为一旦将其哈希值提交到主链中，它们必须至少知道新块的输出消息队列。

为了缓解这种情况，新块必须收集来自其他验证者的签名（例如，相邻分片链的任务组联合的三分之二），证明这些验证者确实有此块的副本，并且如果需要，愿意将它们发送给任何其他验证者。只有在这些签名出现后，新块的哈希才能包含在主链中。

2.6.19. 主链块比分片链块生成更晚。主链块大约每五秒生成一次，分片链块也是如此。然而，虽然所有分片链中的新块的生成基本上同时运行（通常由新的主链块的释放触发），但是故意延迟生成新的主链块，以允许在主链中包含新生成的分片链块的哈希值。

2.6.20. 慢速验证者可能会收到较低的奖励。 如果验证者《慢》，则可能无法验证新的块候选者，并且可能在没有其参与的情况下收集提交新块所需的签名的三分之二。 在这种情况下，它将获得与此块相关的奖励的较低份额。

这为验证者提供了优化其硬件，软件和网络连接的激励，以便尽可能快地处理用戶交易。

但是，如果验证者在提交之前未能对块进行签名，则其签名可能包含在下一个块之一中，然后包含在部分奖励中（指数级递减，具体取决于生成的块数，例如，0.9^x 如果

验证者迟到 k 块）则仍将给予此验证者。

2.6.21. 验证者签名的《深度》。 通常，当验证者对块进行签名时，签名仅证明块的相对有效性：如果此块和其他分片链中的所有先前块都有效，则此块有效。 验证者不能因为将前面的块中提交的无效数据视为理所当然而受到惩罚。

但是，块的验证者签名具有称为《深度》的整数参数。 如果它不为零，则意味着验证者也断言指定数量的先前块的（相对）有效性。 这是《缓慢》或《暂时离线》验证者捕获并签署一些未经签名提交的块的方法。 然后仍会给予他们部分奖励（参见 **2.6.20** ）。

2.6.22. 验证者负责签名的分片链块的相对有效性; 绝对有效性如下。 我们想再次强调，在分片链块 B 上的验证者签名仅证明该块的相对有效性 (或者如果签名具有《深度》d，则可能也是 d 先前块的相对有效性，参见 **2.6.21** ; 但这并不会影响以下的讨论) 换句话说，验证者断言通过应用 **2.2.6** 中描述的块评估函数 *ev_block*，从先前状态 s 获得了分片链的下一个状态 s'：

$$s' = ev\_block(B)(s) \tag{24}$$

以这种方式，如果始发状态 s 证明是《不正确的》（例如，由于先前块之一的无效），则不能惩罚签名块 B 的验证者。 渔民（参见 **2.6.4** ）只有在发现相对无效的区块时才应该投诉。 权益证明系统作为一个整体努力使每个块相对有效，而不是递归（或绝对）有效。但是请注意，如果区块链中的所有区块都相对有效，那么所有区块和整个区块链都是绝对有效的; 使用区块链长度的数学归纳可以很容易地显示这个陈述。 通过这种方式，可以轻松验证的块的相对有效性的断言一起证明了整个区块链的绝对有效性。

注意，通过对块 B 进行签名，验证者在给定始发状态 s 的情况下断言该块是有效的（即，(24) 的结果不是值 $\perp$ 表示不能计算下一个状态）。 以这种方式，验证者必须对在 (24) 的评估期间进接的始发状态的细胞进行最小的正式检查。

例如，假设期待包含从提交到块中的交易进接的帐户的始发余额的细胞原来具有零始发字节而不是预期的八或十六。然后始发余额根本无法从细胞中检索出来，并在尝试处理块时发生《未处理的异常》。 在这种情况下，验证者不应该在受到惩罚的下场上签署这样的阻止。

2.6.23. 签署主链块。 主链块的情况有所不同：通过签署主链块，验证者不仅断言其相对有效性，而且断言所有先前块的相对有效性，直到该验证者承担其责任时的第一个块

（但不是更进一步）。

2.6.24. 验证者的总数。到目前为止所描述的系统中，要选择的验证者总数的上限 T（参见 **2.6.7** ）不能超过，比如几百或一千，因为所有验证者都应该参与拜占庭容错共识协议用于创建每个新的主链块，并且不清楚这些协议是否可以扩展到数千个参与者。更重要的是，主链块必须收集所有验证者中至少三分之二的签名（通过权益），并且这些签名必须包含在新块中（否则系统中的所有其他节点都没有理由相信没有自己验证的新块）。如果超过一千个（每个主链块中必须包含一千个验证者签名），这将意味着每个主链块中有更多数据，[24] 将由所有完整节点存储并通过网络传播，并且花费更多的处理能力检查这些签名（在权益证明系统中，完整节点不需要自己验证块，但是他们需要检查验证者的签名）。

虽然限制 T 到一千个验证者似乎对电报开放网络(TON)区块链的部署的第一阶段已经足够了，但是当分片链的总数变得如此之大以至于几百个验证者不足以处理时，必须为未来的增长做出规定。 为此，我们引入了一个额外的可配置参数 $T' \leq T$（最初等于 T），并且只有顶部 T' 选举的验证者（按比例）预期会创建和签署新的主链块。

2.6.25. 权力下放的制度。人们可能会怀疑像电报开放网络(TON)区块链这样的证明制度系统，依靠 $T \approx 1000$ 验证者来创建所有分片链和主链块，必然会变得《过于集中》，而不是传统的工作量证明区块链，比如比特币或以太坊，与其每个人（原则上）都可以开采一个新区块，而没有明确的矿工总数上限。

然而，流行的工作量证明区块链，如比特币和以太坊，目前需要大量的计算能力（高《哈希率》）来挖掘新块，并且成功的概率不可忽略。因此，新区块的开采往往集中在几个大型企业的手中，他们向数据中心投入了大量资金，数据中心充满了为采矿优化的定制设计硬件; 并且掌握在几个大型共享挖掘的手中，这些共享挖掘集中并协调了那些无法自己提供足够《哈希率》的大群人的努力。

因此，截至2017年，超过百分之七十五的新的以太坊或比特币区块由不到十名矿工生产。事实上，两个最大的以太坊共享挖掘共同产生了超过一半的新区块！显然，这种系统比依赖 $T \approx 1000$ 节点生成新块的系统集中得多。

人们可能还会注意到成为电报开放网络(TON)区块链验证者所需的投资——即购买硬件（例如，几个高性能服务器）和权益（如果有必要，可以通过一组提名者轻松收集;参见 **2.6.3** ）——比成为一个成功的独立比特币或以太坊矿工所需的要低得多。 事实上， **2.6.7** 的参数L 将迫使提名者不加入最大的《共享挖掘》（即已经积累了最大权益的

[24] 可组合的多重签名（例如，基于[4]的配对）可以在一定程度上减轻空间需求，因为它们只需要一个位图来表示参与多重签名的验证者的子集，并且这种签名的验证是 没有比一个普通签名的验证贵得多。 然而，这种可组合多重签名的时间和空间要求仍然是线性的，尽管具有较小的常数。

验证者），而是寻找目前正在接受提名者资金的小型验证者，甚至创建新的验证者，因为这将允许更高的验证者的比例 s_i'/s_i ——并且还有提名者的——的权益被用，从而从采矿中获得更大的回报。 通过这种方式，TON权益证明系统实际上鼓励分散（创建和使用更多验证者）并惩罚集中化。

2.6.26. 块的相对可靠性。 块的（相对）可靠性只是已签署此块的所有验证者的总权益。换句话说，如果这个区块被证明无效，这就是某些参与者会失去的金额。 如果关注的是转移价值低于块的可靠性的交易，可以认为它们足够安全。 从这个意义上讲，相对可靠性是外部观察者在特定区块中可以具有的信任度量。

请注意，我们说的是块的相对可靠性，因为如果前一个块和所有其他所引用的分片链块有效（参见 **2.6.22** ），则可以保证块有效。

块的相对可靠性在提交后可以增长 - 例如，当添加了迟来的验证者签名时（参见 **2.6.21** ）。 另一方面，如果这些验证者中的一个由于其与其他块相关的不当行为而失去其部分或全部股份，则块的相对可靠性可能降低。

2.6.27. 《加强》区块链。 重要的是为验证者提供激励，以尽可能地提高块的相对可靠性。 实现此目的的一种方法是向验证者分配一小笔奖励，以便将签名添加到其他分片链的块中。 即使是《候选》的验证人，如果他们同意，他们已经存入了不足以通过权益进入最高 T 验证者并被纳入全局验证者组（参见 **2.6.7** ）的权益，可能会参与此活动（如果他们同意的话） 保持他们的权益冻结而不是在失去选举后撤回权益。 这些潜在的验证者可以同时成为渔民（参见 **2.6.4** ）：如果他们反正要检查某些块的有效性，他们干脆选择报告无效块并收集相关的奖励。

2.6.28. 块的递归可靠性。 还可以将块的递归可靠性定义为其相对可靠性的最小值以及它所引用的所有块的递归可靠性（即，主链块，先前的分片链块和相邻的分片链的一些块）。 换句话说，如果该块被证明是无效的，或者因为它本身无效或者因为它所依赖的块之一是无效的，那么至少这个数量的钱将被某人丢失。 如果真的不确定是否该信任块中的某交易，则应该计算该块的递归可靠性，而不仅仅计算相对可靠性。

在计算递归可靠性时退得太后是没有意义的，因为如果我们退得太后，我们将看到由验证者签署的块，其权益已经解冻并撤回。 在任何情况下，我们都不允许验证者自动重新考虑那样旧的块（即，如果它使用当前的可配置参数值，则是在两个月前创建的），并创建从它们开始的分叉或纠正它们，使用《垂直区块链》的帮助（参见 **2.1.17** ），即使它们证明是无效的。 我们假设两个月的时间段为检测和报告任何无效块提供了充足的机会，因此如果在此期间没有对块进行质询，则根本不可能受到质疑。

2.6.29. 轻节点的权益证明的后果。 电报开放网络(TON)区块链使用的权益证明方法的一个重要结果是电报开放网络(TON)区块链的轻节点（运行轻型客户端软件）不需要下载所有分片链甚至主链区块的《标头》以便能够自行检查由完整节点提供给它的默克尔证据的有效性作为其查询的答案。

实际上，因为最新的分片链块哈希包含在主链块中，所以完整节点可以容易地提供默克尔树证明，即给定的分片链块从主链块的已知哈希开始有效。接下来，轻节点只需要知道主链的第一个块（在它那里第一组验证者被宣布），其中（或至少其中的哈希）可以内置到客户端软件中，并且仅大约每个月发生一个主链块，其中宣布新选出的验证者集，因为该块将由前一组验证者签署。从那开始，它可以获得几个最新的主链块，或者至少它们的标头和验证者签名，并将它们用作检查由完整节点提供的默克尔树证明的基础。

2.7 拆分和合并分片链

电报开放网络(TON)区块链最具特色和独特的特征之一是它能够在负载变得过高时自动将分片链分成两部分，并在负载消退时将它们合并（参见 **2.1.10** ）。我们必须详细讨论它，因为它的独特性及其对整个项目可扩展性的重要性。

2.7.1. 分片配置。 回想一下，在任何给定的时刻，每个工作链 w 被分成一个或几个分片链(w, s)（参见 **2.1.8** ）。 这些分片链可以由二叉树的叶子表示，具有根$(w, \varnothing)$，并且每个非叶节点(w, s)具有子 $(w, s.0)$ 和 $(w, s.1)$ 。 通过这种方式，属于工作链 w 的每个帐户都被分配给一个分片，并且知道当前分片链配置的每个人都可以确定包含帐户 账户_*id* 的分片(w, s)：它是唯一一个以二进制字符串 s 作为 账户_*id* 的前缀的分片。

分片配置——即此分片二叉树，或给定 w 的所有活跃 (w, s) 的集合 (对应于分片二叉树的叶子)——是主链状态的一部分，并且对于每个跟踪主链的人都可用。[25]

2.7.2. 最近的分片配置和状态。 回想一下，最新的分片链块的哈希值包含在每个主链块中。 这些哈希以分片二叉树（实际上是树的集合，每个工作链一个）而组织。 这样，每个主链块都包含最新的分片配置。

2.7.3. 宣布并执行分片配置中的更改。可以通过两种方式更改分片配置：分片 (w, s) 可以分为两个分片 $(w, s.0)$ 和 $(w, s.1)$ ，或两个《兄弟》分片 $(w, s.0)$ 和 $(w, s.1)$ 可以合并为一个分片 (w, s) 。

[25] 实际上，分片配置完全由最后一个主链块决定; 这简化了对分片配置的进接。

这些分割和合并操作提前几个（例如, 2^6 ;这是可配置参数）块就被预先通告，首先在相应的分片链块的《标头》中，然后在引用这些分片链块的主链块中。所有相关方都需要该预先通知以准备计划的改变（例如，构建覆盖多播网络以分发新创建的分片链的新块，如 **3.3** 中所讨论的）。然后提交更改，首先进入分片链块 (的标头) (如果是拆分; 对于合并，两个分片链的块应该提交更改)，然后传播到主链块。通过这种方式，主链块不仅定义了创建之前的最新分片配置，还定义了下一个分片配置。

2.7.4. 新分片链的验证者任务组。 回想一下，每个分片（即每个分片链）通常被分配一个验证者子集（验证者任务组），专门用于创建和验证相应分片链中的新块（参见 **2.6.8** ）。这些任务组在一段时间内（大约一小时）被选出，并且提前一段时间（大约一小时）被知道，并且在此期间是不可变的。[26]

但是，由于拆分和合并操作，实际的分片配置可能会在此期间发生更改。必须将任务组分配给新创建的分片。这样做如下：

请注意，任何活跃分片 (w, s) 都将是某些唯一确定的始发分片 (w, s') 的后代，这意味着 s' 是 s 的前缀，或者它将是始发分片 (w, s') 的子树的根，其中 s 将是每个 s' 的前缀。在第一种情况下，我们只需将始发分片 (w, s') 的任务组也作为新分片 (w, s) 的任务组。在后一种情况下，新分片 (w, s) 的任务组将是所有是分片树中 (w, s) 的后代的始发分片 (w, s') 的任务组的并集。

通过这种方式，每个活动分片 (w, s) 都被分配了一个明确定义的验证者子集（任务组）。分割分片时，两个子代都从始发分片继承整个任务组。合并两个分片时，它们的任务组也会合并。

跟踪主链状态的任何人都可以为每个活跃分片计算验证者任务组。

2.7.5. 在始发任务组的责任期间限制拆分/合并操作。 最终，将考虑新的分片配置，并且将自动为每个分片分配新的专用验证者子集（任务组）。在此之前，必须对拆分/合并操作施加一定的限制; 否则，如果始发分片快速分成2^k 个新分片，则始发任务组可能最终同时验证2^k 个分片链，k为大数。

这是通过对可以从始发分片配置（用于选择当前负责的验证者任务组的配置）中移除活动分片配置的距离施加限制来实现的。例如，如果 s' 是 s 的前任（即，s' 是二进制串 s 的前缀），则可能要求分片树中从活动分片 (w,s) 到始发分片 (w, s') 的距离不得超过三。如果 s' 是 s 的后继（即，s 是 s' 的前缀），则不能超过二。否则，不允许拆分或合并操作。

粗略地说，人们在给定验证者任务组的责任期间对分片（例如，三个）或合并（例如，两个）的次数施加限制。在通过合并或拆分创建分片之后，它不能在一段时间

[26] 除非由于签署无效块而临时或永久禁止某些验证者，如果如此它们将自动从所有任务组中排除。

内重新配置（某些块）。

2.7.6. 确定拆分操作的必要性。 分片链的拆分操作由某些形式条件触发（例如，如果对于六十四个连续块，则分片链块至少90%已满）。 这些条件由分片链任务组监视。 如果满足这些条件，则首先在新的分片链块的标头中包括《拆分准备》标志（并且传播到引用该分片链块的主链块）。 然后，在几个块之后，《拆分提交》标志包含在分片链块的标头中（并传播到下一个主链块）。

2.7.7. 执行拆分操作。 在《拆分提交》标志包含在分片链 (w, s) 的块 B 中之后，该分片链中不能有后续块 B'。 相反，将分别创建分片链 $(w, s.0)$ 和 $(w, s.1)$ 的两个块 B_0' 和 B_1'，两者都将块 B 称为它们的前一个块（并且它们都将通过标志指示在标头中：分片已被拆分）。 下一个主链块将包含新分片链的块 B' 和 B_0' 的哈希值; 不允许包含分片链 (w, s) 的新块 B' 的哈希，因为《拆分提交》事件已经被提交到先前的主链块中。

请注意，两个新的分片链将由与旧验证者相同的验证者任务组验证，因此它们将自动获得其状态的副本。 从无限分片范式的角度来看，状态拆分操作本身非常简单（参见 **2.5.2**）。

2.7.8. 确定合并操作的必要性。 以某些形式条件也检测得到分片合并操作的必要性（例如，对于六十四个连续块，两个兄分片链块的大小之和不超过最大块大小的60%）。 这些正式条件还应考虑这些区块所消耗的汽油总量，并将其与当前区块汽油限制进行比较，否则区块可能会很小，因为有一些密集型计算的交易会阻止包含更多交易。

这些条件由兄弟分片 $(w, s.0)$ 和 $(w, s.1)$ 的验证者任务组监视。 请注意，兄弟必然是超立方体路由的邻居（参见 **2.4.19**），因此来自任何分片的任务组的验证者将在某种程度上监视兄弟分片。

当满足这些条件时，任何一个验证者子组（两者之一）都可以通过发送特殊消息向另一个建议它们合并。 然后，它们两个组合成一个临时的《合并任务组》，具有组合成员资格，能够运行拜占庭容错一致性算法，并在必要时传播块更新和块候选。

如果他们就合并的必要性和准备程度达成共识，那么《合并准备》标志将被提交到每个分片链的某些块的标头中，以及兄弟任务组的至少三分之二的验证者的签名（并且传播到下一个主链块，以便每个人都可以为即将进行的重新配置做好准备）。 但是，他们继续为某些预定义数量的块创建单独的分片链块。

2.7.9. 执行合并操作。 之后，当来自两个始发任务组的并集的验证者准备好成为合并的分片链的验证者时（这可能涉及从兄弟分片链和状态合并操作的状态转移），它们提交

《合并提交》标志在他们的分片链块的标头中（此事件传播到下一个主链块），并停止在单独的分片链中创建新块（一旦出现合并提交标志，则禁止在单独的分片链中创建块）。 相反，合并的分片链块被创建（由两个始发任务组的并集），在其《标头》中引用它的两个《前面的块》。 这反映在下一个主链块中，它将包含新创建的合并分片链块的哈希值。 之后，合并的任务组继续在合并的分片链中创建块。

2.8 区块链项目的分类

我们将通过将电报开放网络(TON)区块链与现有和提议的区块链项目进行比较来结束对TON区块链的简要讨论。 然而，在此之前，我们必须对区块链项目进行充分的一般分类。 基于此分类的特定区块链项目的比较推迟到 **2.9** 。

2.8.1. 区块链项目的分类。 作为第一步，我们建议区块链的一些分类标准（即区块链项目）。 任何此类分类都有些不完整和肤浅，因为它必须忽略所考虑项目的一些最具体和独特的特征。 但是，我们认为这是至少提供区块链项目区域的粗略和近似地图的必要的第一步。

我们考虑的标准清单如下：

- 单区块链与多区块链架构（参见 **2.8.2** ）
- 共识算法：权益证明与工作量证明（参见 **2.8.3** ）
- 对于权益证明系统，使用了精确的块生成，验证和一致性算法（两个主要选项是委任权益证明与拜占庭容错;参见 **2.8.4** ）
- 支持“任意”（图灵完备）智能合约（参见 **2.8.6** ）

多区块链系统有额外的分类标准（参见 **2.8.7** ）：

- 成员区块链的类型和规则：同构，异构（参见 **2.8.8** ），混合（参见 **2.8.9** ）。 联盟（参见 **2.8.10** ）。
- 内部或外部的主链的不存在或存在（参见 **2.8.11** ）

- 原生支持分片（参见 **2.8.12** ）。 静态或动态分片（参见 **2.8.13** ）。
- 成员区块链之间的相互作用：松散耦合和紧密耦合的系统（参见 **2.8.14** ）

2.8.2. 单区块链与多区块链项目。第一个分类标准是系统中区块链的数量。最老，最简单的项目包括一个区块链（简称《单链项目》）;更复杂的项目使用（或者更确切地说，计划使用）多个区块链（《多链项目》）。

单链项目通常更简单，被测试得更好;他们经受住了时间的考验。它们的主要缺点是低性能，或者至少是低交易吞吐量，对于通用系统而言，其在每秒十（比特币）到小于一百 [27] （以太坊）个交易的水平上。一些特殊系统（例如比特股）能够每秒处理数万个专用交易，代价是处理区块链状态以适应内存，并将处理限制为预定的特殊交易集，然后执行, 通过用C++等语言编写的高度优化的代码（这里没有VM）。

多链项目承诺每个人都渴望的可扩展性。 它们可能支持更大的总状态和每秒更多的交易，但代价是使项目更加复杂，并且其实现更具挑战性。 因此，有很少的多链项目已经在运行，但大多数提议的项目都是多链。 我们相信未来属于多链项目。

2.8.3. 创建和验证块：工作量证明与权益证明。 另一个重要的区别是用于创建和传播新块的算法和协议，检查它们的有效性，如果它们出现，则选择几个分叉中的一个。

两种最常见的范例是工作量证明 (PoW) 和权益证明 (PoS) 。工作量证明方法通常允许任何节点创建（《挖掘》）新块（并获得与挖掘块相关的一些奖励），如果它足够幸运地解决了无用的计算问题（通常涉及计算大量的哈希值），在其他竞争者设法做到这一点之前。在有分叉的情况下（例如，如果两个节点发布两个其他有效但不同的块来跟随前一个），则最长的分叉获胜。通过这种方式，区块链不可变性的保证是基于生成区块链所花费的工作量（计算资源）：任何想要创建此区块链分支的人都需要重新做这项工作来创建已提交块的替代版本。为此，需要控制创建新块所花费的总计算能力的50%以上，否则备用分叉将成为最长的成功率。

权益证明方法基于由一些特殊节点（验证者）付出的大量权益（由加密货币提名）来断言他们已经检查（验证）了一些块并且发现它们是正确的。验证者签署块，并为此获得一些小奖励;但是，如果一个验证者被发现签署了一个不正确的区块，并且提供了证据，那么其部分或全部权益将被没收。通过这种方式，区块链的有效性和不变性的保证由验证者对区块链有效性的总权益量给出。

在激励验证者（代替工作量证明的矿工）执行有用计算（需要检查或创建新块，特别是通过执行块中列出的所有交易）的方面，权益证明方法更为自然计算否则无用的哈希。通过这种方式，验证者将购买更适合于处理用戶交易的硬件，以便接收与这些交

[27] 暂时更差不多是十五。 但是，正在计划进行一些升级，以使以太坊交易吞吐量增加数倍。

易相关的奖励，从整个系统的角度来看，这似乎是非常有用的投资。

然而，权益证明系统在实施上更具挑战性，因为必须提供许多罕见但可能的条件。 例如，一些恶意验证者可能合谋破坏系统以提取一些利润（例如，通过改变它们自己的加密货币余额）。 这导致了一些非平凡的游戏理论问题。

简而言之，权益证明更自然，更有前途，特别是对于多链项目（因为如果有很多区块链，工作量证明需要大量的计算资源），但必须更仔细地考虑和实施。 目前大多数运行区块链项目，特别是最老的项目（如比特币和至少原始的以太坊），使用工作量证明。

2.8.4. 权益证明的变种。 委任权益证明 (DPoS) 与拜占庭容错 (BFT) 。 虽然工作量证明算法彼此非常相似，并且主要区别在于必须为挖掘新块而计算的哈希函数，但是权益证明算法有更多可能性。 他们值得对自己进行细分。

基本上，必须回答以下关于证明证明算法的问题：

- 谁可以生成（《挖掘》）新块——任何完整节点，或者只是（比较）小验证者子集的成员？（大多数权益证明系统需要生成新块并由几个指定验证者之一签名。）

- 验证者是否通过其签名保证块的有效性，或者是所有完整节点都被要求自己验证所有块？（可扩展的权益证明系统必须依赖验证者签名，而不是要求所有节点验证所有区块 链的所有块。）

- 是否有预先知道的下一个区块链块的指定生产者，以至于没有其他生产者可以生产该区块?

- 新创建的块最初只由一个验证者（其生产者）签名，还是必须从一开始就收集大多数验证者签名?

虽然根据这些问题的答案，似乎有24 种可能的权益证明算法类别，但实际上的区别归结为两种主要的权益证明方法。 实际上，设计用于可扩展多链系统的大多数现代权益证明算法以相同的方式回答前两个问题：只有验证者才能生成新块，并且它们保证块有效性，而不需要所有完整节点（只凭自己）检查所有块的有效性。

至于最后两个问题，他们的答案结果高度相关，基本上只留下两个基本选项：

- 委任权益证明(*DPoS*)：每个块都有一个众所周知的指定生产者; 没有其他生产

者可 以产生那块; 新块最初只由其生成验证者签名。

- 拜占庭容错(*BFT*)权益证明算法：有一个已知的验证者子集，其中任何一个都可以建议一个新的块; 在被释放到其他节点之前必须由大多数验证者验证和签名的几个建议候选者中的实际下一个块的选择是通过拜占庭容错共识协议的版本来实现的。

2.8.5. 委任权益证明 (DPOS) 和拜占庭容错 (BFT) 权益证明的比较。 拜占庭容错方法的优点在于，新生产的块从一开始就具有大多数验证者的签名，证明其有效性。 另一个优点是，如果大多数验证者正确执行拜占庭容错共识协议，则根本不会出现任何分叉。 另一方面，拜占庭容错算法往往非常复杂，需要更多时间让验证者子集达成共识。 因此，不能经常生成块。 这就是为什么我们期望电报开放网络(TON)区块链（从这种分类的角度来看是一个拜占庭容错项目）每五秒只产生一次块。 在实践中，这个间隔可能会减少到两到三秒（尽管我们不保证这一点），但不会更少，如果验证者遍布全球。

委任权益证明算法具有非常简单和直接的优点。 它可以经常产生新的块——比方说，每两秒一次，或者甚至每秒一次, [28] 因为它依赖于事先已知的指定块生成者。

但是，委任权益证明要求所有节点——或至少所有验证者——验证收到的所有块，因为生成和签署新块的验证者不仅确认了该块的相对有效性，而且还确认了它所引用的前一个块的有效性，并且链中所有以前的块（可能一直到当前验证者子集的责任期的开始）。在当前验证者子集上存在预定顺序，因此对于每个块，存在指定的生成者（即，期望生成该块的验证者）;这些指定的生产者以循环方式轮换。通过这种方式，块首先仅由其生成验证者签名;然后，当下一个块被开采时，它的生产者选择引用这个块而不是它的前任之一（否则它的块将位于较短的链中，这可能会在将来失去《最长的分叉》竞争），下一个块的签名本质上也是前一个块的附加签名。通过这种方式，新块逐渐收集更多验证者的签名——例如，在生成接下来的二十个块所需的时间内收集二十个签名。一个完整的节点要么需要等待这二十个签名，要么自己验证块，从一个充分确认的块（比如二十个块以前）开始，这可能不是那么容易。

委任权益证明算法的明显缺点是，新块（以及提交到其中的交易）仅在挖掘了二十多个块之后才能实现相同的信任程度（ **2.6.28** 中讨论的《递归可靠性》），而比较拜占庭容错算法: 拜占庭容错立即提供这种信任程度（比如二十个签名）。 另一个缺点是委任权益证明使用《最长叉赢》方法切换到其他分叉; 如果至少一些生产者在我们感兴趣的生产者之后未能生成后续的块（或者由于网络分区或复杂的攻击而未能观察到这些块），这使得分叉很可能发生。

我们认为拜占庭容错方法虽然比委任权益证明更复杂，并且需要更长的时间间

[28] 有些人甚至声称委任权益证明的块生成时间为半秒，这似乎并不现实，如果验证者分散在好几个洲。

隔，但更适合《紧密耦合》（参见 **2.8.14** ）多链系统，因为其他区块链几乎可以立即开始运转，在新的块中查看已提交的交易以后（例如，生成针对它们的消息），而不等待二十次有效性确认（即，接下来的二十个块），或者等待接下来的六个块以确保没有叉出现并且自行验证新块（在可扩展的多链系统中验证其他区块链的块可能代价过高）。因此，它们可以实现可扩展性，同时保持高度可靠性和可用性（参见 **2.8.12** ）。

另一方面，委任权益证明可能是《松散耦合》多链系统的一个很好的选择，其中不需要区块链之间的快速交互——例如，如果每个区块链（《工作链》）代表一个单独的分布式交换，并且区块链之间交互仅限于罕见的代币从一个工作链转移到另一个工作链（或者更确切地说，以一个接近一比一的速率交易：一个工作链中的一个山寨币交换另一个山寨币）。这是在比特股项目中实际完成的，它非常成功地使用了委任权益证明。

总而言之，虽然委任权益证明可以生成新块并将交易更快地包含在它们中（块之间的间隔更小），这些交易达到了在其他区块链和脱链应用程序中使用它们为《被提交》和《不可变》所需的信任级别，比拜占庭容错系统慢得多——比如说，在三十秒内[29] 而不是五秒。更快的交易包含并不意味着更快的交易提交。如果需要快速的区块链间交互，这可能会成为一个巨大的问题。在这种情况下，必须放弃委任权益证明 (DPoS) 并选择拜占庭容错权益证明 (BFT PoS)。

2.8.6. 支持交易中的图灵完备代码，即基本上任意的智能合约。区块链项目通常在其区块中收集一些交易，这以一种被认为有用的方式改变区块链状态（例如，将一些加密货币从一个账户转移到另一个账户）。一些区块链项目可能只允许某些特定的预定义类型的交易（例如，从一个账户到另一个账户的价值转移，只要提供了正确的签名）。其他区块链项目可能会在交易中支持某种有限形式的脚本。最后，一些区块链支持在交易中执行任意复杂的代码，使系统（至少在原理上）能够支持任意应用程序，只要系统的性能允许。这通常与《图灵完备的虚拟机和脚本语言》相关联（意味着任何可以用任何其他计算语言编写的程序都可以重写以在区块链内执行）和《智能合约》（这是驻留在区块链中的程序）。

当然，对任意智能合约的支持使系统真正灵活。 另一方面，这种灵活性需要付出代价：这些智能合约的代码必须在某个虚拟机上执行，并且当有人想要创建或验证块时，必须每次为块中的每个交易执行此操作。 与预定义和不可变的简单交易类型集相比，这会降低系统性能，可以通过在诸如C++（而不是某些虚拟机）之类的语言中实现它们的处理来优化它们。

最终，在任何通用区块链项目中，对图灵完备智能合约的支持似乎都是可取的;

[29] 例如，EOS是至今为止提出的最好的委任权益证明项目之一，承诺四十五秒确认和区块链间交互延迟（参见[8]，“交易确认”和“链间通信延迟”部分）

否则，区块链项目的设计者必须事先决定他们的区块链将用于哪些应用程序。 事实上，比特币区块链中缺乏对智能合约的支持是为什么必须创建新的区块链项目以太坊的主要原因。

在（异构;参见 **2.8.8** ）多链系统中，通过在一些区块链（即工作链）中支持图灵完备智能合约，以及其他区块链中一小组预定义的高度优化的交易，可以拥有《两全其美》的结果。

2.8.7. 多链系统的分类。 到目前为止，该分类对于单链和多链系统都是有效的。 但是，多链系统允许多个分类标准，反映了系统中不同区块链之间的关系。 我们现在讨论这些标准。

2.8.8. 区块链类型：同构和异构系统。 在多链系统中，所有区块链可以基本上是相同类型并且具有相同的规则（即，使用相同格式的交易，用于执行智能合约代码的相同虚拟机，共享相同的加密货币等等）），这种相似性被明确利用，但每个区块链中的数据不同。 在这种情况下，我们说系统是同构的。 否则，不同的区块链（在这种情况下通常称为工作链）可以具有不同的《规则》。 那么我们就说系统是异构的。

2.8.9. 混合异构同构系统。 有时我们有一个混合系统，其中有几组类型或规则用于区块链，但是存在许多具有相同规则的区块链，并且这个事实被明确地利用了。 它是一个混合的异构同构系统。 据我们所知，电报开放网络(TON)区块链是这种系统的唯一例子。

2.8.10. 具有若干工作链的异构系统具有相同的规则或联盟。 在一些情况下，具有相同规则的若干区块链（工作链）可以存在于异构系统中，但是它们之间的交互与具有不同规则的区块链之间的相互作用相同（即，它们的相似性未被明确利用）。 即使他们似乎使用《相同》的加密货币，他们实际上使用不同的《山寨币》（加密货币的独立化身）。有时人们甚至可以使用某些机制将这些山寨币以接近一比一的速度转换。但是，这并不能使我们认为系统是同构的; 它仍然是异构的。 我们说这种具有相同规则的异构工作链集合是联盟。

虽然制作允许人们使用相同规则（即联盟）创建多个工作链的异构系统看起来似乎是构建可扩展系统的廉价方式，但这种方法也存在许多缺点。 从本质上讲，如果有人在许多具有相同规则的工作链中主办大型项目，她就不会获得大型项目，而是获得该项目的许多小实例。 这就像拥有一个聊天应用程序（或游戏），允许在任何聊天（或游戏）房间中最多有五十个成员，但通过创建新房间来《扩展》以在必要时容纳更多用戶。 因此，很多用戶可以参与聊天或游戏，但我们可

以说这样的系统真正可扩展吗?

2.8.11. 主链的存在，外部或内部。有时，多链项目有一个独特的《主链》（有时称为《控制区块链》），例如，它用于存储系统的整体配置（所有活动区块链的集合，或者更确切地说是工作链），当前的验证者集（用于权益证明制度），等等。有时其他区块链被《束缚》到主链上，例如通过将其最新块的哈希值提交到其中（这也是电报开放网络(TON)区块链所做的事情）。

在某些情况下，主链是外部的，这意味着它不是项目的一部分，而是一些其他预先存在的区块链，最初与新项目的使用完全无关，并且不知道它。例如，可以尝试使用以太坊区块链作为外部项目的主链，并为此目的将特殊智能合约发布到以太坊区块链中（例如，用于选择和惩罚验证者）。

2.8.12. 分片支持。 一些区块链项目（或系统）本身支持分片，这意味着几个（必然是同构的;参见 **2.8.8** ）区块链被认为是单个（从高层次的角度来看）虚拟区块链的分片。 例如，可以创建具有相同规则的256个分片区块链（《分片链》），并根据其 账户_*id* 的第一个字节将帐户的状态保持在一个选定的分片中。

分片是扩展区块链系统的一种自然方法，因为如果它被正确实现，系统中的用户和智能合约根本不需要知道分片链的存在。 实际上，当负载变得过高时，人们通常希望为现有的单链项目（例如以太坊）添加分片。另一种扩展方法是使用 **2.8.10** 中描述的异构工作链的《联盟》，允许每个用户将她的帐户保存在她选择的一个或多个工作链中，并必要时将资金从她的帐户从一个工作链转移到另一个工作链中，基本上执行一比一的山寨币交换操作。这种方法的缺点已在 **2.8.10** 中讨论过。

但是，分片不是那么容易以快速可靠的方式实现的，因为它意味着不同分片链之间的大量消息。例如，如果账户在 N 个分片之间均匀分配，并且唯一的交易是从一个账户到另一个账户的简单资金转账，那么在单个区块链中只会执行所有交易的一小部分 $(1 / N)$;几乎所有 $(1 - 1 / N)$ 交易都涉及两个区块链，需要区块链间通信。如果我们希望这些交易快速，我们需要一个快速系统来在分片链之间传输消息。换句话说，区块链项目需要在 **2.8.14** 中描述的意义上《紧密耦合》。

2.8.13. 动态和静态分片。 分片可能是动态的（如果在必要时自动创建其他分片）或静态分片（当有预定义数量的分片时，最多只能通过硬分叉进行更改）。 大多数分片提义都是静态的; 电报开放网络(TON)区块链使用动态分片（参见 **2.7** ）。

2.8.14. 区块链之间的相互作用：松散耦合和紧密耦合的系统。 可以根据组成区块链之间

支持的交互级别对多区块链项目进行分类。

最低级别的支持是不同区块链之间没有任何相互作用。 我们在这里不考虑这种情况，因为我们宁愿说这些区块链不是一个区块链系统的一部分，而只是相同区块链协议的单独实例。

下一级别的支持是对区块链之间的消息传递没有任何具体支持，原则上使交互成为可能，但是很尴尬。我们称这种系统为《松散耦合》;在他们中，必须发送消息并在区块链之间传递价值，好像它们是属于完全独立的区块链项目的区块链一样（例如，比特币和以太坊;想象两方想要把一些保存在比特币区块链中的比特币，交换成保存在以太网区块链中的以太币）。换句话说，必须在源区块链的块中包含出站消息（或其生成交易）。然后，她（或其他一方）必须等待足够的确认（例如，给定数量的后续块）以将始发的交易视为《已提交》和《不可变》，以便能够基于它的存在而执行外部操作。只有这样，才能提交将消息中继到目标区块链中的交易（可能伴随着对始发交易的存在的引用和默克尔树证明）。

如果在传输消息之前没有等待足够长的时间，或者由于其他原因无论如何都发生了分支，则两个区块链的连接状态变得不一致：从未在第一个区块链（的最终选择的分支） 生成过的消息被传递到第二个区块链中。

有时通过标准化消息的格式以及所有工作链的块中输入和输出消息队列的位置来添加对消息传递的部分支持（这在异构系统中尤其有用）。 虽然这在一定程度上促进了消息传递，但它在概念上与先前的情况没有太大不同，因此这种系统仍然算《松散耦合》。

相比之下，《紧密耦合》系统包括在所有区块链之间提供快速消息传递的特殊机制。 期望的行为是能够在原始区块链的块中生成消息之后立即将消息传递到另一个工作链。 另一方面，《紧密耦合》系统也可望在形成叉的情况下保持整体一致性。 虽然这两个要求一看似乎是矛盾的，但我们认为电报开放网络(TON)区块链使用的机制（将分片链块哈希包含在主链块中;使用《垂直》区块链来修复无效块，参见 **2.1.17** ;超立方体路由，参见 **2.4.19** ;即时超立方体路由，参见 **2.4.20** ）使其成为一个《紧密耦合》系统，也许是至今为止唯一的《紧密耦合》系统。

当然，构建《松散耦合》系统要简单得多; 但是，快速有效的分片（参见 **2.8.12** ）要求系统《紧密耦合》。

2.8.15. 简化分类。 几代区块链项目。 到目前为止，我们建议的分类将所有区块链项目拆分为大数量的分类。 但是，我们使用的分类标准在实践中恰好相关。 通过一些例子，这使我们能够建议区块链项目分类的简化《几代》方法，作为对现实的非常粗略的近似。尚未实施和部署的项目以斜体显示; 一代里最重要的特征以粗体显示。

- 第一代：单链，工作量证明，不支持智能合约。 示例：比特币 (2009) 和许多其他无趣的模仿者（莱特币，门罗币，...）。

- 第二代：单链，工作量证明，智能合约支持。 示例：以太坊 (2013年; 2015年部署），至少以其原始形式。

- 第三代：单链，权益证明，智能合约支持。 示例：未来以太坊 (2018年或以后）。

- 替代的第三代 (3') 代：多链，权益证明，不支持智能合约，松散耦合。 示例：比特股（2013-2014;使用委任权益证明）。

- 第四代：多链，权益证明，智能合约支持，松散耦合。 示例：EOS（2017;使用委任权益证明），波卡（2016;使用拜占庭容错）。

- 第五代：多链，权益证明与拜占庭容错，智能合约支持，紧密耦合，分片。 示例：电报开放网络(TON) (2017)。

虽然并非所有区块链项目都属于这些类别中的一个，但大多数都属于这些类别。

2.8.16. 改变区块链项目《基因组》的复杂性。 上述分类定义了区块链项目的《基因组》。 这个基因组非常《僵化》：一旦项目部署并且被许多人使用，几乎不可能改变它。 人们需要一系列硬分叉（这需要大多数社区的批准），即使这样，为了保持向后兼容性，改变也需要非常保守（例如，改变虚拟机的语义可能打破现有的智能合约）。 另一种方法是使用不同的规则创建新的《侧链》，并以某种方式将它们绑定到始发项目的区块链（或区块链）。 可以使用现有单区块链项目的区块链作为外部主链，用于一个新的独立项目。[30]

我们的结论是，项目的基因组一旦部署就很难改变。 即使从工作量证明开始并计划在未来用权益证明替换它也是相当复杂的。[31] 将分片添加到最初设计的项目而不支持

[30] 例如，等离子项目计划使用以太坊区块链作为其（外部）主链; 否则它与以太坊没有多大的互动，它有可能是由与以太坊项目无关的团队建议和实施的。

[31] 截至2017年，以太坊仍在努力从工作量证明过渡到工作量证明 + 权益证明组合系统; 我们希望它有一天会成为一个真正的权益证明系统。

它们似乎几乎是不可能的。[32] 事实上，将智能合约的支持添加到项目中（即比特币） 最初设计不支持这些功能的设计被认为是不可能的（或者至少是大多数比特币社区不受欢迎的），最终导致创建了一个新的区块链项目，以太坊。

2.8.17. TON区块链的基因组。 因此，如果想要建立一个可扩展的区块链系统，就必须从一开始就仔细选择其基因组。 如果系统的目的是要支持将来在部署时未知的某些其他特定功能，那么它应该从一开始就支持《异构》工作链（具有可能不同的规则）。 为了使系统真正可扩展，它必须从一开始就支持分片; 只有当系统《紧密耦合》时（参见 **2.8.14** ），分片才有意义，因此这反过来意味着主链的存在，快速的区块链间消息系统，拜占庭容错权益证明的使用等等。

当考虑到所有这些影响时，为电报开放网络(TON)区块链项目做出的大部分设计选择都是自然的，而且几乎是唯一可能的选择。

2.9 与其他区块链项目的比较

我们通过试着在包含现有和建议的区块链项目的地图上找到电报开放网络(TON)区块链的位置来结束我们对TON区块链及其最重要和独特特征的简要讨论。我们使用 **2.8** 中描述的分类标准以统一的方式讨论不同的区块链项目，并构建这样的《区块链项目图》。我们将此地图表示为表 1，然后分别简要讨论几个项目，以指出它们可能不适合一般方案的特性。

2.9.1. 比特币[19]; https://bitcoin.org/. 比特币 (2009) 是第一个也是最着名的区块链项目。这是一个典型的第一代区块链项目：它是单链的，它使用工作量证明和《最长叉胜》分叉选择算法，并且它没有图灵完备的脚本语言（但是，支持没有循环的简单脚本）。比特币区块链没有帐户的概念;它使用UTXO（未使用的交易输出）模型。

2.9.2. 以太坊[5]; https://ethereum.org/. 以太坊 (2015) 是第一个支持图灵完备智能合约的区块链。 因此，

2.9.3. 未来币; https://nxtplatform.org/. 未来币 (2014) 是第一个基于权益证明的区块链和货币。 它仍然是单链的，没有智能合约支持。

[32] 以太坊的建议可以追溯到2015年; 目前尚不清楚如何在不破坏以太坊或创建基本独立的并行项目的情况下实施和部署它们。

项目	年	代	共识算法	支持任意代码	单链/多链系统	异构/同构多链系统	支持分片	区块链之间的相互作用
比特币	2009	1	工作量证明	不	单			
以太坊	2013, 2015	2	工作量证明	是	单			
未来币	2014	2+	权益证明	不	单			
Tezos	2017, ?	2+	权益证明	是	单			
Casper	2015, (2017)	3	工作量证明/权益证明	是	单			
比特股	2013, 2014	3'	委任权益证明	不	多	异	无	松散耦合
EOS	2016, (2018)	4	委任权益证明	是	多	异	无	松散耦合
波卡	2016, (2019)	4	权益证明拜占庭容错	是	多	异	无	松散耦合
宇宙币	2017, ?	4	权益证明拜占庭容错	是	多	异	无	松散耦合
电报开放网络 (TON)	2017, (2018)	5	权益证明拜占庭容错	是	多	混合	动态	松散耦合

表1：一些值得注意的区块链项目的总结。 表格的纵列是：项目 - 项目名称; 年 - 宣布的年和发布的年; G. - 代（参见 **2.8.15** ）; Cons. - 共识算法（参见 **2.8.3** 和 **2.8.4** ）; Sm. - 支持任意代码（智能合约; 参见 **2.8.6** ）; Ch. - 单链/多链系统（参见 **2.8.2** ）; R. - 异构/同构多链系统（参见 **2.8.8** ）; Sh. - 支持分片（参见 **2.8.12** ）; Int. - 区块链之间的相互作用，松散或紧密（参见 **2.8.14** ）。

2.9.4. Tezos; https://www.tezos.com/. *Tezos* (2018或更高版本) 是一个基于PoS 的单区块链项目。我们在这里提到它是因为它的独特功能：它的块解释函数*ev_block*（参见 **2.2.6** ）不是固定的，而是由OCaml模块决定的，可以通过将新版本提交到区块链中来升级（并收集一些）对提议的变更投票）。通过这种方式，人们将能够通过首先部署《香草》*Tezos*区块链，然后逐步改变所需方向的块解释功能来创建自定义单链项目，而无需任何硬分叉。

这个想法虽然很有趣，却有明显的缺点，它禁止在其他语言（如C++）中进行任何优化实现，因此基于*Tezos*的区块链注定会降低性能。我们认为通过发布所提出的块解释函数*ev_trans*的正式规范可能已经获得了类似的结果，而没有确定特定的实现。

2.9.5. Casper。 [33] Casper 是即将推出的以太坊权益证明算法; 它在2017年（或2018年）的逐步部署，如果成功，将把以太坊变成单链权益证明或混合工作量证明 + 权益证明系统，并提供智能合约支持，将以太坊转变为第三代项目。

2.9.6. 比特股 [14];https://bitshares.org. 比特股 (2014) 是一个基于分布式区块链的交换平台。 它是一个没有智能合约的异构多区块链委任权益证明系统; 假设区块链状态适合内存，它只允许一小部分预定义的专用交易类型实现其高性能，这些类型可以在C++中有

[33] https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/

效实现。 它也是第一个使用委任权益证明 (DPoS) 的区块链项目，至少为某些特殊目的证明了它的可行性。

2.9.7. EOS [8]; https://eos.io. EOS (2018或更高版本) 是一种提议的异构多区块链委任权益证明系统，具有智能合约支持和对消息传递的一些最小支持（在 **2.8.14** 中描述的意义上仍然是松散耦合的）。这是以前成功创建比特股和SteemIt项目的同一团队的尝试，展示了委任权益证明共识算法的优点。可扩展性将通过为需要它的项目创建专用工作链来实现（例如，分布式交换可能使用支持一组特殊优化交易的工作链，类似于比特股所做的那样），并通过创建具有相同规则的多个工作链（在 **2.8.10** 中描述的联盟）。这种可扩展性方法的缺点和局限性已在这些地方讨论过：参看 **2.8.5** , **2.8.12** 和 **2.8.14** ，为了更详细地参考委任权益证明，分片，工作链之间的交互及其对区块链系统可扩展性的影响。

同时，即使一个人无法通过EOS或其他方式《在区块链内创建面书》（参见 **2.9.13** ），我们认为EOS可能成为一个方便的平台，用于一些高度专业化的弱互动分布式应用程序，类似于比特股（分散交换）和SteemIt（分散式博客平台）。

2.9.8. 波卡 [24]; https://polkadot.io/ . 波卡 (2019年或更高版本) 是最精心设计和最详细的多线权益证明项目之一; 它的发展由以太坊联合创始人之一领导。 该项目是我们地图上距离电报开放网络(TON)区块链最近的项目之一。 （事实上，我们的术语《渔民》和《提名者》来自波卡项目。）

波卡是一个异构的松散耦合多链权益证明项目，具有拜占庭容错 (BFT) 共识来生成新块和主链（可能是外部的——例如，以太坊区块链）。 它还使用超立方体路由，有点像（**2.4.19** 中所述）电报开放网络(TON)的慢速版本。

它的独特之处在于它不仅可以创建公共区块链，还可以创建私人区块链。 这些私人区块链也可以与其他公共区块链进行交互，不管是不是波卡。

向对方转移资金，或者用于大型公司可能对于私人区块链技术的任何其他用途。

但是，波卡没有分片支持，也没有紧密耦合。 这有点妨碍了它的可扩展性，它的可扩展性与EOS类似。 （也许好一点，因为波卡使用拜占庭容错权益证明而不是委任权益证明。）

2.9.9. Universa; https://universa.io. 我们在这里提到这个不寻常的区块链项目的唯一原因是因为它是迄今为止唯一一个通过明确引用类似于我们的无限分片范式的项目（参见 **2.1.2** ）。 它的另一个特点是它绕过了与拜占庭容错相关的所有复杂性，承诺只有项目的可信和许可合作伙伴才会被认可为验证者，因此他们永远不会提交无效的块。 这是一个

有趣的决定; 然而，它本质上使区块链项目有意集中，区块链项目通常要避免（在受信任的集中式环境中工作为什么需要区块链?）。

2.9.10. 等离子; https://plasma.io. 等离子 (2019?) 是另一位以太坊联合创始人的非常规区块链项目。它应该在不引入分片的情况下减轻以太坊的某些限制。从本质上讲，它是与以太坊独立的项目，引入了（异构）工作链的层次结构，绑定到顶层的以太坊区块链（用作外部主链）。资金可以从层次结构中的任何区块链转移（从作为根的以太坊区块链开始），以及要完成的工作的描述也被转移。然后在子工作链中完成必要的计算（可能需要将更多始发工作的部分转发到树下），将结果传递出去，并收集奖励。实现一致性和验证这些工作链的问题被（支付渠道启发）机制所规避，允许用戶单方面将资金从行为不端的工作链中提取到其父工作链（尽管速度缓慢）

通过这种方式，等离子可能成为绑定到以太坊区块链的分布式计算的平台，类似于《数学协处理器》。但是，这似乎不是实现真正的通用可伸缩性的方法。

2.9.11. 专门区块链项目。还有一些专门的区块链项目，例如FileCoin（一个激励用戶提供磁盘空间来存储愿意为其付费的其他用戶的文件的系统），Golem（一种基于区块链的平台，用于为三维渲染等专业应用租借和提供计算能力）或SONM（另一个类似的计算能力借贷项目）的功能。这些项目在区块链组织层面上没有引入任何新概念;相反，它们是特定的区块链应用程序，可以通过在通用区块链中运行的智能合同来实现，前提是它可以提供所需的性能。因此，此类项目可能会使用现有或计划中的区块链项目之一作为其基础，例如EOS，波卡或电报开放网络(TON)。如果一个项目需要《真正的》可扩展性（基于分片），那么最好使用TON;如果通过定义自己的一系列工作链（明确针对其目的进行优化）来满足在《联盟》环境中工作的内容，则可以选择EOS或波卡。

2.9.12. TON区块链。 TON（电报开放网络）区块链（计划于2018年）是我们在本文档中描述的项目。 它被设计成第一个第五代区块链项目 - 即拜占庭容错权益证明-多链项目，混合同构/异构，支持（可分片）自定义工作链，具有本机分片支持，并且紧密耦合（特别是能够几乎立即在分片之间转发消息，同时保持所有分片链的一致状态）。 因此，它将是一个真正可扩展的通用区块链项目，能够容纳基本上可以在区块链中实现的任何应用程序。 当通过电报开放网络(TON)项目的其他组件（参见 **1**）进行扩充时，其可能性进一步扩大。

2.9.13. 是否可以《将面书上传到区块链》？ 有时人们声称可以实现面书的规模上的社交网络作为驻留在区块链中的分布式应用程序。 通常，最喜欢的区块链项目被引用为此

类应用程序的可能《主机》。

我们不能说这是技术上的不可能。 当然，需要一个具有真正分片（即电报开放网络(TON)）的紧密耦合的区块链项目，以便这样一个大型应用程序不会工作得太慢（例如，将驻留在一个分片链中的用户的消息和更新传递给居住在另一个分片链中的朋友，并且有合理的延迟）。 但是，我们认为这不是必需的，也永远不会完成，因为价格过高。

让我们考虑《将面书上传到区块链》作为思想实验;任何其他类似规模的项目也可以作为一个例子。一旦将面书上传到区块链中，目前由面书服务器完成的所有操作将被序列化为某些区块链中的交易（例如，电报开放网络(TON)的分片链），并且将由这些区块链的所有验证者执行。如果我们希望每个块收集至少二十个验证者签名（立即或最终，如在委任权益证明系统中），则必须执行每个操作至少二十次。类似地，面书的服务器在其磁盘上保存的所有数据将保存在相应分片链的所有验证者的磁盘上（即，至少二十个副本）。

因为验证者本质上是面书当前使用的服务器（或者可能是服务器集群，但这不影响此参数的有效性），我们看到在区块链里运行面书相关的总硬件开支至少是比以传统面书实施的情况高出二十倍。

事实上，开支仍然会高得多，因为区块链的虚拟机比运行优化的编译代码的《单独的中央处理器》慢，并且其存储未针对特定于面书的问题进行优化。 人们可以通过制作适合面书的一些特殊交易的特定工作链来部分缓解这个问题; 这是比特股和 EOS实现高性能的方法，也属于电报开放网络(TON)区块链。 然而，一般的区块链设计本身仍然会施加一些额外的限制，例如必须将所有操作注册为块中的交易，在默克尔树中组织这些交易，计算和检查它们的默克尔哈希值，以进一步传播此块，等等。

因此，一个保守的估计是，为了验证托管该规模的社交网络的区块链项目，人们需要一百倍于与面书现在使用的服务器相同性能的服务器。有人必须为这些服务器付费，无论是拥有分布式应用程序的公司（想象在每个面书页面上看到七百个广告而不是七个）或其用戶。无论哪种方式，这似乎在经济上都不可行。

我们认为，不是所有内容都应该上传到区块链中。例如，没有必要将用戶照片保留在区块链中;在区块链中注册这些照片的哈希值并将照片保存在分布式的脱链存储器（例如FileCoin或电报开放网络(TON)存储）中将是一个更好的主意。这就是为什么 TON不仅仅是一个区块链项目，而是以 TON区块链为中心的几个组件（ TON P2P网络， TON存储， TON服务）的集合，如第 **1** 章和第 **4** 章所述。

3 电报开放网络(TON)

任何区块链项目不仅需要块格式和区块链验证规则的规范，还需要用于传播新块，发送和收集交易候选者(等等)的网络协议。 换句话说，每个区块链项目都必须建立专门的对等网络。 该网络必须是点对点的，因为区块链项目通常被期待为分散的，因此不能依赖于集中的服务器组并使用传统的客戶端与服务器架构，例如，传统的在线银行应用程序。即使轻型客戶端（例如，轻型加密钱包智能手机应用程序）必须以类似客戶端与服务器的方式连接到完整节点，如果其先前的对等体出现故障，实际上可以自由连接到另一个完整节点，只要用于连接到完整节点的协议足够标准化。

虽然比特币或以太坊等单区块链项目的网络需求可以很容易地满足（基本上需要构建一个《随机》的点对点覆盖网络，并通过流言协议传播所有新的块和交易候选者），多区块链项目，如电报开放网络(TON)区块链，要求更高（例如，必须能够只跟从一些分片链的更新，不一定能跟从所有的）。 因此，TON区块链和TON项目的网络部分作为一个整体值得进行简短的讨论。

另一方面，一旦支持TON区块链所需的更复杂的网络协议到位，事实证明它们可以很容易地用于不一定与TON区块链的直接需求相关的目的，从而提供更多的可能性和灵活性 用于在TON生态系统中创建新服务。

3.1 抽象数据报网络层

构建电报开放网络(TON)协议的基础是 (TON) 抽象（数据报）网络层。 它使所有节点能够假定某些《网络标识》，由256位《抽象网络地址》表示，并且仅使用这些256位网络地址进行通信（作为第一步发送数据报）以识别发送者和接收者。 特别是，不需要担心IPv4或IPv6地址，UDP端口号等; 它们被抽象网络层隐藏。

3.1.1. 抽象网络地址。 抽象网络地址或抽象地址，或简称地址，是一个256位整数，基本上等于256位ECC公钥。 该公钥可以任意生成，从而创建节点要多少有多少的不同网络标识。 但是，必须知道相应的私钥才能接收（和解密）用于这种地址的消息。

实际上，地址本身并不是公钥; 相反，它是序列化TL对象（参见 **2.2.5** ）的256位哈希(哈希 = SHA256），它可以根据其构造函数（前四个字节）描述几种类型的公钥和地址。 在最简单的情况下，这个序列化的TL对象只包含一个4字节的幻数和一个256位的椭圆曲线加密 (ECC) 公钥; 在这种情况下，地址将等于这个36字节结构的哈希。 但是，可以使用2048位RSA密钥或任何其他公钥加密方案。

当节点得知另一个节点的抽象地址时，它还必须接收其《原像》（即序列化的TL对象，其哈希等于该抽象地址），否则它将无法加密并将数据报发送到该地址。

3.1.2. 较低级别的网络。 UDP实现。 从几乎所有电报开放网络(TON)组件的角度来看，唯一存在的是一个网络（抽象数据报网络层）能够（不可靠地）将数据报从一个抽象地址发送到另一个抽象地址。 原则上，抽象数据报网络层 (ADNL) 可以在不同的现有网络技术上实现。 但是，我们将在IPv4 / IPv6网络（例如互联网或内部网）中通过UDP实现它，如果UDP不可用，则使用可选的TCP回退。

3.1.3. 最简单的实例，ADNL相比UDP。 从发送方的抽象地址向任何其他抽象地址（具有已知的原像）发送数据报的最简单情况可以如下实现。

假设发送方以某种方式知道拥有目的地抽象地址的接收方的IP地址和UDP端口，并且接收方和发送方都使用从256位ECC公钥派生的抽象地址。

在这种情况下，发送者只是通过其ECC签名（使用其私钥完成）及其源地址（或源地址的原像，如果尚不知道接收者知道该原像）来增加要发送的数据报。 结果使用接收者的公钥加密，嵌入到UDP数据报中并发送到接收者的已知IP和端口。 因为UDP数据报的前256位包含接收者的抽象地址，所以接收者可以识别应该使用哪个私钥来解密数据报的其余部分。 只有在那之后才会显示发送者的身份。

3.1.4. 安全性较低的方式，发送方的地址为明文。 当接收者和发送者的地址以明文形式保存在UDP数据报中时，有时安全性较低的方案就足够了; 使用ECDH（椭圆曲线 迪菲-赫尔曼）将发送者的私钥和接收者的公钥组合在一起，生成一个256位的共享密钥（之后被使用），以及未加密的部分中包含的随机256位随机数， 导出用于加密的AES密钥。 例如，可以通过在加密之前将原始明文数据的哈希连接到明文来提供完整性。

这种方法的优点是，如果预计在两个地址之间交换多个数据报，则只能计算一次共享密钥然后缓存; 然后，加密或解密下一个数据报将不再需要较慢的椭圆曲线操作。

3.1.5. 频道和频道标识符。 在最简单的情况下，携带嵌入式TON ADNL（电报开放网络抽象数据报网络层）数据报的UDP数据报的前256位将等于接收方的地址。 然而，通常它们构成频道标识符。 有不同类型的频道。 其中一些是点对点的; 它们是由希望在未来交换大量数据的双方创建的，并通过运行经典或椭圆曲线迪菲-赫尔曼（如果需要额外的安全）交换（如 **3.1.3** 或 **3.1.4** 所述）加密的几个数据包来生成共享秘密，或者只是由一方生成随机共享秘密并将其发送给另一方。

在此之后，从共享秘密中结合一些附加数据（例如发送者和接收者的地址），例

如，通过哈希，该标识符被用作UDP数据报的前256位，该数据报携带借助于该共享秘密加密的数据。

3.1.6. 频道作为隧道标识符。 通常，《频道》或《频道标识符》简单地选择处理接收机已知的入站UDP数据报的方式。 如果频道是接收者的抽象地址，则按照 **3.1.3** 或 **3.1.4** 的规定进行处理; 如果频道是 **3.1.5** 中讨论的已建立的点对点频道，则处理包括借助共享秘密解密数据报，参见所述。

特别地，当直接接收者简单地将所接收的消息转发给其他人 - 实际接收者或另一个代理时，频道标识符实际上可以选择《隧道》。 一些加密或解密步骤（让人联想到《洋葱路由》[11]甚至《大蒜路由》[34]）可能会在此过程中完成，而另一个频道标识符可能会用于重新加密的转发数据包 (例如，可以使用对等频道将分组转发到路径上的下一个接收者)。

通过这种方式，可以在电报开放网络(TON)抽象数据报网络层的级别上添加对《隧道》和《代理》的一些支持——与TOR或I^2P(隐形网计划)项目提供的相似——而不会影响所有更高级别的功能TON网络协议，这对这种添加会是不可知。 TON代理服务利用了这个机会（参见 **4.1.11** ）。

3.1.7. 零频道和引导问题。 通常，TON ADNL（电报开放网络抽象数据报网络层）节点将具有一些《邻居表》，其包含关于其他已知节点的信息，例如它们的抽象地址及其原像（即，公钥）及其IP地址和UDP端口。 然后，它将通过使用从这些已知节点获取的信息作为特殊查询的答案逐渐扩展此表，并且有时会修剪过时的记录。

但是，当TON ADNL（电报开放网络抽象数据报网络层）节点刚刚启动时，可能会发生它不知道任何其他节点，并且只能得知节点的IP地址和UDP端口，而不能得知它的抽象地址。 发生这种情况：例如，如果轻客戶端无法进接任何先前缓存的节点以及任何硬编码到软件中的节点，并且必须要求用戶输入节点的IP地址或网域名称系统的域， 通过网域名称系统解决。

在这种情况下，节点将数据包发送到相关节点的特殊《零频道》。 这不需要知道接收者的公钥（但信息仍应包含发送者的身份和签名），因此信息无需加密即可传输。它应该通常仅用于获取接收者的身份（可能是为此目的而创建的一次性身份），然后以更安全的方式开始通信。

一旦知道至少一个节点，就很容易通过更多条目填充《邻居表》和《路由表》，从发送到已知节点的特殊查询的答案中得知它们。

并非所有节点都需要处理发送到零频道的数据报，但用于引导轻客戶端的那些节点应该支持此功能。

[34] https://geti2p.net/en/docs/how/garlic-routing

3.1.8. ADNL(抽象数据报网络层)上类似TCP的流协议。 ADNL是基于256位抽象地址的不可靠（小尺寸）数据报协议，可用作更复杂网络协议的基础。例如，可以使用ADNL作为IP的抽象替代来构建类似TCP的流动协议。但是，电报开放网络(TON)项目的大多数组件都不需要这样的流动协议。

3.1.9. ADNL上的RLDP(可靠大数据报协议)。 使用基于ADNL的可靠的任意大小的数据报协议（称为RLDP）代替类似TCP的协议。例如，可以使用这种可靠的数据报协议将远程过程调用查询发送到远程主机并从它们接收答案（参见 **4.1.5** ）。

3.2 电报开放网络(TON)分布式哈希表：类似Kademlia的分布式哈希表

电报开放网络(TON)分布式哈希表(DHT)在TON项目的网络部分中起着至关重要的作用，用于定位网络中的其他节点。例如，想要将交易提交到分片链的客户端可能希望找到该分片链的验证者或对照者，或者至少某个可能将客户端的交易中继到对照者的节点。这可以通过在电报开放网络分布式哈希表中查找特殊密钥来完成。电报开放网络分布式哈希表的另一个重要应用是它可以用来快速填充新节点的邻居表（参见 **3.1.7** ），只需查找随机密钥或新节点的地址即可。如果节点对其入站数据报使用代理和隧道，则它在电报开放网络分布式哈希表中发布隧道标识符及其入口点（例如，IP地址和UDP端口）；那么希望将数据报发送到该节点的所有节点将首先从分布式哈希表获得该联系信息。

电报开放网络分布式哈希表是类似*Kademlia*的分布式哈希表系列的成员[16]。

3.2.1. 电报开放网络分布式哈希表的密钥。 电报开放网络分布式哈希表的密钥只是256位整数。在大多数情况下，它们被计算为TL序列化对象的SHA256（参见 **2.2.5** ），称为密钥的原像或密钥描述。在某些情况下，电报开放网络(TON)节点的抽象地址（参见 **3.1.1** ）也可以用作电报开放网络分布式哈希表的密钥，因为它们也是256位的，并且它们也是TL序列化对象的哈希。例如，如果一个节点不害怕发布其IP地址，任何知道其抽象地址的人都可以通过将该地址作为分布式哈希表中的密钥而查找来找到它。

3.2.2. 分布式哈希表的值。 分配给这些256位密钥的值基本上是有限长度的任意字节串。这种字节串的理解由相应密钥的原像决定;它通常由查找密钥的节点和存储密钥的节点被得知。

3.2.3. 分布式哈希表的节点。 半永久性网络标识。电报开放网络分布式哈希表的钥值映射保留在分布式哈希表的节点上——基本上是电报开放网络(TON)的所有成员。为此，除了

3.1.1 中描述的任何数量的短暂和永久抽象地址之外，TON网络的任何节点（可能除了一些非常轻的节点）至少具有一个《半永久地址》，其中将其识别为电报开放网络分布式哈希表的成员。这个半永久性或分布式哈希表地址不应该经常更改，否则其他节点将无法找到他们正在寻找的密钥。如果节点不想显示其《真实》标识，则它会生成一个单独的抽象地址，仅用于参与分布式哈希表。但是，此抽象地址必须是公共的，因为它将与节点的IP地址和端口相关联。

3.2.4. Kademlia距离。 现在我们有256位密钥和256位（半永久）节点地址。我们在256位序列的集合上引入所谓的*XOR*距离或*Kademlia*距离 d_k，由下式给出：

$$d_k(x, y) := (x \oplus y) \text{ 理解为无符号的256位整数} \tag{25}$$

这里 $x \oplus y$ 表示相同长度的两个比特序列的按位eXclusive OR (XOR)。

*Kademlia*距离在所有256位序列的 2^{256} 集上引入度量。特别是，当且仅当 $x = y$，$d_k(x, y) = d_k(y, x)$ 和 $d_k(y, z) \le d_k(x, y) + d_k(y, z)$ 时，我们才得到 $d_k(x, y) = 0$。另一个重要特性是在距*x*的任何给定距离处只有一个点：$(x : d_k(x, y) = d_k(x, y')$ 意味着 $y = y')$。

3.2.5. 类似Kademlia的分布式哈希表和电报开放网络分布式哈希表。 我们说具有256位密钥和256位节点地址的分布式哈希表 (DHT) 是类似*Kademlia*的分布式哈希表，如果期望将 s 个Kademlia最接近 K 的节点上的密钥K的值保持（即，从其地址到 K 的Kademlia距离最小的节点。）

这里 s 是一个小参数，比如 $s = 7$，需要提高分布式哈希表的可靠性（如果我们只将密钥保存在一个节点上，最接近 K 的那个节点，如果仅该节点脱机，则该钥的值将丢失）。

根据这个定义，电报开放网络分布式哈希表是类似Kademlia的分布式哈希表。它是通过 **3.1** 中描述的抽象数据报网络层协议实现的。

3.2.6. *Kademlia*路由表。 参与类似Kademlia的分布式哈希表的任何节点通常都维持一个Kademlia路由表。在电报开放网络分布式哈希表的情况下，它由 $n = 256$ 个桶组成，编号从 0 到 $n - 1$。第 i 个桶将包含关于一些已知节点的信息（固定数量 t 的《最佳》节点，可能还有一些额外的候选者）位于节点地址 a 的 2^i 到 $2^i + 1 - 1$ 的Kademlia距离。[35] 此信息包括其（半永久）地址，IP地址和UDP端口，以及一些可用性信息，例如上次ping的时间和延迟。

[35] 如果存储桶中有足够多的节点，则根据*Kademlia*距离的前四位，它可以进一步细分为八个子存储桶。这将加速分布式哈希表的查找。

当Kademlia节点因为某个查询的结果得知任何其他Kademlia节点时，它将其包含在其路由表的合适桶中，首先作为候选者。 然后，如果该桶中的一些《最佳》节点失败（例如，长时间不响应ping查询），则可以用一些候选者替换它们。 通过这种方式，Kademlia路由表保持填充状态。

Kademlia路由表中的新节点也包含在 **3.1.7** 中描述的抽象数据报网络层邻居表中。 如果经常使用来自*Kademlia*路由表的桶中的《最佳》节点，则可以建立 **3.1.5** 中描述的意义上的频道以便于加密数据报。

电报开放网络分布式哈希表的一个特殊功能是它试着选择具有最小往返延迟的节点作为Kademlia路由表的桶的《最佳》节点。

3.2.7. (Kademlia网络查询。) Kademlia节点通常支持以下网络查询：

- Ping——检查节点可用性。
- 存储（密钥，值）——要求节点将值保存为密钥密钥的值。 对于电报开放网络分布式 哈希表，存储查询稍微复杂一些（参见 **3.2.9** ）。
- FIND_NO（密钥，l）——要求节点将 l 个Kademlia最近的已知节点（从其Kademlia 路由表）返回到密钥。
- FIND_VALUE（密钥，l）——与上面相同，但如果节点知道与密钥密钥对应的值，则只返回该值。

当任何节点想要查找密钥 K 的值时，它首先创建 s' 个节点的 S 组（s' 为小值，比如 $s' = 5$），相对于Kademlia距离中最接近 K 的所有已知节点（即，它们来自*Kademlia*路由表）。然后向它们每个发送一个FIND_VALUE查询，并将其答案中提到的节点包含在 S 中。然后，如果之前没有完成，那么 S 中最接近 K 的 s' 节点也会收到一个FIND_VALUE查询，并且过程继续，直到找到值或 S 集停止增长。这是关于Kademlia距离最接近 K 的节点的一种《波束搜索》。

如果要设置某个密钥K的值，则对 $s' \geq s$ 运行相同的过程，使用FIND_NO查询而不是FIND_VALUE，找到距离 K 最近的 s 个节点。然后，将存储查询发送给所有节点。

在类似Kademlia的分布式哈希表的实现中有一些不那么重要的细节（例如，任何节点都应该查找 s 个离自己最近的节点，比如每小时一次，并通过存储查询重新发布所有存储的密钥）。我们暂时会忽略它们。

3.2.8. 引导Kademlia节点。 当Kademlia节点联机时，它首先通过查找自己的地址来填充其Kademlia路由表。 在此过程中，它标识 *s* 个离自己最近的节点。 它可以从它们下载所有已知的（密钥，值）对来填充它的分布式哈希表部分。

3.2.9. 在电报开放网络分布式哈希表中存储值。 在电报开放网络分布式哈希表中存储值与一般的类似Kademlia的分布式哈希表略有不同。 当有人希望存储一个值时，她不仅要向存储查询提供密钥 *K* 本身，还要提供其原像——即TL序列化的字符串（开头有几个预定义的TL构造函数之一），其中包含密钥的《描述》。 这个密钥描述稍后由节点以及密钥和值保存。

密钥描述描述了所存储对象的《类型》，其《拥有者》以及未来更新时的《更新规则》。拥有者通常通过密钥描述中包含的公钥来标识。如果包含它，通常只接受由相应私钥签名的更新。存储对象的《类型》通常只是一个字节字符串。但是，在某些情况下，它可能更复杂——例如，输入隧道描述（参见 **3.1.6** ）或节点地址的集合。

《更新规则》也可以不同。在某些情况下，它们只允许用新值替换旧值，前提是新值由拥有者签名（签名必须被保留为值的一部分，以后在获取这个密钥的值后由任何其他节点检查）。在其他情况下，旧值会以某种方式影响新值。例如，它可以包含序列号，只有在新序列号较大时才会覆盖旧值（以防止重放攻击）。

3.2.10. 电报开放网络分布式哈希表中的分布式《洪流跟踪器》和《网络兴趣组》。 另一个有趣的案例是，当值包含节点列表时——可能包含其IP地址和端口，或仅包含其抽象地址——并且《更新规则》在于将请求者包括在此列表中，前提是她可以确认自己的身份。

该机制可以用于创建分布式《洪流跟踪器》，其中对某个《洪流》（即，某个文件）感兴趣的所有节点可以找到对相同的洪流感兴趣或已经具有副本的其他节点。

电报开放网络(TON)存储（参见 **4.1.8** ）使用该技术来查找具有所需文件副本的节点（例如，分片链状态或旧块的快照）。 但是，更重要的用途是创建《覆盖多播子网》和《网络兴趣组》（参见 **3.3** ）。 这个想法是只有一些节点对特定分片链的更新感兴趣。 如果分片链的数量变得非常大，找到对同一分片感兴趣的节点（就是找到一个也）可能变得复杂。 这种《分布式洪流跟踪器》提供了一种查找这些节点的便捷方式。 另一种选择是从验证者请求它们，但这不是可扩展的方法，验证者可能选择不回应来自任意未知节点的此类查询。

3.2.11. 备用密钥。 到目前为止所描述的大多数《密钥类型》在其TL描述中具有额外的32位整数字段，通常等于零。 但是，如果无法从电报开放网络分布式哈希表中检索或更

新通过哈希描述获得的密钥，则增加该字段中的值，并进行新的尝试。 以这种方式，通过在受到攻击的密钥附近创建许多抽象地址并控制相应的分布式哈希表节点，不能《捕获》和《审查》密钥（即，执行密钥保留攻击）。

3.2.12. 定位服务。 位于电报开放网络(TON)中并通过 **3.1** 中描述的TON抽象数据报网络层（基于其构建的更高级别协议）提供的某些服务可能希望在某处发布其抽象地址，以便其客户知道在哪里找到它们。

但是，在电报开放网络(TON)区块链中发布服务的抽象地址可能不是最好的方法，因为抽象地址可能需要经常更改，并且因为为了可靠性或负载平衡目的而提供多个地址。

另一种方法是将公钥发布到电报开放网络(TON)区块链中，并使用一个特殊的分布式哈希表密钥，指示公钥作为TL描述字符串（参见 **2.2.5** ）中的《拥有者》，以发布服务的抽象地址的最新的列表。 这是TON服务利用的方法之一。

3.2.13. 找到电报开放网络(TON)区块链账户的拥有者。 在大多数情况下，TON区块链帐户的拥有者不希望与抽象网络地址相关联，尤其是IP地址，因为这可能会侵犯他们的隐私。 但是，在某些情况下，TON区块链帐户的拥有者可能希望发布可以联系她的一个或多个抽象地址。

典型的情况是电报开放网络(TON)付款《闪电网络》（参见 **5.2** ）中的节点，即时加密货币转移的平台。 公共TON支付节点可能不仅希望与其他对等方建立支付频道，而且还要发布可用于在稍后时间联系它以便沿着已建立的频道转移支付的抽象网络地址。

一种选择是在创建支付频道的智能合约中包含抽象网络地址。 更灵活的选择是在智能合约中包含公钥，然后按照 **3.2.12** 中的说明使用分布式哈希表。

最自然的方式是使用控制电报开放网络(TON)区块链中的帐户的相同私钥来在电报开放网络(TON)分布式哈希表中签署和发布关于与该帐户相关联的抽象地址的更新。 这几乎与 **3.2.12** 中描述的方式相同; 但是，所使用的分布式哈希表密钥需要一个特殊的密钥描述，只包含账户 *_id*本身，等于《帐户描述》的SHA256，其中包含帐户的公钥。 包含在此分布式哈希表密钥值中的签名也将包含帐户描述。

以这种方式，用于定位电报开放网络(TON)区块链账户的一些拥有者的抽象网络地址的机制变得可用。

3.2.14. 找到抽象地址。 请注意，电报开放网络(TON)分布式哈希表虽然通过TON抽象数据报网络层实现，但它本身也被TON抽象数据报网络层用于多种用途。

其中最重要的是从256位抽象地址开始定位节点或其联系数据。 这是必要的，因为即使没有其他信息被提供，TON抽象数据报网络层也应该能够将数据报发送到任意256位抽象地址。

为此，我们只简单地查找256位抽象地址为分布式哈希表中的密钥。 要么找到具有该地址的节点（即，使用该地址作为公共半持久分布式哈希表地址），在这种情况下，可以学习其IP地址和端口; 或者，可以取回输入隧道描述作为所讨论的密钥的值，由正确的私钥签名，在这种情况下，该隧道描述将用于将抽象数据报网络层数据报发送到预期的接收者。

请注意，为了使抽象地址《公共》（可从网络中的任何节点进接），其拥有者必须将其用作半永久性分布式哈希表地址，或者发布（在等于所考虑的抽象地址的分布式哈希表密钥中） 输入隧道描述，其另一个公共抽象地址（例如，半永久地址）作为隧道的入口点。 另一种选择是发布其IP地址和UDP端口。

3.3 覆盖网络和多播信息

在像TON区块链这样的多区块链系统中，即使是完整节点通常也只对某些分片链获得的更新（即新块）感兴趣。 为此，必须在电报开放网络(TON)内部建立一个特殊的覆盖（子）网络，在 **3.1** 中讨论的抽象数据报网络层协议之上，每个分片链一个。

因此，需要构建对任何愿意参与的节点开放的任意覆盖子网。 基于抽象数据报网络层的特殊流言协议将在这些覆盖网络中运行。 特别是，这些流言协议可用于在这种子网内传播（广播）任意数据。

3.3.1. 覆盖网络。 覆盖（子）网络简单地是在一些较大网络内实现的（虚拟）网络。 通常只有较大网络的某些节点参与覆盖子网，并且这些节点（物质的或虚拟的）之间只有一些《链路》是覆盖子网的一部分。

以这种方式，如果包含网络被表示为图（在数据报网络的情况下可能是完整图，例如抽象数据报网络层，其中任何节点可以容易地与任何其他节点通信），则覆盖子网络是该图的子图。

在大多数情况下，覆盖网络使用基于较大网络的网络协议构建的一些协议来实现。 它可以使用与较大网络相同的地址，或使用自定义地址。

3.3.2. 在电报开放网络(TON)中的覆盖网络。 TON中的覆盖网络建立在 **3.1** 中讨论的抽象数据报网络层协议之上; 它们也使用256位抽象数据报网络层抽象地址作为覆盖网络中的地址。 每个节点通常选择其抽象地址之一也作为其在覆盖网络中的地址。

与抽象数据报网络层相比，电报开放网络(TON)覆盖网络通常不支持将数据报发送到任意其他节点。 相反，在一些节点之间建立一些《半永久链路》（相对于所考虑的覆盖网络称为《邻居》），并且消息通常沿着这些链路（即，从节点到其邻居之一）转发。 通过这种方式，TON覆盖网络是抽象数据报网络层网络（完整）图内的（通常不是完整的）子图。

可以使用专用的对等抽象数据报网络层频道（参见 **3.1.5** ）实现与TON覆盖网络中的邻居的链路。

覆盖网络的每个节点维护一个邻居列表（相对于覆盖网络），包含它们的抽象地址（它们用于在覆盖网络中识别它们）和一些链路数据（例如，用于与它们通信的抽象数据报网络层频道）。

3.3.3. 私人和公共覆盖网络。 一些覆盖网络是公共的，这意味着任何节点都可以随意加入它们。 其他的覆盖网络是私人的，这意味着只允许某些节点被接纳（例如，那些可以证明其身份为验证者的节点。）一些私人覆盖网络甚至可以不被《普通公众》所知。 关于这种覆盖网络的信息仅对某些可信节点可用; 例如，它可以用公钥加密，并且只有具有相应私钥副本的节点才能解密该信息。

3.3.4. 中央控制的覆盖网络。 一些覆盖网络由一个或多个节点集中控制，或由一些广为人知的公钥的拥有者集中控制。 其他的是分散的，这意味着没有特定的节点负责它们。

3.3.5. 加入覆盖网络。 当一个节点想要加入一个覆盖网络时，它首先必须得知它的256位网络标识符，通常等于覆盖网络描述的SHA256——一个TL序列化对象（参见 **2.2.5** ），它可能包含， 例如，覆盖网络的中央（即，其公钥，也许是其抽象地址[36]），具有覆盖网络名称的字符串，电报开放网络(TON)区块链分片标识符（如果这是与该分片相关的覆盖网络），等等。

有时可以从网络标识符开始恢复覆盖网络描述，只需在电报开放网络(TON)分布式哈希表中查找即可。 在其他情况下（例如，对于私人覆盖网络），必须获得网络描述以及网络标识符。

3.3.6. 找到覆盖网络的一个成员。 在一个节点获知它想要加入的覆盖网络的网络标识符和网络描述之后，它必须找到属于该网络的至少一个节点。

对于不想加入覆盖网络但只想与覆盖网络通信的节点，也需要这样做; 例如，可能存在专用于收集和传播特定分片链的交易候选者的覆盖网络，并且客戶端可能想要连接到该网络的任何节点以建议交易。

[36] 或者，抽象地址可以存储在分布式哈希表中，如 **3.2.12** 中所述。

用于找到覆盖网络的成员的方法在该网络的描述中定义。 有时（特别是对于私人网络），必须已经知道能够加入的成员节点。 在其他情况下，一些节点的抽象地址包含在网络描述中。 更灵活的方法是在网络描述中仅指示负责网络的中央，然后抽象地址将通过由该中央签名的某些分布式哈希表密钥的值来获得。

最后，真正分散的公共覆盖网络可以使用 **3.2.10** 中描述的《分布式洪流跟踪器》机制，也可以借助电报开放网络分布式哈希表实现。

3.3.7. 找到覆盖网络的更多成员。 创建链接。 一旦找到覆盖网络的一个节点，就可以向该节点发送特殊查询，请求其他成员的列表，例如，被查询的节点的邻居，或在它们中间随机选择的一些节点。

这使得加入成员能够通过选择一些新得知的网络节点并建立到它们的链接（即，如在 **3.3.2** 所概述的专用抽象数据报网络层点对点频道）来填充关于覆盖网络的《邻接》或《邻居列表》。 在这之后，将向所有邻居发送特殊消息，指示新成员已准备好在覆盖网络中工作。 邻居把它们于新成员的链接包括在它们的邻居列表中。

3.3.8. 维持邻居列表。 覆盖网络节点必须时常更新其邻居列表。 一些邻居，或者至少是他们的链接（频道）可能会停止回应; 在这种情况下，必须将这些链接标记为《已中止》，必须试着重新连接到这样的邻居，并且如果这些尝试失败，则必须销毁链接。

另一方面，每个节点有时会从随机选择的邻居请求其邻居列表（或在这个邻居列表中的随机选择），并通过向其添加一些新发现的节点来使用它来部分更新其自己的邻居列表，并删除一些旧的节点，无论是通过随机的方法还是取决于它们的回应时间和数据报丢失统计数据。

3.3.9. 覆盖网络是一个随机子图。 这样，覆盖网络成为抽象数据报网络层网络内的随机子图。 如果每个顶点的度数至少为三（即，如果每个节点连接到至少三个邻居），则已知该随机图以几乎等于一的概率连接。 更确切地说，n 个顶点被断开的随机图概率是指数般小的，并且如果 $n \geq 20$，则可以完全忽略该概率。（当然，这不适用于全局网络分区的情况，当分区不同侧面的节点没有机会相互了解时。）另一方面，如果 n 小于二十，则要求每个顶点具有至少十个邻居就够了。

3.3.10. TON覆盖网络经过优化，可降低延迟。 TON覆盖网络优化由先前方法生成的《随机》网络图，如下所述。每个节点都试图保留至少三个具有最小往返时间的邻居，很少改变这个《快速邻居》列表。同时，它还至少有三个完全随机选择的《慢邻居》，因此覆盖网络图总是包含一个随机子图。这是必须的，为了保持连接性并防止将覆盖网络分

成几个未连接的区域子网。至少三个《中间邻居》被选择并保留，其具有中间往返时间，以一定常数（实际上是快速和慢速邻居的往返时间的函数）为界。

通过这种方式，覆盖网络的图形仍然保持足够的随机性以进行连接，但是针对较低的延迟和较高的吞吐量进行了优化。

3.3.11. 覆盖网络中的流言协议。覆盖网络通常用来运行所谓的流言协议之一，其实现一些全局目标，同时让每个节点仅与其邻居交互。例如，有一些流言协议可以构建一个（不是太大的）覆盖网络的所有成员的大概列表，或者计算一个（任意大的）覆盖网络的成员数量的估计，仅使用每个节点的有限的记忆（详见[**22** ,4.4.3]或[**3**]）。

3.3.12. 将网络覆盖为广播域。在覆盖网络中运行的最重要的流言协议是广播协议，其用途在于将由网络的任何节点或者可能由指定的发送方节点之一生成的广播消息传播到所有其他节点。

实际上有几种广播协议，针对不同的用例进行了优化。其中最简单的接收新广播消息并将它们传达到尚未发送该消息副本的所有邻居。

3.3.13. 更复杂的广播协议。 某些应用可能需要更复杂的广播协议。 例如，为了广播规模大的消息，可以向邻居发送不是新接收的消息本身，而是其哈希(或新消息的哈希集合）。 邻居可以在得知先前没看见的消息哈希之后请求消息本身，例如，使用 **3.1.9** 中讨论的可靠的大数据报协议 (RLDP) 进行传送。 这样，新消息将仅从一个邻居下载。

3.3.14. 检查覆盖网络的连接性。如果存在必须在这覆盖网络中的已知节点（例如，覆盖网络的《拥有者》或《创建者》），就可以检查覆盖网络的连接性。那么，所讨论的节点不时地广播包含当前时间，序列号及其签名的短消息。任何其他节点如果在不久前收到这样的广播，则可以确定它仍然与覆盖网络连接。该协议可以扩展到几个众所周知的节点;例如，它们都将发送这样的广播，并且所有其他节点将期望从超过一半的众所周知的节点接收广播。

在用于传播特定分片链的新块（或仅新块标头）的覆盖网络的情况下，节点检查连接性的好方法是跟踪到目前为止接收的最新块。因为块通常每五秒生成一次，如果超过三十秒没有收到新块，则该节点可能已经与覆盖网络断开连接。

3.3.15. 流式广播协议。最后，还有用于电报开放网络(TON)覆盖网络的流式广播协议，用于在一些分片链（《分片链任务组》）的验证者之间传播块候选者，他们为此目的创建私人覆盖网络。可以使用相同的协议将新的分片链块传播到该分片链的

所有完整节点。

该协议已在 **2.6.10** 中概述：新的（大）广播消息被分成例如 N 个一千字节的块;通过诸如里德-所罗门码或喷泉代码（例如，RaptorQ代码[15] [21]或在线代码[17]）之类的擦除代码将这些块的序列扩充为 $M \geq N$ 个块，并且这些M块以递增的块编号顺序流式传输到所有邻居。参与节点收集这些块直到它们可以恢复始发的大规模消息（必须成功接收至少 N 个块），然后指示它们的邻居停止发送流的新块，因为现在这些节点可以自己生成后续的块，因为它们拥有始发消息的副本。这些节点继续生成流的后续块并将它们发送给它们的邻居，除非邻居表明这不再是需要的。

以这种方式，节点在进一步传播之前不需要完整地下载大规模消息。 这最大限度地减少了广播延迟，尤其是与 **3.3.10** 中描述的优化结合使用时。

3.3.16. 基于现有的覆盖网络构建新的覆盖网络。 有时人们不想从头开始构建覆盖网络。相反，人们可以已知一个或多个先前存在的覆盖网络，并且要求新覆盖网络的成员资格与这些覆盖网络的组合成员资格重叠。

当一个电报开放网络(TON)分片链被分成两个，或者两个兄弟分片链被合并为一个时，就会出现一个重要的例子（参见 **2.7** ）。在第一种情况下，必须为每个新的分片链构建用于将新块传播到完整节点的覆盖网络;然而，可以预期这每一个新的覆盖网络中都包含在原始分片链的块传播网络中（并且包括其大约一半的成员）。在第二种情况下，用于传播合并的分片链的新块的覆盖网络将大概地包括与被合并的两个兄弟串联链相关的两个覆盖网络的成员的并集。

在这种情况下，新覆盖网络的描述可以包含对相关现有覆盖网络列表的显式或隐式引用。希望加入新覆盖网络的节点可以检查它是否已经是这些现有网络之一的成员，并且在这些网络中查询它们的邻居是否也对新网络感兴趣。在肯定答复的情况下，可以向这些邻居建立新的点对点频道，并且它们可以包括在新覆盖网络的邻居列表中。

这种机制并不能完全取代 **3.3.6** 和 **3.3.7** 中描述的一般机制;相反，它们都是并行运行的，用于填充邻居列表。这是为了防止无意中将新的覆盖网络分成几个未连接的子网。

3.3.17. 覆盖网络中的覆盖网络。另一个有趣的案例是电报开放网络(TON)付款的实施（即用于即时脱链价值转移的《闪电网络》;参见 **5.2** ）。在这种情况下，首先构建包含《闪电网络》的所有传输节点的覆盖网络。但是，其中一些节点已在区块链中建立了支付频道;除了 **3.3.6** , **3.3.7** 和 **3.3.8** 中描述的一般覆盖网络算法选择的任何《随机》邻居之外，它们还必须始终是该覆盖网络中的邻居。这些与建立的支付频道的邻居的《永久链接》用于运行特定的闪电网络协议，从而有效地在周围（几乎总是连接）的覆盖网络内创建覆盖子网（不一定连接，如果出现问题）。

4 电报开放网络(TON)服务和应用程序

我们已经详细讨论了TON区块链和TON网络技术。 现在我们解释一些可以将它们组合在一起以创建各种服务和应用程序的方法，并讨论TON项目本身提供的一些服务，从一开始或者从后面讲起。

4.1 电报开放网络(TON)服务实施策略

我们首先讨论如何在电报开放网络(TON)生态系统内实现不同的区块链和网络相关的应用程序和服务。 首先，进行简单分类：

4.1.1. 应用和服务。我们将交替使用《应用》和《服务》这两个词。但是，存在一种微妙且有点模糊的区别：应用程序通常直接向人类用户提供某些服务，而服务通常由其他应用程序和服务利用。例如，TON存储是一种服务，因为它的目的是代表其他应用程序和服务保留文件，虽然人类用户也可以直接使用它。一个假设的《区块链中的面书》（参见 **2.9.13** ）或电报信使，如果通过TON网络提供（即实施为“ton服务”;参见 **4.1.6** ），则更像是一个应用程序，即使某些《机器人》可能会在没有人干涉的情况下自动进接它。

4.1.2. 应用的位置：链上，脱链或混合。为电报开放网络(TON)生态系统设计的服务或应用程序需要保留其数据并将数据处理到某处。这导致以下应用程序（和服务）的分类：

- 链上应用程序（见 **4.1.4** ）：所有数据和处理都在TON区块链中。
- 脱链应用程序（见 **4.1.5** ）：所有数据和处理都在TON区块链之外，在TON网络上提供的服务器上。
- 混合应用程序（见 **4.1.7** ）：一些（但不是全部）数据和处理都在TON区块链中;其余的是通过TON网络提供的脱链服务器。

4.1.3. 集中化：集中式和分散式或分布式应用程序。另一个分类标准是应用程序（或服务）是依赖于集中式服务器集群，还是真正《分布式》（参见 **4.1.9** ）。所有链上应用程序都会自动分散和分布。脱链和混合应用可能表现出不同程度的集中化。

现在让我们更详细地考虑上述可能性。

4.1.4. 纯粹的《链上》应用程序：驻留在区块链中的分布式应用程序。 **4.1.2** 中提到的一种可能的方法是在电报开放网络(TON)区块链中完全部署《分布式应用程序》，作为一个智能合约或一组智能合约。所有数据将作为这些智能合约的永久状态的一部分保留，并且与项目的所有交互将通过发送到这些智能合约或从这些智能合约接收的（TON区块链）消息来完成。

我们已经在 **2.9.13** 中讨论过这种方法有其缺点和局限性。它也有它的优点：这样的分布式应用程序不需要运行服务器或存储其数据（它在"区块链"中运行——即在验证器的硬件上运行），并享有区块链极高（拜占庭）的可靠性和可进接性。这种分布式应用程序的开发人员不需要购买或租用任何硬件; 她需要做的就是开发一些软件（即智能合约的代码）。在那之后，她将有效地从验证者那里租用计算能力，并用Gram支付它，无论是她自己支付还是将这个负担放在用户的肩上。

4.1.5. 纯网络服务：《ton网站》和《ton服务》。 另一个极端的选择是在某些服务器上部署服务，并通过 **3.1** 中描述的抽象数据报网络层协议将其提供给用户，也可以使用一些更高级别的协议，如 **3.1.9** 中讨论的可靠大数据报协议，可用于发送远程过程调用查询，以任何自定义格式提供服务并获取这些查询的答案。通过这种方式，该服务将完全脱链，并将驻留在电报开放网络(TON)中，几乎不使用电报开放网络(TON)区块链。

TON区块链可能仅用于找到服务的抽象地址，如 **3.2.12** 中所述，也许借助于诸如TON域名系统（参见 **4.3.1** ）之类的服务来促进把类似域名的人们可读的字符串翻译成抽象地址。

在某种程度上，抽象数据报网络层网络（即电报开放网络 (TON) 类似于隐形网计划 (I^2P) ，这种（几乎）纯粹的网络服务类似于所谓的《隐形网服务》（即具有隐形网计划地址作为其入口点，并通过隐形网计划网络提供给客户端）。我们会说，驻留在电报开放网络(TON)中的这种纯粹的网络服务是《ton服务》。

《隐形网服务》可以实现HTTP作为其客户端与服务器协议;在电报开放网络(TON)环境中，《TON服务》可能只是使用可靠大数据报协议（参见 **3.1.9** ）数据报来传输HTTP查询和回应。如果它使用电报开放网络(TON)域名系统允许通过人们可读的域名把其抽象地址查到，那么对网站的类比就变得几乎完美了。有人甚至可以编写一个专门的浏览器，或者在用户机器上本地运行的特殊代理（《ton代理》），从用户使用的普通网路浏览器接受任意HTTP查询（一旦是本地IP地址和TCP端口）将代理输入到浏览器v的配置中，并通过电报开放网络(TON)将这些查询转发到服务的抽象地址。然后，用户将具有类似于万维网 (WWW) 的浏览体验。

在隐形网计划生态系统中，这种《隐形网服务》被称为《隐形网网站》。人们也可以在电报开放网络(TON)生态系统中轻松创建《ton网站》。这有点通过TON域名系统

等服务的存在而得到促进，TON域名系统利用电报开放网络(TON)区块链和电报开放网络分布式哈希表将 (TON) 域名转换为抽象地址。

4.1.6. 电报信使作为ton服务; MTProto超过可靠大数据报协议。 我们顺便提一下，电报信使 [38] 用于客戶端与服务器交互的MTProto协议 [37] 可以很容易地嵌入到 **3.1.9** 中讨论的可靠大数据报协议中，从而有效地将电报转换为*c*服务。 由于电报开放网络(TON)代理技术可以为ton网站或ton服务的最终用戶透明地开启，实施的层次低于可靠大数据报协议和抽象数据报网络层协议（参见 **3.1.6** ），这将使电报不能被阻挡。 当然，其他消息传递和社交网络服务也可能从这项技术中受益。

4.1.7. 混合服务：部分脱链，部分链上。有些服务可能使用混合方法：进行大部分脱链处理，但也有一些链上部分（例如，向用戶注册其义务，反之亦然）。通过这种方式，部分状态仍将保留在电报开放网络(TON)区块链中（即，不可变的公共分类帐），并且服务或其用戶的任何不当行为都可能受到智能合约的惩罚。

4.1.8. 示例：保持文件脱链; 电报开放网络(TON)存储。 电报开放网络(TON)存储做出了这种服务的一个例子。在最简单的形式中，它允许用戶通过保持链上只存储要存储的文件的哈希值来脱链存储文件，并且可能是智能合约，合约中同意在给定时间段内为了预先商定的费用保留有问题的文件。实际上，文件可以被细分为一些小尺寸（例如，一千字节）的块，通过如同里德-所罗门码或喷泉代码的擦除代码来增强，可以为增强的块序列构造默克尔树哈希。 ，这个默克尔树哈希可以在智能合约中发布，而不是与文件的通常哈希一起发布。这有点让人联想到文件存储在洪流中的方式。

一种更简单的存储文件形式是完全脱链的：可以为新文件创建一个《洪流》，并使用电报开放网络分布式哈希表作为此洪流的《分布式洪流跟踪器》（参见 **3.2.10** ）。这对于流行的文件实际上可能非常有效。 但是，没有任何可用性保证。 例如，一个假设的《区块链面书》（参见 **2.9.13** ），可能会选择让其用戶的个人资料照片完全脱链地保留在这种《洪流》里，可能会失去普通（不是特别受欢迎的）用戶的照片 ，或至少有可能无法长时间呈现这些照片。 电报开放网络(TON)存储技术主要是脱链的，但使用链上智能合约来强制存储文件的可用性，可能更适合此工作。

4.1.9. 分散的混合服务，或《雾服务》。 到目前为止，我们已经讨论了集中式混合服务和应用。 虽然它们的链上组件以分散和分布的方式处理，位于区块链中，但它们的脱链组件依赖于服务提供商以通常的集中方式控制的一些服务器。 可以从其中一家大公司提供

[37] https://core.telegram.org/mtproto
[38] https://telegram.org/

的云计算服务中租用计算能力，而不是使用某些专用服务器。 但是，这不会导致服务的脱链组件的分散化。

实现服务的脱链组件的分散方法在于创建一个市场，任何拥有所需硬件并愿意租用其计算能力或磁盘空间的人都会向需要它们的人提供服务。

例如，可能存在一个注册表（也可能称为《市场》或《交换》），其中所有对保持其他用户的文件感兴趣的节点都会发布其联系信息，以及它们的可用存储容量，可用性策略和 价格。 那些需要这些服务的人可能会在那里查找，如果对方同意，则在区块链中创建智能合约并上传文件以进行脱链存储。 通过这种方式，像电报开放网络(TON)存储这样的服务变得真正分散，因为它不需要依赖任何集中的服务器集群来存储文件。

4.1.10. 示例：《雾计算》平台作为分散的混合服务。当想要执行某些特定计算（例如，三维渲染或训练神经网络）时，就是这种分散的混合应用的另一个例子，通常需要特定且昂贵的硬件。那些拥有这种设备的人可能会通过类似的《交换》提供服务，那些需要这些服务的人会租用他们，双方的义务通过智能合约进行登记。这类似于《雾计算》平台，如 Golem (https://golem.network/) 或 SONM (https://sonm.io/)，保证提供的。

4.1.11. 示例：电报开放网络(TON)代理是一项雾服务。 TON代理提供了雾服务的另一个示例，其中希望提供其服务（有或没有补偿）作为抽象数据报网络层网络流量的隧道的节点可能会注册，而需要它们的节点可能会根据价格，延迟和带宽提供选择其中一个节点。之后，可以使用由TON付款提供的支付渠道来处理那些代理服务的小额支付，例如，每转移128千字节就收集付款。

4.1.12. 示例：电报开放网络(TON)付款是一项雾服务。 TON付款平台（参见 **5** ）也是这种分散的混合应用程序的一个例子。

4.2 连接用户和服务提供商

我们在 **4.1.9** 中已经看到，《雾服务》（即混合分散服务）通常需要一些市场，交易所或注册管理机构，在这些地方那些需要特定服务的人可能会遇到提供这些服务的人。

这些市场很可能实现为集中或分布式的链上，脱链或混合服务本身。

4.2.1. 示例：连接到电报开放网络(TON)付款。例如，如果想要使用TON付款（参见 **5** ），第一步是找到《闪电网络》的至少一些现有的传输节点（参见 **5.2** ），并与它们建立支付频道，如果他们肯。借助于《周围》覆盖网络可以找到一些节点，该覆盖网络应

该包含所有传输闪电网络节点（参见 **3.3.17** ）。但是，尚不清楚这些节点是否愿意创建新的支付频道。因此，需要注册表，其中准备创建新链接的节点可以发布它们的联系信息（例如，它们的抽象地址）。

4.2.2. 示例：将文件上传到电报开放网络(TON)存储。类似地，如果想要将文件上传到TON存储器，她必须找到一些愿意签署智能合约的节点来绑定它们以保留该文件的副本（或者任何低于特定大小限制的文件）。因此，需要提供用于存储文件的服务的节点的注册表。

4.2.3. 链上，混合和脱链注册管理机构。这样的服务提供商注册表可以完全在链上实现，借助智能合约将注册表保留在其永久存储中。然而，这将是非常缓慢和昂贵的。混合方法更有效，其中相对较小且很少更改的链上注册表仅用于指出某些节点（通过其抽象地址或其公钥，可用于定位实际抽象地址，如 **3.2.12** ），提供脱链（集中）注册服务。最后，分散的，纯粹的脱链方法可能包括公共覆盖网络（参见 **3.3** ），那些愿意提供服务的人，或那些想要购买某人服务的人，只需通过他们的私钥进行广播他们的优惠。如果要提供的服务非常简单，甚至可能不需要广播优惠：覆盖网络本身的大概成员资格可以用作愿意提供特定服务的人的《注册表》。然后，需要此服务的客戶端可能会定位（参见 **3.3.7** ）并查询此覆盖网络的某些节点，然后查询其邻居，如果已知的节点尚未准备好满足其需求。

4.2.4. 侧链中的注册表或交换。实施分散式混合注册管理机构的另一种方法是创建一个独立的专用区块链（《侧链》），由其自己的一系列自我声明的验证者维持，他们在链上智能合约中发布其身份，并为所有相关方提供网络进接权，通过专用的覆盖网络收集交易候选者和广播块的更新（参见 **3.3** ）。然后，此侧链的任何完整节点都可以维持自己的共享注册表副本（基副本等于此侧链的全局状态），并处理与此注册表相关的任意查询。

4.2.5. 工作链中的注册表或交换。另一种选择是在电报开放网络(TON)区块链内创建专用工作链，专门用于创建注册，市场和交换。与使用基本工作链中的智能合约相比，这可能更有效，更便宜（参见 **2.1.11** ）。但是，这仍然比在侧链中维持注册管理机构更昂贵（参见 **4.2.4** ）。

4.3 进接电报开放网络(TON)服务

我们在 **4.1** 中讨论了可能用于创建驻留在电报开放网络(TON)生态系统中的新服务和应用程序的不同方法。 现在我们讨论如何进接这些服务，以及电报开放网络(TON)将提供的一些《帮助服务》，包括TON域名系统和TON存储。

4.3.1. 电报开放网络(TON)域名系统：大部为链上分层域名服务。 TON域名系统是一种预定义的服务，它使用一组智能合约来保存从人们可读域名到抽象数据报网络层网络节点和TON区块链账户和智能合约的（256位）地址的映射。

虽然任何人原则上可以使用电报开放网络(TON)区块链来实现这样的服务，但是当应用程序或服务想要将人们可读标识符转换为地址时，默认情况下使用具有众所周知的接口的这种预定义服务是有用的。

4.3.2. 电报开放网络(TON)域名系统用例。 例如，希望将一些加密货币转移给另一个用户或商家的用户可能更愿意记住该用户或商家的帐户的TON域名系统的域名，而不是保留他们的256位帐户标识符和复制粘贴他们进入轻型钱包客户端的收件人字段。

同样，TON域名系统可用于定位智能合约的帐户标识符或ton服务和ton网站的入口点（参见 **4.1.5** ），启用专业客户端（《ton浏览器》）或通常的互联网浏览器结合专门的ton代理扩展或独立应用程序，为用户提供类似万维网的浏览体验。

4.3.3. 电报开放网络(TON)域名系统智能合约。 电报开放网络(TON)域名系统通过特殊（域名系统）智能合约树实现。 每个域名系统智能合约都负责注册某些固定域的子域。 将保留电报开放网络(TON)域名系统的第一级域的《根》域名系统智能合约位于主链中。 其帐户标识符必须硬编码到希望直接进接电报开放网络(TON)域名系统数据库的所有软件中。

任何域名系统智能合约都包含一个Hashmap，将可变长度的以空值结尾的UTF-8字符串映射到它们的《值》中。 此散列图实现为二进制帕特里夏树，类似于 **2.3.7** 中描述的，但支持可变长度位串作为钥。

4.3.4. 域名系统Hashmap或电报开放网络(TON)域名系统记录的值。至于这些值，它们是由TL方案描述的《电报开放网络(TON)域名系统记录》（参见 **2.2.5** ）。它们由一个《幻数》组成，选择一个支持的选项，然后选择一个帐户标识符，或一个智能合约标识符，或一个抽象网络地址（参见 **3.1** ），或一个用于定位抽象地址的公钥服务（参见 **3.2.12** ），或覆盖网络的描述，等等。一个重要的案例是另一个域名系统智能合约：在这种情况下，该智能合约用于解析其域的子域。通过这种方式，可以为不同的域创建单独的注册表，由这些域的拥有者控制。

这些记录还可能包含到期时间，缓存时间（通常非常大，因为更新区块链中的值通常很昂贵），并且在大多数情况下是对相关子域拥有者的引用。拥有者有权更改此记录（特别是拥有者字段，从而将域名转移给其他人的控制权），并延长它。

4.3.5. 注册现有域的新子域。 为了注册现有域的新子域，人们只需向智能合约发送消息，智能合约是该域的注册商，包含要注册的子域（即密钥），几种预定义格式之一的值，拥有者的身份，到期日期以及域拥有者确定的一些加密货币。

子域名以《先到先得》的方式注册。

4.3.6. 从域名系统智能合约中检索数据。 原则上，如果智能合约的持久存储内的Hashmap的结构和位置被得知，那么包含域名系统智能合约的主链或分片链的任何完整节点都可以查找该智能合约的数据库中的任何子域。

但是，这种方法仅适用于某些域名系统智能合约。 如果使用非标准域名系统智能合约，它将失败。

相反，可以使用基于通用智能合约接口和获取方法的方式（参见 **4.3.11** ）。 任何域名系统智能合约都必须定义一个带有《已知签名》的《获取方法》，该方法被调用以查找密钥。 由于这种方法对其他智能合约也有意义，特别是那些提供链上和混合服务的合同，我们在 **4.3.11** 中详细解释了这一点。

4.3.7. 翻译电报开放网络(TON)域名系统域。 一旦任何完整节点（单独或代表某个轻客户端）可以查找任何域名系统智能合约的数据库中的条目，就可以递归地翻译任意电报开放网络(TON)域名系统域名，从众所周知的固定根DNS智能合约（帐户）标识符开始。

例如，如果想要翻译 A.B.C，则在根域数据库中查找密钥 .C，.B.C 和 A.B.C. 如果找不到第一个，但第二个被找到，并且其值是对另一个域名系统智能合约的引用，则在该智能合约的数据库中查找 A 并检索最终值。

4.3.8. 为轻型节点翻译电报开放网络(TON)域名系统域。 通过这种方式，主链的完整节点以及域查找过程中涉及的所有分片链可以在没有外部帮助的情况下将任何域名转换为其当前值。 轻节点可以请求完整节点代表它执行此操作并返回值以及默克尔树证明（参见 **2.5.11** ）。 这种默克尔树证明将使轻节点能够验证答案是否正确，因此与通常的域名系统协议相比，这种电报开放网络(TON)域名系统回应不能被恶意拦截器《假装》。

因为没有节点被要求成为关于所有分片链的完整节点，所以实际的电报开放网络(TON)域名系统域转换将涉及这两种策略的组合。

4.3.9. 专用的《电报开放网络(TON)域名系统服务器》。 可以提供简单的《电报开放网络(TON)域名系统服务器》，它将接收远程过程调用《域名系统》查询（例如，通过 **3.1** 中描述的抽象数据报网络层或可靠大数据报协议），请求服务器转换给定域，通过转发一些子查询来处理这些查询。 如有必要，可以使用其他（完整）节点，并返回始发查询的答案，如果需要，可以通过默克尔树证明进行扩充。

这样的《域名系统服务器》可以使用 **4.2** 中描述的方法之一向任何其他节点，尤其是轻客戶端提供其服务（免费或不免费）。 请注意，如果这些服务器被认为是电报开放网络(TON)域名系统服务的一部分，它们将有效地将其从分布式链上服务转换为分布式混合服务（即《雾服务》）。

这就结束了我们对电报开放网络(TON)域名系统服务的简要概述，电报开放网络(TON)域名系统服务是TON区块链和TON网络实体的人们可读域名的可扩展的链式注册表。

4.3.10. 访问智能合约中保存的数据。 我们已经看到，有时需要进接存储在智能合约中的数据而不改变其状态。

如果知道智能合约实现的细节，就可以从智能合约的持久存储中提取所有需要的信息，这些信息可供智能合约所在的分片链的所有完整节点使用。但是，这是一种非常不优雅的做法，很大程度上取决于智能合约的实施。

4.3.11. 智能合约的《获取方法》。更好的方法是在智能合约中定义一些获取方法，即某些类型的入站消息在交付时不会影响智能合约的状态，但会生成一个或多个包含获取方法的《结果》的输出消息。以这种方式，可以从智能合约获得数据，而仅知道它实现具有已知签名的获取方法（即，要发送的入站消息的已知格式和作为结果要接收的出站消息）。

这种方式更加优雅，符合面向对象编程 (OOP) 。但是，到目前为止它有一个明显的缺陷：一个必须实际将交易提交到区块链中（将获取消息发送到智能合约），等待它由验证者提交和处理，从新块中提取答案，并且支付汽油费（即，因为在验证者的硬件上执行获取方法）。这是浪费资源：获取方法不会改变智能合约的状态，因此不需要在区块链中执行。

4.3.12. 试着执行智能合约的获取方法。我们已经评论过（参见 **2.4.6** ）任何完整节点都可以试着执行任何智能合约的任何方法（即，向智能合约传递任何消息），从智能合约的给定状态开始，而不实际提交相应的交易。完整节点可以简单地将所考虑的智能合约的代码加载到电报开放网络(TON)虚拟机中，从分片链的全局状态（被分片链的所有完整节

点都知道）初始化其持久存储，并执行智能合约代码。入站消息作为其输入参数。创建的输出消息将产生此计算的结果。

这样，任何完整节点都可以评估任意智能合约的任意获取方法，前提是它们的签名（即入站和出站消息的格式）是已知的。节点可以跟踪在该评估期间进接的分片链状态的细胞，并创建所执行的计算的有效性的默克尔树证明，以获得可能已经要求整个节点这样做的轻节点（参见 **2.5.11** ）。

4.3.13. TL方案中的智能合约接口。 回想一下智能合约实现的方法（即它接受的输入消息）本质上是一些TL序列化的对象，可以通过TL方案来描述（参见 **2.2.5** ）。 得到的输出消息也可以用相同的TL方案来描述。 通过这种方式，智能合约提供给其他账户和智能合约的界面可以通过TL方案形式化。

特别是，智能合约支持的获取方法（的一部分）可以通过这种形式化的智能合约接口来描述。

4.3.14. 智能合约的公共接口。 请注意，可以发布形式化的智能合约接口，以TL方案（表示为TL源文件;参见 **2.2.5** ）或序列化形式 [39] 发布 - 例如，在特殊领域，在智能合约帐户描述中，存储在区块链中，或单独存储，如果此接口将被多次引用。 在后一种情况下，支持的公共接口的哈希可以合并到智能合约描述中而不是合并到接口描述本身。

这种公共接口的一个例子是域名系统智能合约，它应该实现至少一种用于查找子域的标准获取方法（参见 **4.3.6** ）。 注册新子域的标准方法也可以包含在域名系统智能合约的标准公共接口中。

4.3.15. 智能合约的用户界面。智能合约的公共界面的存在也有其他好处。例如，钱包客户端应用程序可以在根据用户的请求检查智能合约时下载这样的界面，并显示智能合约支持的公共方法列表（即可用动作的列表），可能具有一些人们可读的评论，如果在正式界面中有评论提供。在用户选择这些方法之一之后，可以根据TL方案自动生成表单，其中将提示用户所选方法所需的所有字段以及所需的加密货币量（例如，Gram），附在此请求上。提交此表单将创建一个新的区块链交易，其中包含从用户的区块链帐户发送的刚刚编写的邮件。

通过这种方式，如果这些智能合约已经发布了他们的界面，则用户将能够通过填写和提交某些表格以用户友好的方式与钱包客户端应用程序中的任意智能合约进行交互。

4.3.16. 《ton服务》的用户界面。 事实证明，《ton服务》（即驻留在电报开放网络(TON)

[39] TL方案可以自己也被TL序列化;比照https://core.telegram.org/mtproto/TL-tl

中的服务以及通过抽象数据报网络层和可靠大数据报协议接受查询 **3** ;参见 **4.1.5** ）也可能从TL方案描述的公共接口中获益（ 参见 **2.2.5** ）。 客戶端应用程序（例如轻型钱包或《ton浏览器》）可能会提示用戶选择其中一种方法并使用界面定义的参数填写表单，类似于刚刚在 **4.3.15** 中讨论的内容。 唯一的区别是生成的TL序列化消息不作为区块链中的交易提交; 相反，它作为远程过程调用查询被发送到所讨论的《ton服务》的抽象地址，并且根据形式接口（即，TL方案）解析和显示对该查询的回应。

4.3.17. 通过电报开放网络(TON)域名系统定位用户界面。 包含ton服务或智能合约帐戶标识符的抽象地址的电报开放网络(TON)域名系统记录还可以包含描述该实体的公共（用戶）接口或几个被支持的接口的可选字段。 然后，客戶端应用程序（无论是钱包，ton浏览器还是ton代理）将能够以统一的方式下载界面并与相关实体（无论是智能合约还是ton服务）进行交互。

4.3.18. 模糊链上和脱链服务之间的区别。通过这种方式，最终用戶模糊了链上，脱链和混合服务（参见 **4.1.2** ）之间的区别：她只需将所需服务的域名输入到她的浏览器的地址行中或钱包中，其余部分由客戶端应用程序天衣无缝地处理。

4.3.19. 轻型钱包和电报开放网络(TON)实体浏览器可以内置到电报信使客户端中。现在出现了一个有趣的机会。实现上述功能的轻型钱包和TON实体浏览器可以嵌入到电报信使智能手机客戶端应用程序中，从而将该技术带给超过两亿人。用戶可以通过在消息中包含电报开放网络(TON) 统一资源标志符（参见 **4.3.22** ）来向TON实体和资源发送超链接;如果选择了这样的超链接，将由接收方的电报客戶端应用程序在内部打开，并且将开始与所选实体的交互。

4.3.20. 《ton网站》作为支持HTTP接口的ton服务。ton网站只是一种支持HTTP接口的ton服务，可能还有其他一些接口。 可以在相应的电报开放网络(TON)域名系统记录中公布该支持。

4.3.21. 超链接。 请注意，ton网站返回的HTML页面可能包含ton超链接——即通过特制统一资源标志符方案引用其他ton网站，智能合约和帐戶（参见 **4.3.22** ）——包含抽象网络地址 ，帐戶标识符或人们可读的电报开放网络(TON)域名系统域。 然后，当用戶选择它时，《ton浏览器》可能会跟随这样的超链接，检测要使用的界面，并显示 **4.3.15** 和 **4.3.16** 中概述的用戶界面表格。

4.3.22. 超链接网址可以指定一些参数。 超链接网址不仅可以包含电报开放网络(TON)域名系统域或所讨论的服务的抽象地址，还可以包含要调用的方法的名称以及其部分或全部参数。 可能的统一资源标志符方案可能如下所示：

ton：// <域> / <方法> ? <字段1> = <值1> & <字段2>=?. . .

当用户在ton浏览器中选择这样的链接时，要么立即执行操作（特别是如果它是智能合约的获取方法，匿名调用），要么显示部分填充的表单，被用户明确确认并提交（这可能是付款表格所必需的）。

4.3.23. 发帖动作。 ton网站可以嵌入到HTML页面中，它返回一些看似常见的发帖表单，发帖操作通过合适的电报开放网络(TON)网址引用ton网站，ton服务或智能合约。 在这种情况下，一旦用户填写并提交该自定义表单，就会立即或在明确确认之后采取相应的操作。

4.3.24. 电报开放网络(TON)万维网。 所有这些将导致创建整个网络的交叉引用实体，驻留在TON网络中，终端用户可通过ton浏览器访问该网络，从而为用户提供类似万维网的浏览体验。 对于终端用户，这将最终使区块链应用程序与他们已经习惯的网站基本相似。

4.3.25. 电报开放网络(TON)万维网的优点。 这种链上和脱链服务的《TON万维网》与传统服务相比具有一些优势。 例如，付款本质上集成在系统中。 用户身份可以始终呈现给服务（通过在交易和远程过程调用请求上自动生成的签名），或随意隐藏。 服务无需检查和重新检查用户凭据; 这些凭证可以一次性发布在区块链中。 用户网络匿名可以通过TON代理轻松保存，并且所有服务都将是不可阻止的。 微支付也很容易，因为ton浏览器可以与TON支付系统集成。

5 电报开放网络(TON)付款

我们将在本文中简要讨论的电报开放网络(TON)项目的最后一个组成部分是TON付款，这是（微）支付渠道和《闪电网络》价值转移的平台。 它将启用《即时》支付，而无需将所有交易提交到区块链中，支付相关的交易费用（例如，用于消耗的气油），并等待五秒钟直到包含所讨论的交易的块被确认。

这种即时支付的总体开销很小，人们可以将它们用于小额支付。 例如，电报开放网络(TON)文件存储服务可能会为每128百万字节的下载数据向用户收费，或者付费的电报开放网络(TON)代理可能需要为每中继128千字节的流量提供一些微小的小额支付。

虽然电报开放网络(TON)付款可能会晚于TON项目的核心组件发布，但最初需要考虑一些因素。 例如，用于执行电报开放网络(TON)区块链智能合约代码的TON虚拟机（TON VM;参见 **2.1.20** ）必须支持默克尔树证明的一些特殊操作。 如果始发设计中不存在此类支持，则在稍后阶段添加它可能会成为问题（参见 **2.8.16** ）。 但是，我们将看到TON虚拟机可以自然支持《智能》支付频道（参见 **5.1.9** ）。

5.1 付款频道

我们首先讨论点对点支付频道，以及如何在TON区块链中实施这些频道。

5.1.1. 支付频道的提议。假设 A 和 B 两方都知道他们将来需要相互支付很多款项。他们不是将每笔付款作为区块链中的交易，而是创建一个《共享资金》（或者可能是一个只有两个账户的小型私人银行），并为其提供一些资金： A 贡献 a 个代币，B 贡献 b 个代币。这是通过在区块链中创建一个特殊的智能合约并将钱汇给它来实现的。

在创建《共享资金》之前，双方同意某个协议。他们将跟踪共享资金的状态——即共享资金中的余额。最初，状态是 (a, b) ，意思是 a 个代币实际上属于 A ，b 个代币属于 B. 然后，如果 A 想要向 B 支付 d 个代币，他们可以简单地同意新状态是 $(a', b') = (a - d; b + d)$ 。之后，如果 B 想要向 A 支付 d' 个代币，则状态将变为 $(a'', b'') = (a' + d', b' - d')$，依此类推。

所有这些共享资金内余额的更新都是完全脱链的。当双方决定从共享资金中取出应付资金时，他们会根据共享资金的最终状态这样做。这是通过向智能合约发送特殊消息来实现的，其中包含商定的最终状态 (a^*, b^*) 以及 A 和 B 的签名。然后智能合约发送 a^* 个代币到 A^*，b^* 个代币到 B 然后自我销毁。

此智能合约以及 A 和 B 用于更新共享资金状态的网络协议是 A 和 B 之间的简单支付频道。根据 **4.1.2** 中描述的分类，它是一种混合服务：它的状态一部分存在于区块

链（智能合约）中，但其大大部状态更新都是脱链（通过网络协议）执行的。如果一切顺利，双方将能够按照自己的意愿执行尽可能多的付款（唯一的限制是频道的《容量》不会超支——即，他们在付款频道中的余额都保持不变是非负的），只将两个交易提交到区块链中：一个用于打开（创建）支付频道（智能合约），另一个用于关闭（销毁）它。

5.1.2. 无信任的支付频道。 前面的例子有些不切实际，因为它假设双方都愿意合作，绝不会欺骗以获得一些优势。 例如，想象一下，A 选择不用 $a' < a$ 来表示最终余额 $(a',\ b')$。这将使 B 陷入困境。

为了防止这种情况，人们通常会尝试开发无信任的支付频道协议，这些协议不要求各方相互信任，并制定惩罚任何试图欺骗的一方的规定。

这通常借助签名来实现。 支付频道智能合约知道 A 和 B 的公钥，并且如果需要可以检查他们的签名。 支付频道协议要求各方签署中间状态并将签名彼此发送。 然后，如果其中一方欺骗——例如，假装支付渠道的某些状态从未存在——可以通过在该状态上显示其签名来证明其不当行为。 支付频道智能合约充当《链上仲裁者》，能够处理双方关于彼此的投诉，并通过没收所有资金并将其授予另一方来惩罚有罪方。

5.1.3. 简单的双向同步无信用支付频道。 考虑以下更实际的例子：让支付频道的状态由三元组 (δ, i, o_i) 描述，其中 i 是状态的序列号（它原来是零，然后当后续状态出现时它加一），δ_i 是频道的不平衡（意味着 A 和 B 分别拥有 $a+\delta_i$ 和 $b-\delta_i$ 个代币），o_i 是允许生成下一个状态的一方（A 或 B）。 在进一步取得进展之前，每个状态必须由 A 和 B 签署。

现在，如果 A 想要将 d 个代币转移到支付频道内的 B，并且当前状态是 $S_i=(\delta, i, o_i)$，其中 $o_i=A$，那么它只是创建一个新状态 $S_{i+1} = (\delta_i - d, i+1, o_{i+1})$，签名，然后将其与签名一起发送给 B. 然后 B 通过签名并将其签名副本发送给 A 来确认。之后，双方都拥有新签名的副本，并且可能会发生新的转移。

如果 A 想要在具有 $o_i = B$ 的状态 S_i 中将代币转移到 B，则它首先要求 B 提交具有相同不平衡 $\delta_{i+1}=\delta_i$ 的后续状态 S_{i+1}，但是具有 $o_{i+1}=A$。之后，A 将能够进行转移。

当双方同意关闭支付频道时，他们都将他们的特殊最终签名放在他们认为是最终的状态 S_k 上，并通过发送最终状态来调用支付频道智能合约的干净的或双边最终确定方法以及两个最终签名。

如果另一方不同意提供其最终签名，或它停止回应，则可以单方面关闭该频道。为此，希望这样做的一方将调用单方面的最终确定方法，向智能合约发送其最终状态的版本，最终签名以及具有另一方签名的最新状态。在此之后，智能合约不会立即对收到的最终状态采取行动。相反，它等待一段时间（例如，一天）以使另一

方呈现其最终状态的版本。当另一方提交其版本并且结果与已提交的版本兼容时，《真实》最终状态将由智能合约计算并用于相应地分配资金。如果另一方未能将其最终状态的版本呈现给智能合约，则根据所呈现的最终状态的唯一副本重新分配资金。

如果双方中的一方作弊——例如，签名两个不同的状态为最终状态，或者通过签署两个不同的下一个状态 S_{i+1} 和 S'_{i+1}，或者通过签署无效的新状态 S_{i+1}（例如，具有不平衡 $\delta_{i+1} < -a$ 或 $> b$）——然后另一方可以将此不当行为的证据提交给智能合约的第三种方法。有罪的一方将立即完全失去其在支付渠道中的份额而受到惩罚。

这种简单的支付频道协议在某种意义上是公平的，即无论是否有另一方的合作，任何一方都可以随时获得应付款，并且如果它试图作弊，可能会失去承诺支付频道的所有资金。

5.1.4. 同步支付频道作为一个简单的带有两个验证者虚拟区块链。 以上简单同步支付频道的示例可以如下重新制作。想象一下状态 $S_0, S_1, ..., S_n$ 的序列实际上是一个非常简单的区块链的块序列。该区块链的每个块基本上仅包含区块链的当前状态，并且可能是对前一个区块的引用（即，其哈希）。A 和 B 双方都充当此区块链的验证者，因此每个区块必须收集其两个签名。区块链的状态 S_i 定义了下一个区块的指定生产者 o_i，因此 A 和 B 之间没有产生下一个区块的竞赛。允许生产者 A 创建将资金从 A 转移到 B 的块（即，减少不平衡：$\delta_i+1 \leq \delta_i$），B 只能将资金从 B 转移到 A（即增加 δ）。

如果两个验证者就区块链的最终区块（和最终状态）达成一致，则通过收集双方的特殊《最终》签名来最终确定，并将其与最终区块一起提交给频道智能合约进行处理和相应地重新分配资金。

如果验证者签署了无效区块，或者创建了一个分叉，或者签署了两个不同的最终区块，则可以通过向智能合约提供其不当行为的证明来惩罚它，该合同充当两个验证者的《链上仲裁者》；然后，违规方将失去保留在支付渠道中的所有资金，这类似于失去其权益的验证者。

5.1.5. 异步支付频道作为具有两个工作链的虚拟区块链。 **5.1.3** 中讨论的同步支付频道具有一定的缺点：在另一方确认之前的交易之前，不能开始下一笔交易（支付频道内的汇款）。这可以通过用两个交互虚拟工作链（或者说是分片链）的系统替换 **5.1.4** 中讨论的单个虚拟区块链来解决。

这些工作链中的第一个工作链仅包含 A 的交易，其块只能由 A 生成；其状态为 $S_i = (i, \phi_i, j, \psi_j)$，其中 i 是块序列号（即到目前为止由 A 执行的交易计数或汇款），ϕ_i 是到目前为止从 A 转移到 B 的总量，j 是 A 知道的 B 区块链中最近有效区块的序列号，ψ_j 是在其 j 交易中从 B 转移到 A 的金额。放在第 j 个块上的 B 的签名也应该是这个状态的一部分。

也可以包括该工作链的前一个块和另一个工作链的第j个块的哈希。S_i的有效性条件包括$\phi_i \geq 0$，如果$i>0$则$\phi_i \geq \phi_i - 1$，$\psi_j \geq 0$，以及$-a \leq \psi_j - \phi_i \leq b$。

类似地，第二个工作链仅包含B的交易，其块仅由B生成；它的状态是$T_j = (j, \psi_j, i, \phi_i)$，具有相似的有效性条件。

现在，如果A想要将一些钱转移到B，它只需在其工作链中创建一个新块，签名并发送给B，而无需等待确认。

支付频道由A签署（其版本）区块链的最终状态（具有特殊的《最终签名》），B签署其区块链的最终状态，并将这两个最终状态呈现给支付频道智能合约的干净的最终确定方法完成。单边最终确定也是可能的，但在这种情况下，智能合约必须等待另一方提出其最终状态的版本，至少在某个宽限期内。

5.1.6. 单向支付频道。 如果只有A需要向B付款（例如，B是服务提供商，A是其客户），则可以创建单向支付频道。从本质上讲，它只是**5.1.5**中描述的第一个工作链，没有第二个工作链。相反，可以说**5.1.5**中描述的异步支付频道包括两个单向支付频道，或《半频道》，由同一智能合约管理。

5.1.7. 更复杂的支付频道。承诺。我们将在后面的**5.2.4**中看到，《闪电网络》（参见**5.2**）通过多个支付频道的链条实现即时汇款，需要从所涉及的支付频道获得更高的复杂程度。

特别是，我们希望能够实施《承诺》或《有条件汇款》：A同意向B发送c个代币，但B只有在满足某个条件时才会获得钱，例如，如果B可以用一个已知的v值给$\text{HASH}(u) = v$表示一些字符串u。否则，A可以在一段时间后收回钱。

这样的承诺可以通过简单的智能合约轻松实现。但是，我们希望承诺和其他类型的有条件汇款可以在支付频道中进行脱链，因为它们大大简化了《闪电网络》中存在的一系列支付频道的资金转移（参见**5.2.4**）。

5.1.4和**5.1.5**中概述的《支付频道作为简单的区块链》描绘在这里变得很方便。现在我们考虑一个更复杂的虚拟区块链，其状态包含一组这样未实现的《承诺》，以及锁定在这些承诺中的资金数量。这个区块链——或异步情况下的两个工作链——必须通过它们的哈希引用前面的块。然而，一般机制仍然相同。

5.1.8. 复杂的支付频道智能合约面临的挑战。请注意，虽然复杂的支付频道的最终状态仍然很小，而且《干净》的最终确定很简单（如果双方已就其应付金额达成一致，并且双方已签署协议，没有其他事情需要完成），单边最终确定和惩罚欺诈行为的方法需要更加复杂。实际上，他们必须能够接受不当行为的默克尔树证明，并检查支付频道区块链

的更复杂的交易是否已得到正确处理。

换句话说，支付频道智能合约必须能够使用默克尔树证明，检查其《哈希有效性》，并且必须包含支付频道（虚拟）区块链的 *ev_trans* 和 *ev_block* 功能（参见 **2.2.6** ）的实现。

5.1.9. 电报开放网络(TON)虚拟机支持《智能》支付频道。 用于运行TON区块链智能合约代码的TON虚拟机可以执行《智能》或复杂支付频道所需的智能合约（参见 **5.1.8** ）。

在这一点上，《一切都是一袋细胞》范式（参见 **2.5.14** ）变得非常方便。由于所有块（包括短暂支付频道区块链的块）都表示为细胞袋（并由一些代数数据类型描述），并且消息和默克尔树证明也同样，因此可以轻松地将默克尔树证明嵌入到发送到付款频道智能合约的入站信息。默克尔树证明的《哈希条件》将被自动检查，并且当智能合约进接所呈现的《默克尔树证明》时，它将把它当作相应的代数数据类型的值——尽管是不完整的，有些树的子树被包含被省略的子树的默克尔哈希的特殊节点替换。然后，智能合约将使用该值，这可能代表支付频道（虚拟）区块链及其状态的块，并将评估该区块链在这个区块和以前的状态的 *ev_block* 功能（参见 **2.2.6** ）。然后，计算结束，并且可以将最终状态相比于块中宣称的状态，或者相比于在尝试进接缺席子树时抛出《缺席节点》异常，指示默克尔树证明无效。

通过这种方式，使用电报开放网络(TON)区块链智能合约，智能支付频道区块链验证码的实施变得非常简单。 有人可能会说电报开放网络(TON)虚拟机内置了对检查其他简单区块链有效性的支持。 唯一的限制因素是默克尔树证明的大小将被合并到智能合约的入站信息中（即进入交易）。

5.1.10. 智能支付频道内的简单支付频道。 我们想讨论在现有支付频道内创建简单（同步或异步）支付频道的可能性。

虽然这看起来有点令人费解，但理解和实施并不比 **5.1.7** 中讨论的《承诺》困难得多。 基本上，如果出现一些哈希问题的解决方案，而不是承诺向另一方支付 c 个代币，A 承诺根据其他（虚拟）支付频道区块链的最终结算向 C 支付最多 c 个代币。 一般来说，这个其他支付频道区块链甚至不需要在 A 和 B 之间; 它可能涉及一些其他方，比如 C 和 D，愿意分别将 c 和 d 个代币投入其简单的支付频道。 （这种可能性在后面的 **5.2.5** 中被利用。）

如果包含的支付频道是不对称的，则需要将两个承诺提交到两个工作链中：如果《内部》简单支付频道的最终结算产生负的最终不平衡 δ，$0 \leq -\delta \leq c$ ，A 将承诺向 B 支付 $-\delta$ 个代币; 如果 δ 为正，则 B 必须承诺向 A 支付 δ。 另一方面，如果包含支付频道是对称的，这可以通过将具有参数 (c, d) 的单个《简单支付频道创建》交易提交到单个支付频道

区块链中来完成（这将冻结属于 A 的 c 个代币），然后由 B 进行特殊的《确认交易》（这将冻结 B 的 d 个代币）。

我们期望内部支付频道非常简单（例如， **5.1.3** 中讨论的简单同步支付频道），为了最小化要提交的默克尔树证明的大小。 在 **5.1.7** 中描述的意义上，外部支付频道必须是《智能的》

5.2 付款频道网络，或《闪电网络》

现在我们准备讨论电报开放网络(TON)付款的《闪电网络》，它可以实现任意两个参与节点之间的即时汇款。

5.2.1. 支付频道的限制。 支付频道对于期望在他们之间进行大量汇款的各方非常有用。但是，如果只需要向特定收件人转账一次或两次，那么与她建立支付频道将是不切实际的。 除此之外，这意味着冻结支付频道中的大量资金，并且无论如何都需要至少两个区块链交易。

5.2.2. 支付频道网络，或《闪电网络》。 支付频道网络通过支付频道链接实现资金转移，克服了支付频道的限制。 如果 A 想要将资金转移到 E，她不需要与 E 建立支付频道。通过几个中间节点（例如，四个支付频道，从 A 到 B，从 B 到 C，从 C 到 D，从 D 到 E）连接 A 和 E 的支付频道就足够了。

5.2.3. 支付频道网络概述。 回想一下，支付频道网络，也称为《闪电网络》，由一组参与节点组成，其中一些节点已在它们之间建立了长期支付频道。 我们将在等一下看到这些支付频道必须是 **5.1.7** 意义上的《智能》。 当参与节点 A 想要将钱转移到任何其他参与节点 E 时，她试图找到在支付频道网络内链接 A 到 E 的路径。 当找到这样的路径时，她沿着这条路径进行《连锁货币转账》。

5.2.4. 连锁货币转账。 假设从 A到 B，从 B 到 C，从 C 到 D，从 D 到 E 都有一系列支付频道。此外，假设 A 想要将 x 个代币转移到 E。

一种简单的方法是沿着现有的支付频道将 x 个代币转移到 B，并要求他将钱进一步转发给 C。但是，也不一定知道为什么 B 不会把那钱自己吞并。 因此，必须采用更复杂的方法，不要求所有相关方相互信任。

这可以如下实现。 A 生成一个大的随机数 u 并计算其哈希 $v = \mathrm{HASH}(u)$。 然后她创建了一个承诺，如果有一个带有哈希 v 的数字 u（参见 **5.1.7** ），在她的支付渠道中有

B，则向*B*支付 x 币。这个承诺包含 *v*，但不是 *u*，它仍然保密。

之后，*B* 在其支付频道中创建了与 *C* 类似的承诺。 他并不害怕给出这样的承诺，因为他知道 *A* 给他的类似承诺的存在。如果 *C* 提出哈希问题的解答来收集 *B* 承诺的 *x* 个代币，那么 *B* 将立即提交这个解答给 *A* 来从 *A* 收集 *x* 个代币。

然后创建 *C* 到 *D* 和 *D* 到 *E* 的类似承诺。 当承诺全部到位时，*A* 通过将解答 *u* 传达给所有相关方——或者只是向 *E* 传达，来触发转移。

在本说明书中省略了一些细节。 例如，这些承诺必须具有不同的到期时间，并且承诺的金额可能在链条上略有不同（*B* 可能只承诺向 *C* 提供*x* - ∈ 硬币，其中 ∈ 是预先约定的小额转运费）。 我们暂时忽略这些细节，因为它们与理解支付频道如何运作以及如何在电报开放网络(TON)中实施这些细节并不太相关。

5.2.5. 支付频道链中的虚拟支付频道。 现在假设 *A* 和 *E* 期望彼此支付很多钱。 他们可能会在区块链中创建一个新的支付频道，但这仍然非常昂贵，因为有些资金会被锁定在这个支付频道中。 另一种选择是在每次付款时使用 **5.2.4** 中描述的连锁货币转账。 但是，这将涉及在所有支付频道的虚拟区块链中的大量网络活动以及大量交易。

另一种方法是在支付频道网络中链接 *A* 到 *E* 的链内创建虚拟支付频道。为此，*A* 和 *E* 为他们的付款创建（虚拟）区块链，就像他们要在区块链中创建支付频道一样。然而，他们不是在区块链中创建支付频道智能合约，而是要求所有中间支付频道—— *A* 到 *B*，*B* 到 *C* 等的连接——在其中创建简单的支付频道，绑定到 *A* 创建的虚拟区块链和 *E*（参见 **5.1.10** ）。换句话说，现在根据 *A* 和 *E* 之间的最终结算转移资金的承诺存在于每个中间支付频道内。

如果虚拟支付频道是单向的，那么这种承诺可以很容易地实施，因为最终的不平衡 δ 将是非正数，因此可以按照 **5.2.4** 中描述的相同顺序在中间支付频道内创建简单的支付频道。他们的到期时间也可以相同的方式设置。

如果虚拟支付频道是双向的，情况会稍微复杂一些。 在这种情况下，应该把转移 δ 个代币的承诺分开，根据最终结算分为两个半承诺，如 **5.1.10** 所述：往正向转移 δ - = max (0, δ) 硬币，并往反向转移 δ + = max (0, δ) 。 这些半承诺可以在中间支付频道中独立创建，一个半承诺链从 *A* 到 *E*，另一个链在相反方向。

5.2.6. 寻找闪电网络中的路径。到目前为止，有一点仍未被讨论：*A* 和 *E* 将如何在支付网络中找到连接它们的路径? 如果支付网络不是太大，则可以使用类似开放式最短路径优先的协议：支付网络的所有节点创建覆盖网络（参见 **3.3.17** ），然后每个节点传播所有可用链路（即，参与的支付频道） 信息，通过流言协议向其邻居提供。最终，所有节点都将拥有参与支付网络的所有支付频道的完整列表，并且能够自己找到最短路径——例

如，通过应用修改的戴克斯特拉算法版本来考虑所涉及的支付频道的《容量》（即，可以沿着它们转移的最大金额）。一旦找到候选路径，就可以通过包含完整路径的特殊抽象数据报网络层数据报进行探测，并要求每个中间节点确认所讨论的支付频道的存在，并根据路径进一步转发该数据报。之后，可以构建链，并且可以运行链转移协议（参见 **5.2.4** ），或者用于在支付链频道内创建虚拟支付频道的协议（参见 **5.2.5** ）。

5.2.7. 优化。 可以在这里进行一些优化。 例如，只有闪电网络的传输节点需要参与 **5.2.6** 中讨论的类似开放式最短路径优先的协议。 希望通过闪电网络连接的两个《叶子》节点将彼此传达他们所连接的传输节点的列表（即，他们已经建立了参与支付网络的支付频道）。 然后，可以检查从一个列表的传输节点连接到另一个列表的传输节点的路径，如上面 **5.2.6** 中所述。

5.2.8. 结论。 我们已经概述了电报开放网络(TON)项目的区块链和网络技术如何足以完成创建TON付款的任务，TON付款是一个脱链即时汇款和小额支付的平台。 该平台对于驻留在TON生态系统中的服务非常有用，使他们可以在需要时轻松地收集小额支付

结论

我们提出了一种可扩展的多区块链架构，能够通过用戶友好的界面支持大规模受欢迎的加密货币和分散式应用程序。

为了实现必要的可扩展性，我们提出了电报开放网络(TON)区块链，一种《紧耦合》多区块链系统（参见 **2.8.14** ），采用从下到上的分片方法（参见 **2.8.12** 和 **2.1.2** ）。为了进一步提高潜在性能，我们引入了两区块链机制来替换无效块（参见 **2.1.17** ）和即时超立方体路由，以便更快地在分片之间进行通信（参见 **2.4.20** ）。 将电报开放网络(TON)区块链与现有和被提议的区块链项目（参见 **2.8** 和 **2.9** ）进行简要比较，显出了这种方法对于寻求每秒处理数百万笔交易的系统的好处。

第三章中描述的电报开放网络(TON)包括了被提议的多区块链基础设施的网络需求。 该网络组件也可以与区块链结合使用，以创建广泛的应用和服务，单独使用区块链是不可能的（参见 **2.9.13** ）。 第四章讨论的这些服务包括电报开放网络(TON)域名系统，一种将人们可读对象标识符转换为其地址的服务; TON存储，一个用于存储任意文件的分布式平台; TON代理，一种匿名网络进接和进接TON支持服务的服务; 和TON付款（参见第 **5** 章），一个跨电报开放网络(TON)生态系统即时脱链资金转移的平台，应用程序可用于小额支付。

电报开放网络(TON)基础设施允许专门的轻客戶端钱包和《ton浏览器》桌面和智能手机应用程序，为最终用戶提供类似浏览器的体验（参见 **4.3.24** ），进行加密货币支付以及与智能合约和其他服务的互动。 电报开放网络(TON)平台可供大众用戶访问。 这样的轻型客戶端可以集成到电报信使客戶端（参见 **4.3.19** ），从而最终为数亿用戶带来了大量基于区块链的应用程序。

引用

[1] L. 拜尔德，Swirlds Hashmap一致性算法：公平，快速，拜占庭容错，Swirlds技术报告 SWIRLDS-TR-2016-01, https://www.swirlds.com/downloads/SWIRLDS-TR-2016-01.pdf , 2016.

[2] P. 巴莱托，M. 纳里格，优质种类的配对友好椭圆曲线，在: Preneel, B., Tavares, S. (Eds.) SAC 2005. LNCS vol. 3897, 第319–331页. Springer, 2006. 可在https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/pfcpo.pdf. 上找到

[3] K. 伯尔曼，可靠的分布式系统：技术，网路服务和应用，Springer，2005。

[4] A. 博勒迪雷娃，基于Gap-迪菲-赫尔曼-组签名方案的门限签名，多重签名和盲签名，在国际公钥密码学理论与实践研讨会（PKC）2003年会议录，LNCS vol. 2567，第31-46页，Springer，2003 。

[5] V. 布特林，以太坊：下一代智能合约和分散式应用平台，https://github.com/ethereum/wiki/wiki/White-Paper , 2013.

[6] M. 本-奥，B. 凯尔默, T. 拉宾，具有最佳弹性的异步安全计算，在第13届年度ACM分布式计算原理研讨会论文集中，第183–192页. ACM, 1994.

[7] M. 卡斯特罗，B. 里斯可夫，等等，实用的拜占庭容错，第三届操作系统设计与实现研讨会论文集（1999），第173–186页, 可在 http://pmg.csail.mit.edu/papers/osdi99.pdf 上找到.

[8] EOS.IO，EOS.IO 技术白皮书，https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md , 2017.

[9] R. 根纳罗，C. 詹特里，B. 帕尔诺，M. 雷克娃，二次跨度程序和简洁的零知识没有概率可检测证明。 在密码学进展--Eurocrypt 2013，第626–645页, 2013.

[10] R. 格纳罗，S. 亚雷吉，H. 克罗日克，T. 拉宾，基于离散对数的密码系统的安全分布式密钥生成，Eurocrypt 99, 1999.

[11] D. 高德士拉格，M. 里德，P. 赛瓦森，匿名和私人互联网连接的洋葱路由，ACM的通信，42, num. 2 (1999), http://www.onion-router.net/Publications/CACM-1999.pdf .

[12] F. 海斯，N. 斯马特，F. 瓦高特伦，重新参考了Eta配对，可在 https://eprint.iacr.org/2006/110.pdf 上找到。

[13] L. 郎波特，R. 肖斯塔克，M. 皮斯，拜占庭将军问题，ACM编程语言和系统交易，4/3 (1982), 第382–401页.

[14] S. 拉里默，比特股的历史，https://docs.bitshares.org/bitshares/history.html , 2013.

[15] M. 路比，A. 肖可拉西，等等，用于物体传递的RaptorQ前向纠错方案，IETF RFC 6330, https://tools.ietf.org/html/rfc6330 , 2011.

[16] P. 麦木克夫，D. 马吉艾，Kademlia：基于XOR度量的点对点信息系统，在第一届对等系统国际研讨会的IPTPS '01修订文件中，第53–65页, 可在http://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf 获取，2002.

[17] P. 麦木克夫，在线代码（扩展摘要），纽约大学计算机科学，TR2002-833, https://cs.nyu.edu/media/publications/TR2002-833.pdf , 2002.

[18] A. 米勒，Yu Xia, 等等，拜占庭容错协议的蜜獾，密码学电子印刷档案 2016/99，https://eprint.iacr.org/2016/199.pdf , 2016.

[19] S. 中本，比特币：点对点电子现金系统，https://bitcoin.org/bitcoin.pdf , 2008.

[20] S. 佩顿 章斯，在库存硬件上实现懒惰的功能语言：无骨架无标签G机，功能编程杂志 **2** (2), 第127–202页, 1992.

[21] A. 肖可拉西，M. 路比，猛禽代码，IEEE信息论交易 **6**，no. 3–4 (2006), p. 212–322.

[22] M. 万 斯丁，A. 塔嫩堡，分布式系统，第三版，2017.

[23] 单基础类型理论，同伦类型理论：数学的单价基础，高等研究院，2013年，可在 https://homotopytypetheory.org/book 获取。

[24] G. 乌德，波卡：异构多链框架的愿景，草案1，https://github.com/w3f/polkadot-white-paper/raw/master/PolkaDotPaper.pdf , 2016.

A. 电报开放网络(TON)代币，或Gram

电报开放网络(TON)区块链的主要加密货币，特别是其主链和基本工作链，是TON代币，也称为Gram (GRM)。 它用于存入成为验证者所需的存款; 交易费，汽油支付（即智能合同消息处理费）和持久存储支付通常也在Gram 中收集。

A.1. 细分和术语。 Gram 被细分为十亿 (10^9) 个较小的单位，称为纳Gram ，*ngram*或简称纳。 所有转账和账户余额均表示为纳的非负整数倍。 其他单位包括：

- 纳，*ngram*或纳Gram是最小的单位，等于10^{-9} Gram 。
- 微或微Gram等于一千 (10^3) 纳。
- 毫是一百万 (10^6) 纳，或千分之一 (10^{-3})Gram。
- 一Gram等于十亿 (10^9) 纳。
- 一公Gram，或kGram，等于一千 (10^3) Gram。
- 兆Gram（或MGram）等于一百万 (10^6) Gram，即10^{15}纳。
- 最后，一个千兆Gram，或GGram，等于十亿 (10^9) Gram，或10^{18} 纳。

不需要更大的单位，因为Gram的初始供应量将限制在五十亿 ($5 \cdot 10^9$) 克（即五千兆Gram）。

A.2. 用于表示汽油价格的较小单位。 如果出现较小单位的必要性，将使用等于2^{16}纳Gram的《微单位》。 例如，汽油价格可以用微单位表示。 然而，实际支付的费用（按汽油价格和消耗的气油量计算）将始终向下舍入到2^{16}个微单位的最接近的倍数，并表示为整数纳。

A.3. 原始供应，采矿奖励和通货膨胀。 Gram的总供应量最初限制在五千兆Gram（即五十亿克或$5 \cdot 10^{18}$纳）。

随着对采用新的主链和分片链块的验证者的奖励积累，这种供应将缓慢增加。 我

们最初预计，如果验证者勤勉地履行其职责，签署所有区块，永不脱机并且永远不会签署无效区块，那么这些奖励将达到每年验证者权益的大约20%（确切的数字可能在未来调整）。 通过这种方式，验证者将有足够的利润来投资处理不断增长的用戶交易量所需的更好和更快的硬件。

我们预计，在任何特定时刻，Gram总供应量中最多只有10% [40] 将被绑定在验证者权益里。 这将产生每年2%的通货膨胀率，因此，将使三十五年内Gram的总供应量增加一倍（达到一万兆Gram）。 从本质上讲，这种通货膨胀代表了社区所有成员向验证者支付的费用，用于保持系统正常运行。

另一方面，如果验证者行为不端，其部分或全部权益将被作为惩罚，其中较大部分随后将被《烧毁》，从而减少Gram的总供应量。 这可能导致通货紧缩。 罚款的一小部分可以重新分配给验证人或《渔民》，他们提交了有罪验证者不当行为的证明。

A.4. Gram的原价。 要出售的第一个Gram的价格约等于 $0.1（美元）。 随后出售的每一Gram的定价（至少）比前一个高出十亿分之一。 这样，待投入流通的第 n 个Gram将以（不小于）大约出售

$$p(n) \approx 0.1 \bullet (1 + 10^{-9})n \text{ 美元} \tag{26}$$

或大约等量的其他（加密）货币。

A.4.1. 指数价格的加密货币。 我们说Gram是一个指数级的加密货币，这意味着要投入流通的第n个Gram的价格至少是这公式给出的《参考价格》$p(n)$

$$p(n) = p_0 \bullet e\alpha n \tag{27}$$

具体值$p_0 = 0.1$美元和$\alpha = 10^{-9}$。

更确切地说，一旦 n 个代币投入流通，新代币的一小部分 d_n 值（至少）$p(n)d_n$ 美元。（这里 n 不一定是整数。）

这种加密货币的其他重要参数包括 n，流通中的代币总数，以及$N \geq n$，可存在的代币总数。 对于Gram，最初$N = 5 \bullet 10^9$。

A.4.2. 前 n 个代币的总价。 要进入流通的指数价格加密货币（例如，Gram）的前 n 个代币的总参考价格$T(n) = \int_0^n p(n)\, d_n \approx p(0) + p(1) + \ldots + p(n - 1)$可以通过以下方式计算

[40] 验证者权益的最大总量是区块链的可配置参数，因此必要时可以通过协议强制执行此限制。

$$T(n) = p_0 \bullet \alpha^{-1}(e^{\alpha n} - 1) \tag{28}$$

A.4.3. 下一个硬币的总价格。 在 n 个先前存在的硬币之后投入流通的 Δn 个代币的总参考价格 $T(n + \Delta n) - T(n)$ 可通过以下方式计算：

$$T(n + \Delta n) - T(n) = p_0 \bullet \alpha^{-1}(e\alpha^{(n+\Delta n)} - e^{\alpha n}) = p(n) \bullet \alpha^{-1}(e^{\alpha \Delta n} - 1) \tag{29}$$

A.4.4. 购买总价值为 T 的下一批代币。假设 n 个代币已经投入流通，并且想要花费 T（美元）购买新代币。 通过将 $T(n + \Delta n) - T(n) = T$ 置于 (29) 中，可以计算新获得的硬币数量Δn

$$\Delta n = \alpha^{-1} \log(1 + \frac{(T \bullet \alpha)}{p(n)}) \tag{30}$$

假设所有新硬币都以其参考价格出售。

当然，如果 $T \lll p(n)\alpha^{-1}$，那么 $\Delta n \approx T/p(n)$。

A.4.5. Gram的市场价格。 当然，如果自由市场价格低于 $p(n) = p_0 e^{\alpha n} \approx 0.1 \bullet (1 + 10^{-9})^n$ 一旦 n 个Gram 进入流通，就不太可能有人从电报开放网络(TON)储备中购买新的Gram；他们会选择在自由市场上购买他们的Gram，而不会增加Gram流通量的总量。 另一方面，电报开放网络(TON)储备出售Gram的能力可能会限制Gram价格的任何升级。 这意味着Gram的市场价格可能不太容易出现突然的峰值（和下跌）；这很重要，因为权益（验证者存款）被冻结至少一个月，如果汽油价格变化不太快，区块链运行更加顺畅和可预测。

A.4.6. 买回 Gram 。 如果Gram的市场价格低于$0.5 \bullet p(n)$，当流通中总共有 n 个Gram（即没有保留在由电报开放网络(TON)储备控制的特殊账户）时，TON储备保留购回Gram的权利并减少 n，即Gram在流通的总数量。 这可能有助于防止Gram汇率的突然下降。

A.4.7. 以更高的价格出售新的 Gram。电报开放网络(TON)储备保留根本不出售任何剩余的Gram，或以高于 $p(n)$ 的价格出售它们的权利，但从不以较低的价格出售（考虑到快速变化的汇率的不确定性）。

A.5. 使用未分配的 Gram。 TON储备不会使用大部分未分配的Gram（大约 $5 \bullet 10^9 - n$ 个

Gram）——即那些存在TON储备特别帐户中和明确与之相关的其他帐户中的——除了出售之外的任何其他目的，如在 **A.4** ， **A.4.7** 和 **A.6** 中概述，并将先前未分配的Gram的一预定义部分转移为开发者和生态系统激励，如在 **A.5.1** 和 **A.5.2** 中概述的。

A.5.1. 一些未分配的Gram将提供给开发人员。 在电报开放网络(TON)区块链的部署期间，将预定数量的未分配的Gram（例如，两百兆Gram，相当于总供应量的4%）作为向开源电报开放网络(TON)软件的开发者支付的《奖励》进行传输，有四年归属期。

A.5.2. 一些未分配的Gram将被保留用于生态系统激励。 预定数量的Gram（例如，五百兆Gram，相当于初始总供应量的10%）将被保留用于《生态系统激励》，例如，以鼓励安装第三方验证器和电报开放网络(TON)存储和TON代理节点——例如 通过支付它们来存储TON区块链的旧块或代理所选服务子集的网络流量。 另一个激励的例子可以是奖励用户激活他们的电报开放网络(TON)钱包。

A.5.3. 增加电报开放网络(TON)储备参考价格。 在电报开放网络(TON)区块链启动后对TON生态系统（参见 **A.5.2** ）和TON开发者共享资金（参见 **A.5.1** ）进行分配后，Gram的TON储备参考价格 $p(n)$ 将立即生效上升一定数量，可提前计算。 其余未分配的Gram将由TON储备使用，如上文 **A.5** 中所述。

A.6. 批量销售克。 当很多人同时想要购买大量的Gram时，不要立即处理他们的订单是对的，因为这样达到的结果会非常依赖于特定订单的时间和他们的处理顺序。

相反，可以在某个预定义的时间段（例如，一天或一个月）期间收集购买克的订单，然后立即一起处理。 如果具有第 i 个订单价值 T_i 美元的 k 个订单到达，则总金额 $T = T_1 + T_2 + ... + T_k$ 根据 (30) 用于购买 Δn 个新代币，并且分配了第 i 个订单的发送者这些代币的 $\Delta n \bullet T_i / T$. 通过这种方式，所有买家以相同的平均价格每Gram付 $T / \Delta n$ 美元获得他们的Gram。

之后，开始新一轮收购新Gram的订单。

随着电报开放网络(TON)的推出和Gram的初始分发，这个《批量销售》系统预计将被替换为根据公式 (30) 从TON储备直接销售Gram的系统。