Cosmos

分布式账本网络

Jae Kwon jae@tendermint.com

Ethan Buchman ethan@tendermint.com

加入我们的 Slack 一起讨论吧！

注意：我们会对内容进行定期更新，您可以随时进行查阅，谢谢！

目录

介绍

开源的生态系统、去中心化的文件共享、以及公共的加密货币，这一系列技术的成功让人们开始了解到，去中心化互联网协议是可以用来彻底改善社会经济基础架构的。我们见证了专业区块链应用的诞生，比如比特币 [1]（加密货币），Zerocash[2]（私有加密货币），也看到了大众化智能合约平台，比如以太坊[3]，此外还有其他无数针对 EVM（以太坊虚拟机）的分布式应用，如 Augur（预测市场）以及 The DAO [4]（投资俱乐部）。

但是，到目前为止，这些区块链已经暴露了各种缺陷，包括总能量低效、功能不佳或受限、并且缺乏成熟的管理机制。为了扩大比特币交易吞吐量，已经研发了许多诸如隔离见证（Segregated-Witness）[5]和 BitcoinNG[6]这样的解决方案，但是这些垂直扩展方案都因单一物理机容量而受到限制，不然就得损害其可审核性这一特性。闪电网络 [7]可以通过让部分交易完全记录在账本外，来帮助扩大比特币交易额，这个方法非常适合微支付以及隐私保护支付轨道，但是可能无法满足更广泛的扩展需求。

理想的解决方案是在允许多个平行区块链互相操作的同时，保留安全特性。不过事实证明，采用工作量证明很难做到这一点，但也并非不可能。例如合并挖矿可以在完成工作的同时，让母链得以在子链上重复使用。不过这样还是需要通过每个节点，依次对交易进行验证，而且如果母链上大多数哈希力没有积极地对子链进行合并挖矿，那么就很容易遭到攻击。关于可替代区块链网络架构的学术回顾将在辅助材料中呈现，我们会在相关作品中对更多提议及其缺点进行概括。

这里我们要介绍的是 Cosmos，这是一个全新区块链网络架构，能够解决所有问题。Cosmos 是一个涵盖众多独立区块链的网络，叫做“空间”。空间在 Tendermint Core [8]支持下运行，是一个类似实用拜占庭容错的安全共识引擎，兼具高性能、一致性等特点，而且在其严格的分叉责任制保证下，能够防止怀有恶意的参与者做出不当操作。Tendermint Core 的拜占庭容错共识算法，非常适合用来扩展权益证明机制下的公共区块链。

Cosmos 上的第一个空间叫做“Cosmos Hub”（Cosmos 中心）。Cosmos 中心是一种多资产权益证明加密货币网络，它通过简单的管理机制来实现网络的改动与更新。此外，Cosmos 中心还可以通过连接其他空间来实现扩展。

Cosmos 网络的中心及各个空间可以通过区块链间通信（IBC）协议进行沟通，这种协议就是针对区块链的虚拟用户数据报协议（UDP）或者传输控制协议（TCP）。代币可以安全快速地

从一个空间传递到另一个空间，两者之间无需体现汇兑流动性。相反，空间内部所有代币的转移都会通过 Cosmos 中心，它会记录每个空间所持有的代币总量。这个中心会将每个空间与其他故障空间隔离开。因为每个人都将新空间连接到 Cosmos 中心，所以空间今后也可以兼容新的区块链技术。

Tendermint

这一部分将对 Tendermint 共识协议及其用来创建应用程序的界面进行介绍。更多细节，详见附录。

验证人

在经典拜占庭容错（BFT）算法中，每个节点都同样重要。在 Tendermint 网络里，节点的投票权不能为负，而拥有投票权的节点被称作“验证人”。验证人通过传播加密签名或选票，来参与共识协议并商定下一区块。

验证人的投票权是一开始就确定好的，或者根据应用程序由区块链来决定是否有改变。比如，在 Cosmos 中心这种权益证明类应用程序中，投票权可能就是通过绑定为保证金的代币数量来确定的。

注意：像⅔和⅓这样的分数指的是占总投票权的分数，而不是总验证人，除非所有验证人拥有相同币种。而+⅔ 的意思是“超过⅔ ”，⅓+则是“⅓或者更多”的意思。

共识

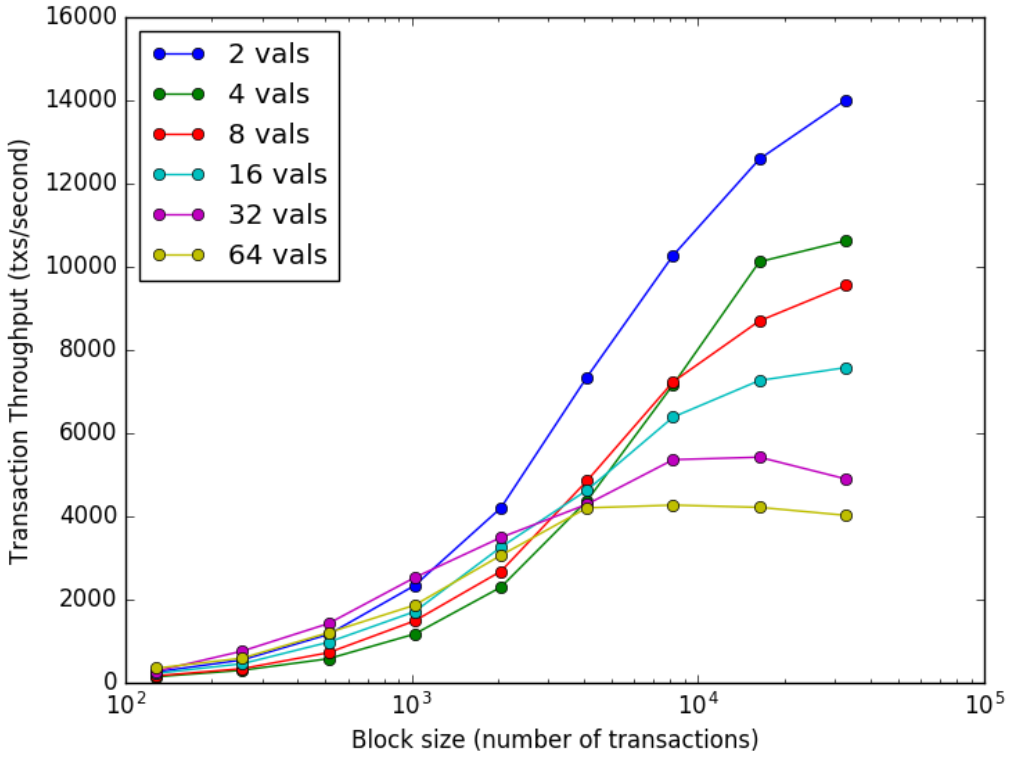
Tendermint 是部分同步运作的拜占庭容错共识协议，这种协议源自 DLS 共识算法[20]。Tendermint 的特点就在于其简易性、高性能以及分叉责任制。协议要求有固定且熟知的一组验证人，其中每个验证人通过公钥进行身份验证。这些验证人会尝试在某个区块上同时达成共识（这里的区块是指一份交易列表）。每个区块的共识轮流进行，每一轮都会有个领头人，或者提议人，由他们来发起区块。之后验证人分阶段对是否接受该区块，或者是否进入下一轮做出投票。每轮的提议人会从验证人顺序列表中按照其选票比例来选择确定。

该协议全部细节请参考此处。

Tendermint 采用由绝对多数的选票（+⅔）选定的最优拜占庭容错算法，以及一套锁定机制来确保安全性。对此他们保证：

- 想要违背安全必须有超过⅓的选票出现拜占庭问题，并且提交超过两个值。
- 如果有任何验证组引起了安全问题，或者说是企图这么做，那么就会被协议发现，一方面针对有冲突的区块进行投票，同时广播那些有问题的选票。

除了其超强安全保障外，Tendermint 还具备其他功效。以商品型云平台为例，Tendermint 共识以分布在五大洲七个数据中心的 64 位节点为基准，其每秒可以处理成千上万笔交易，提交顺序延迟时间为 1-2 秒。而值得关注的是，即使是在极其恶劣的敌对环境中，比如验证人崩溃了或者是遇到蓄谋已久的恶意选票，也能维持这种每秒千笔交易的高绩效。详见下图。



轻客户端

Tendermint 共识算法的主要好处就是它具有安全简易的轻客戶端，这一点使其成为手机和物联网用例的理想工具。比特币轻客户端必须同步运行区块头组成的链，并且找到工作量证明最

多的那一条，而 Tendermint 轻客戶端只需和验证组的变化保持一致，然后简单地验证最新区块中预先提交的+⅔，来确定最新情况。

这种简单的轻客戶端证明机制也可以实现区块链之间的通信。

防止攻击

Tendermint 有各种各样的防御措施来防止攻击，比如远程无利害关系双重花费及审查制度。这个在附录中会进行完整讨论。

TMSP

Tendermint 共识算法是在叫做 Tendermint Core 的程序中实现的。这个程序是一种与应用程序无关的“共识引擎”，可以让任何命中注定的黑匣子软件变为分散复制的区块链。就像 Apache 网页服务器或者 Nginx 是通过通用网关接口（CGI）或快速通用网关接口（FastCGI）来连接 Wordpress（一款博客系统）应用程序一样，Tendermint Core 通过 Tendermint Socket 协议（TMSP）来连接区块链应用程序。因此，TMSP 允许区块链应用程序用任何语言进行编程，而不仅仅是共识引擎写入的程序语言。此外，TMSP 也让交换任何现有区块链堆栈的共识层成为可能。

我们将其与知名加密货币比特币进行了类比。在比特币这种加密币区块链中，每个节点都维持着完整的审核过的 UTXO（未使用交易输出）数据库。如果您想要在 TMSP 基础上，创建出类似比特币的系统，那么 Tendermint Core 可以做到：

- 在节点间共享区块及交易
- 创建规范或不可改变的交易顺序（区块链）

同时，TMSP 应用程序会负责：

- 维护 UTXO 数据库
- 验证交易的加密签名
- 防止出现不存在的交易花费
- 允许客户访问 UTXO 数据库

Tendermint 能够通过为应用程序与共识的形成过程，提供简单的应用程序界面（API），来分解区块设计。

Cosmos 概述

Cosmos 是一种独立平行的区块链网络，其中每条区块链通过 Tendermint 1 这样的经典拜占庭容错共识算法来运行。

网络中第一条区块链将会是 Cosmos 中心。Cosmos 中心通过全新区块链间通信协议来连接其他众多区块链（或将其称之为空间）。中心可以追踪无数代币种类，并且在各个连接的空间里记录代币总数。代币可以安全快速地从一个空间传递到另一个空间，两者之间无需体现汇兑流动性，因为所有空间之间的代币传输都会经过 Cosmos 中心。

这一架构解决了当今区块链领域面临的许多问题，包括应用程序互操作性、可扩展性、以及无缝更新性。比如，从 Bitcoind、Go-Ethereum、CryptoNote、ZCash 或其他区块链系统中衍生出来的空间，都可以接入 Cosmos 中心。这些空间允许 Cosmos 实现无限扩展，从而满足全球交易的需求。此外，空间也完全适用于分布式交易所，反之交易所也支持空间运行。

Cosmos 不仅仅是单一的分布式账本，而 Cosmos 中心也不是封闭式花园或宇宙中心。我们正在为分布式账本的开放网络设计一套协议，这套协议会按照加密学、稳健经济学、共识理论、透明性及可追究制的原则，成为未来金融系统的全新基础。

Tendermint 拜占庭容错股份授权证明机制 (Tendermint-BFT DPoS)

Cosmos 中心是 Cosmos 网络中第一个公共区块链，通过 Tendermint 拜占庭共识算法运行。这个 Tendermint 开源项目于 2014 年开始，旨在解决比特币工作量证明算法的速度、可扩展性以及环境问题。通过采用并提高已经过验证的拜占庭算法（1988 年在麻省理工学院开发），Tendermint 成为了首个在概念上演示加密货币权益证明的团队，这种机制可以解决 NXT 和 BitShares 这些第一代权益证明加密币面临的“无利害关系”（nothing-at-stake）的问题。

如今，实际上所有比特币移动钱包都要使用可靠的服务器来进行交易验证。这是因为工作量证明机制需要在交易被认定为无法逆转前进行多次确认。而在 CoinBase 之类的服务中也已经出现重复花费攻击。

和其他区块链共识系统不同，Tendermint 提供的是即时、可证明安全的移动客户端支付验证方式。因为 Tendermint 的设计完全不支持分叉，所以移动钱包就可以实时接收交易确认，从而在智能手机上真正实现去信任的支付方式。这一点也大大影响了物联网应用程序。

Cosmos 中的验证人（其扮演的角色类似比特币矿工，但是与之不同的是，他们采用加密签名来进行投票）必须是专门用来提交区块的安全机器。非验证人可以将权益代币（也叫做“atom”）委托给任何验证人来赚取一定的区块费用以及 atom 奖励，但是如果验证人被黑客攻击或者违反协议规定，那么就会面临被惩罚（削减）的风险。Tendermint 拜占庭共识的可证明安全机制，以及利益相关方（验证人和委托人）的抵押品保证，为节点甚至是轻客户端提供了可证明、可计量的安全性。

管理

分布式公共账本应该要有一套章程与管理体系。比特币依靠比特币基金会（在一定程度上）及挖矿来协调更新，但是这个过程很缓慢。以太坊在采用硬分叉措施解决 The DAO 黑客事件后，分裂成了 ETH 和 ETC，这主要是因为之前设定社会契约或机制来进行这类决定。

Cosmos 中心的验证人与委托人可以对提案进行投票，从而自动改变预先设置好的系统参数（比如区块容量限制），协调更新，并对人们看得懂的章程进行修订投票，从而管理 Cosmos 中心。这个章程允许权益相关者聚集到一起，来解决盗窃及漏洞等相关问题（比如 The DAO 事件），并快速得出明确的解决方案。

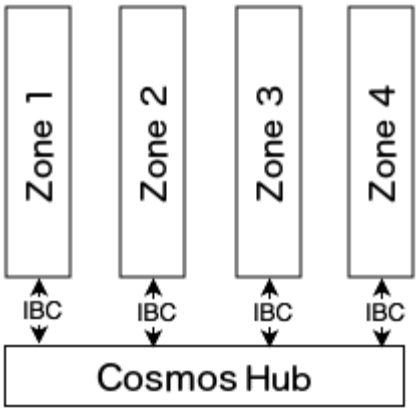
每个空间也具备自己的一套章程及管理机制。比如，Cosmos 中心的章程会强制实现中心的不可改变性（不能重新执行，除了 Cosmos 中心节点实现的漏洞），而每个空间则可自行设置与盗窃及漏洞相关的重新执行政策。

Cosmos 网络能够在政策不同的区块间实现互操作性，这一点可以让客户在无需许可的环境下进行实验，为客户带去了终极自由及潜力。

中心与空间

这里我们将描述一个全新的去中心化与可扩展性模型。Cosmos 网络通过 Tendermint 机制来运行众多区块链。虽然现存提案的目标是创建一个包含全球所有交易顺序的“单一区块链”，Cosmos 允许众多区块链在相互运行的同时，维持互操作性。

在这个基础上，Cosmos 中心负责管理众多独立区块链（称之为“空间”，有时也叫做“碎片”，根据数据库扩展技术“分片”得出）。中心上的空间会源源不断地提交最新区块，这一点可以让中心跟上每个空间状态的变化。同样地，每个空间也会和中心的状态保持一致（不过空间之间不会同彼此的步伐保持一致，除非间接通过中心来实现）。之后信息包就会从一个空间传递到另一个空间，并通过发布梅克尔证明（Merkle-proof）来说明信息已经被传送或接收。这种机制叫做“区块链间通信”，或者简称为“IBC”机制。



任何区块都可以自行成为中心，从而形成非循环图，但是有一点需要阐明，那就是我们只会对简单配置（只有一个中心）以及许多没有中心的空间进行描述。

中心（Hub）

Cosmos 中心区块链承载的是多资产分布式账本，其中代币可以由个体用户或空间本身持有。这些代币能够通过特殊的 IBC 包裹，即“代币包”（coin packet）从一个空间转移到另一个空间。中心负责保持空间中各类代币全球总量不变。IBC 代币宝交易必须由发送人、中心及接收人的区块链执行。

因为 Cosmos 中心在整个系统中扮演着中央代币账本的角色，其安全性极其重要。虽然每个空间可能都是一个 Tendermint 区块链——只需通过 4 个，或者在无需拜占庭容错共识的情况下更少的验证人来保证安全），但是 Cosmos 中心必须通过全球去中心化验证组来保证安全，而且这个验证组要能够承受最严重的攻击，比如大陆网络分割或者由国家发起的攻击。

空间（Zones）

Cosmos 空间是独立的区块链，能够和 Cosmos 中心进行 IBC 信息交换。从 Cosmos 中心的角度看，空间是一种多资产、多签名的动态会员制账户，它可以通过 IBC 包裹进行代币发送与接收。就像加密币账户一样，空间不能转移超出其持有量的代币，不过可以从其他拥有代币的人那里接收代币。空间可能会被指定为一种或多种代币的“来源”，从而赋予其增加代币供应量的权力。

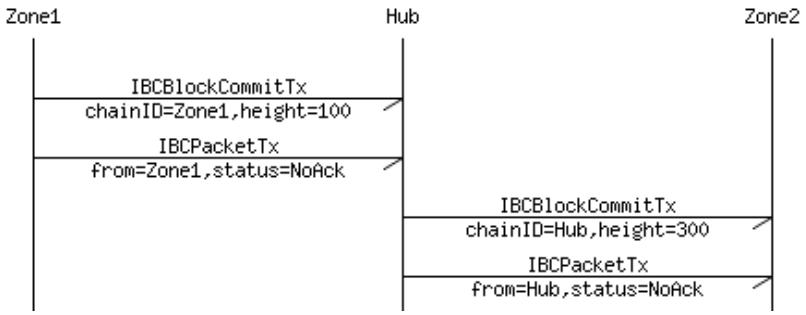
Cosmos 中心的 Atom 或可作为空间（连接到中心）验证人的筹码。虽然在 Tendermint 分叉责任制下，空间出现重复花费攻击会导致 atom 数量减少，但是如果空间中有超过⅔的选票都出现拜占庭问题的话，那这个空间就可以提交无效状态。Cosmos 中心不会验证或执行提交到其他空间的交易，因此将代币传送到可靠空间就是用户的责任了。未来 Cosmos 中心的管理系统可能会通过改善提案，来解决空间故障问题。比如，在检测到袭击时，可以将有些空间（或全部空间）发起的代币转移输出压制下来，实现紧急断路（即暂时中止代币转移）。

区块链间通信（IBC）

现在我们来介绍下中心与空间之前通信的方法。假如现在有三个区块链，分别是“空间 1”、“空间 2”以及“中心”，我们想要“空间 1”生成一个包裹，通过“中心”发送给“空间 2”。为了让包裹从一个区块链转移到另一个区块链，需要在接收方区块链上发布一个证明，来明确发送方已经发起了一个包裹到指定地点。接收方要验证的这个证明，必须和发送方区块头保持一致。这种机制就类似与侧链采用的机制，它需要两个相互作用的链，通过双向传送存在证明数据元（交易），来“知晓”另一方的情况。

IBC 协议可以自然定义为两种交易的使用：一种是 `IBCBlockCommitTx` 交易，这种交易可以让区块链向任何观察员证明其最新区块哈希值；另一种是 `IBCPacketTx` 交易，这种交易则可以证明某个包裹确实由发送者的应用程序，通过梅克尔证明机制（Merkle-proof）传送到了最新区块的哈希值上。

通过将 IBC 机制分裂成两个单独的交易，即 `IBCBlockCommitTx` 交易与 `IBCPacketTx` 交易，我们可以让接收链的本地费用市场机制，来决定承认哪个包裹，与此同时还能确保发送方的完全自由，让其自行决定能够传出的包裹数量。



在上述案例中，为了更新“中心”上“空间 1”的区块哈希（或者说“空间 2”上“中心”的区块哈希），必须将 `IBCBlockCommitTx` 交易的“空间 1”区块哈希值发布到“中心”上（或者将该交易的“中心”区块哈希值发布到“空间 2”中）。

更多关于两种 IBC 交易的信息，请参考 IBCBlockCommitTx 以及 IBCPacketTx。

用例

分布式交易所

比特币借助批量复制的分布式账本技术来保证安全，同样的，我们也可以用这种方式，在区块链上运行，从而降低交易所受内外部攻击的可能性。我们称之为分布式交易所。

如今，加密币社区认为去中心化交易所是基于“原子交叉链”交易（AXC 交易）的交易所。通过这类交易，不同链上的两位用户可以发起两笔传输交易，要么在两个账本上一起执行，要么两个账本都不执行（即原子级）。比如，两位用户可以通过 AXC 交易来进行比特币和以太币之间的交易（或不同账本上的任意两种代币），即使比特币和以太坊之间并没有相互连接。在 AXC 交易模式下的交易所，其好处在于用户双方都不需要相信彼此，也不用相信交易匹配服务。其坏处就是，双方都得在线才能进行交易。

另一种去中心化交易所是在交易所的区块链上运行批量复制的分布式账本。这种交易所的用户可以提交一份限价订单，在关机状态下执行交易。区块链会代表交易者匹配并完成交易。

去中心化交易所可以创建一份大范围限价订单簿，以此来吸引其他交易者。在交易所界，流动性需求越来越高，因此交易所业务界的网络效应也愈发强烈（或者说至少产生了“胜者得益”效应）。目前加密币交易所排名第一的是 Poloniex，其 24 小时交易额为 2000 万美元，而 Bitfinex 以 24 小时 500 万位列第二。在这种强大的网络效应背景下，基于 AXC 的去中心化交易所的交

易额是不可能超过中心化交易所的。去中心化交易所要想和中心化交易所一争高下，那么就需要支持大范围限价订单簿的运行。而只有基于区块链的去中心化交易所可以实现这一点。

Tendermint 的快速交易执行是另一大优势。Cosmos 的空间可以在不牺牲一致性的前提下，通过优先完善快速交易，来实现交易的快速完成——针对双向订单交易，及 IBC（跨区块链通信）代币与其他空间的交易。

根据如今加密币交易所的情况，Cosmos 的一项重大应用就是分布式交易所（也就是 Cosmos DEX）。其交易吞吐能力及提交延时情况可以和那些中心化交易所媲美。交易者可以在离线的状态下提交限价订单。并且，在 Tendermint，Cosmos 中心以及 IBC 的应用下，交易者可以快速地完成资金在交易所及其他空间的转出转入。

和其他加密货币挂钩

享有特权的空间可以作为和其他加密货币挂钩的代币来源。这种挂钩类似 Cosmos 中心与空间之间的关系，两者都必须及时更新彼此最新的区块链，从而验证代币已经从一方转移到另一方的证明。Cosmos 网络上挂钩的空间要和中心以及其他加密货币保持一致。这种间接挂钩的空间可以维持简单的中心逻辑，并且不用了解其他区块链共识战略（比如比特币工作量证明挖矿机制）。

比如，设置有特定验证组的 Cosmos 空间（可能和中心里的验证组一样）可以作为与以太币挂钩的空间，其中基于 Tendermint Socket 协议（TMSP）的应用（即“挂钩空间”里的）有办法和外部以太坊区块链上的（即“起始点”）挂钩合约交换 IBC 信息。通过这一合约，持币人可以先将以太币发送到以太坊的挂钩合约中，然后再将以太币传送到挂钩空间。挂钩合约接收到以太币后，除非同时从挂钩空间处接收到正确的 IBC 包裹，否则这些以太币是无法提取的。而当挂钩空间接收到 IBC 包裹，并证明以太币已被特定以太坊账户的挂钩合约接收后，挂钩空间就会生成存有余额的相关账户。之后，挂钩空间上的以太币（即“已挂钩的以太币”）就可以转进或转出中心了，完成传送到特定以太坊提取地址的交易后，再彻底删除。IBC 包裹可以证明挂钩空间上的交易，这个包裹可以公布到以太坊挂钩合约中，来开放以太币的提取权。

当然，这类挂钩合约也存在风险，比如会出现恶劣的验证组。如果拜占庭投票权超过⅓，就会造成分叉，即从以太坊挂钩合约中提取以太币的同时，还能保持挂钩空间中的挂钩以太币不变。更有甚者，如果拜占庭投票权超过⅔，可能会有人直接对将以太币发送到挂钩合约中（通过脱离原始挂钩空间的挂钩逻辑）的人下手，盗取他们的以太币。

如果将这个挂钩方法完全设计成责任制，那么就有可能解决这一问题。比如，中心及起始点的全部 IBC 包裹可能需要先通过挂钩空间的认可，即让中心或起始点中的钩挂合约对挂钩空间的所有状态转变进行有效验证。中心及起始点要允许挂钩空间的验证人提供抵押品，而挂钩合约的代币转出需要有所延迟（并且抵押品解绑时间也要足够长），从而让单独的审计人有时间发起挑战。我们会以未来 Cosmos 改善提议的形式公开这一系统的设计说明及实现方式，以待 Cosmos 中心的管理系统审批通过。

虽然现在的社会政治环境还不够成熟，不过我们可以做一些延伸，比如让负责国家国币的一些机构（尤其是其银行）组成一个验证组，来实现空间同国家法定货币的挂钩。当然这必须布置好额外的预防措施，只接受法律系统下的货币，从而加强可靠的公证人或大型机构对银行活动的审计。

这一整合或可让空间中所有拥有银行账户的人，将自己银行账户里的美元传输到空间账户中，或者完整的转入中心或其他空间里。

这么看来，Cosmos 中心就是法定货币和加密货币无缝对接的导管，从而解决困扰交易所至今的交互局限问题。

以太坊的扩展

扩展问题一直是以太坊的一个公开问题。目前以太坊节点会处理每笔交易，并且存储所有状态。

因为 Tendermint 提交区块的速度比以太坊工作量证明要快，所以由 Tendermint 共识推动且用于挂钩以太币运行的 EVM（以太坊虚拟机）空间能够强以太坊区块链的性能。此外，虽然 Cosmos 中心及 IBC 包裹技术不能实现每秒合约逻辑的任意执行，但是它可以用来协调不同空间里以太坊合约间的代币变动，通过碎片化方式为以代币为中心的以太坊奠定基础。

多应用一体化

Cosmos 空间可以运行任意应用逻辑，这一点在空间运转初期就已经设定好，通过管理可以不断更新。这种灵活度让 Cosmos 空间得以成为其他加密货币的挂钩载体，比如以太坊或比特币，并且它还能和这些区块链的衍生品挂钩，使用同样的代码库，但是验证组及初始分配有所不同。这样一来就可以运行多种现有加密币框架，比如以太坊、Zerocash、比特币、CryptoNote 等等，

将其同 Tendermint Core 结合，成为通用网络中性能更优的共识引擎，为平台提供更多的交互机遇。此外，作为多资产区块链，每笔交易都有可能包含多个输入输出项，其中每个输入项都可以是任意代币，使 Cosmos 直接成为去中心化交易所，当然这里假设的是订单通过其他平台进行匹配。还有一种替代方案，即让空间作为分布式容错交易所（包含订单簿），这可以算是对现有中心化加密币交易所的严格改进——现有交易所时不时会受到攻击。

空间也可以作为区块链版的企业及政府系统，其原本由一个或多个组织运行的特定服务，现在作为 TMSP 应用在某个空间上运行，从而在不放弃对底层服务控制的前提下，维持公共 Cosmos 网络的安全性及交互性。所以，Cosmos 或可为那些既想使用区块链技术，又不愿将控制权彻底放给分布式第三方的人，提供最佳的运行环境。

缓解网络分区问题

有人认为像 Tendermint 这种支持一致性的共识算法有一个重大问题，那就是网络分割会导致没有一个分区拥有超过⅔的投票权（比如超过⅓在线下），而任何这类网络分割都将中止整个共识。而 Cosmos 架构可以缓解这个问题，它可以使用全球中心，但是空间实行地区自治，然后让每个空间的投票权按照正常的地理位置进行分布。比如，某个一般范例就有可能是针对个别城市或地区的，让他们在运行自己空间的同时，还能共享共同的中心（比如 Cosmos 中心），并且可以在因网络分区导致的中断期间，继续维持地区自治活动。请注意，这样一来在设计稳健的联邦式容错系统过程中，就可以真正地去考虑地理、政治及网络拓扑的特征了。

联邦式名称解析系统

NameCoin 是首批试图通过比特币区块链解决名称解析问题的区块链之一。不幸的是，这个方案存在一些不足。

比如，我们可以通过 Namecoin 来验证*@satoshi*（中本聪）这个号是在过去某个时间点用特定公钥进行注册的。但是，该公约是否更新过我们就不得而知了，除非将该名称最后一次更新以来的全部区块都下载下来。这一点是因为比特币 UTXO 交易模式中梅克尔式模型的局限性所导致的，这类模型中只有交易（而非可变的应用程序状态）会以梅克尔形式加入到区块哈希中。它会让我们证明之后名称更新的存在，而非不存在。因此，我们必须依靠完整节点才能明确这个名称的最近价值，否则就要投入巨大成本来下载整个区块链。

即使在 NameCoin 运用了默克尔化的搜索树，其工作量证明的独立性还是会导致轻客戶端的验证出现问题。轻客戶端必须下载区块链中所有区块头的完整复件（或者至少是自其最后的名称更新后的所有区块头）。这意味着带宽需要会随着时间直线扩展。 [21]此外，在工作量证明制区块链上的名称更改需要等额外的工作量证明验证区块才能进行，这个在比特币上可能要花上一个小时。

有了 Tendermint，我们只需用到由法定数量验证人签署（通过投票权）的区块哈希，以及与名称相关的当前价值的默克尔证明。这点让简易、快速、安全的轻客戶端名称价值验证成为可能。

在 Cosmos 中，我们可以借助这个概念对其进行延伸。Cosmos 中的每个名称注册空间都能有一个相关的最高级别域名（TLD），比如“.com”或者“.org”等，每个名称注册空间都有其本身的管理和登记规则。

发行与激励

Atom 代币

Cosmos Hub（Cosmos 中心）是多资产分布式账本，不过它也有本地代币，叫做 Atom。Atom 是 Cosmos Hub 唯一的权益代币。Atom 是持有人投票、验证或委托给其他验证人的许可证，就像以太坊的以太币以太币一样，Atom 也可以用来支付交易费以减少电子垃圾。额外的通胀 Atom 和区块交易费用就作为验证人及委托人（委托给其他验证人）的奖励。

`BurnAtomTx` 交易可以用来恢复储蓄池中任意比例的代币。

众筹

创世块上的 Atom 代币及验证人的初次分布会是 Cosmos 众销资助人占 75%，预售资助人 5%，Cosmos 公司占 20%。从创世块开始，总 Atom 总量的 1/3 将作为奖励发放给每年绑定的验证人以及委托人。

额外细节详见 Crowdfund Plan。

归属

为了防止那些炒股诈骗的投机者借众筹来进行短期牟利，创世块的 Atom 必须有所归属才能用于转移。每个账户将在为期两年的时间里以每小时恒速授予 Atom，这个速率由创世块 Atom 总量除以(2 * 365 * 24)小时得出。通胀区块获得的 Atom 奖励是预先授予的，可以立即进行转移，因此第一年绑定的验证人及委托人可以挣取比其创世块 Atom 一半还多的奖励。

验证人的数量上限

Tendermint 区块链和比特币之类的工作量证明区块链不同，由于通信复杂度提升，验证人增加，所以速度会更慢。所幸的是，我们可以支持足够多的验证人来实现全球稳健的分布式区块链，使其拥有较短交易验证时间，此外，在提升带宽、内存以及平行电脑计算能力的提升下，在未来支持更多验证人的参与。

在创世块诞生那天，验证人数量最多将设置为 100，之后十年的增长率将在 13%，最终达到 300 位验证人。

```
Year 0: 100

Year 1: 113

Year 2: 127

Year 3: 144

Year 4: 163

Year 5: 184

Year 6: 208

Year 7: 235

Year 8: 265

Year 9: 300

Year 10: 300

...
```

成为创世日后首个验证人

如果 Atom 持有人还没有成为验证人，那么可以通过签署提交 BondTx 交易来成为验证人，其中作为抵押品的 Atom 数量不能为零。任何人在任何时候都可以作为验证人，除非当前验证组的数量超过了最大值。这样的话，除非 Atom 数量比最小验证人持有的有效 Atom（包括受委托

的 Atom）还要多，那么交易才算有效。如果新验证人通过这种方式取代了现有验证人，那么现有验证人就被中止活动，所有 Atom 和受委托的 Atom 都会进入解绑状态。

针对验证人的惩罚

针对验证人必须有一定的惩罚机制，防止他们有意无意地偏离已批准的协议。有些证据可以立即采纳，比如在同样高度和回合的双重签名，或者违反“预投票锁定”的（这一规则在 Tendermint 共识协议中有列出）。这类证据将导致验证人损失良好信誉，而且其绑定的 Atom 还有储备池内一定比例的代币份额——合起来称作其“权益”——也会减少。

有时因为地区网络中断、电力故障或者其他原因，验证人会无法连通。如果在过去随便什么时间点的 `ValidatorTimeoutWindow` 区块中，验证人在区块链中提交的投票没有超过 `ValidatorTimeoutMaxAbsent` 次，那么验证人将会被中止活动，并且从权益中共损失一定的验证人超时罚款（`ValidatorTimeoutPenalty`，默认为 1%）。有些劣行表露的没那么明显，这样的话，验证人就可以在带外协调，强制叫停这类恶意验证人，如果有绝对多数制共识的话。

如果 Cosmos 中心因为超过⅓的投票权在线下合并而出现了中止情况，或者说超过⅓的投票权合并来审查进入区块链的恶意行为，这时候中心就必须借助硬分叉重组协议来恢复。（详见“分叉与审查攻击”）

交易费用

Cosmos Hub 验证人可以接受任何中共类的代币或组合作为处理交易的费用。每个验证人可以主观设置任意兑换率，并且选择它想要进行的交易，只要没有超过区块 Gas 限制（BlockGasLimit）。收集起来的费用剪去下面列出的任意税费后，会再次根据权益相关人绑定的 Atom 比例进行分配，周期是就是每次验证人支付的时间（ValidatorPayoutPeriod，默认为 1 小时）。

在所有交易费用中，储存税（ReserveTax，默认为 2%）将存入储备池来增加储备量，来提高 Cosmos 网络的安全性及价值。普通税（CommonsTax，默认为 3%）合并到普通商品的资金中。这些资金将进入托管人地址（CustodianAddress）根据管理熊进行分配。将投票权委托给其他验证人的 Atom 持有人会支付一定佣金给委托方，而这笔费用可以由每个验证人进行设置。

激励黑客

Cosmos Hub 的安全是一组函数，涉及底层验证人的安全以及委托人的委托选择。为了鼓励发现并及时报告缺陷，Cosmos Hub 允许黑客通过 ReportHackTx 交易来“邀功”，主要就是说明，“这个加点已被攻击，请将奖金发到这个地址”。通过这类功绩，验证人和委托人的行为将被中止，而黑客赏金地址可以收到每个人 Atom 中攻击奖励比率（HackRewardRatio，默认为 5%）。而验证人必须通过使用备份密钥来恢复剩余的 Atom。

为了防止这个特征被滥用于转移未授权的 Atom，ReportHackTx（黑客报告交易）前后验证人和委托人手中的两类 Atom 的比例（授权的与未授权的）将保持不变，而黑客的赏金将包含未授权的 Atom，如果有的话。

管理

Cosmos Hub 通过分布式组织来运行，这类组织要求有一套完备的管理机制，从而协调区块链上的各类变动，比如系统变量参数，以及软件更新、规章更改等。

所有验证人对所有提案的投票负责。如果没能及时对提案做出投票，那么验证人就会在一段时间内自动失去活动权利，这段时间叫做缺席惩罚期（`AbsenteeismPenaltyPeriod`，默认为一周）。

委托人自动继承委托验证人的投票权。这一投票可能会被手动覆盖掉。而未绑定的 Atom 是没有投票权的。

每个提案都需要一定的保证金，即最低提案保证金（`MinimumProposalDeposit` ）代币，这个可以是代币组合也可以是更多代币包括 Atom。对每一个提案，投票人可能会投票来取走保证金呢。如果超过一半的投票人选择取走保证金（比如，由于提案是垃圾信息之类），那么保证金就会进去储备池，除非有任何 Atom 被燃烧。

对于每一个提案，投票人可能会投以下选项：

- 同意
- 强烈同意
- 反对
- 强烈反对
- 弃权

决定采纳（或不采纳）提案需要严格的多数投“同意”或“强烈同意”（或者“反对”及“强烈反对”），但是超过 1/3 的人投“强烈反对”或“强烈支持”的话就可以否决大多数人的决定。如果大多数人的票都被否决，那么每个人都会得到惩罚，即损失否决惩罚费用块那一部分钱（`VetoPenaltyFeeBlocks`，默认是一天的区块值，税费除外），而否决大多数决定的那一方也会受到额外的惩罚，即损失否决惩罚 Atom（`VetoPenaltyAtoms`，默认为 0.1%）。

参数改变提案

这里定义的任何参数都可以发生改变，主要在参数改变提案（`ParameterChangeProposal`）的接受范围内。

文本提案

所有其他提案，比如用来更新协议的提案，都会通过通用的文本提案（`TextProposal`）来协调。

路线图

详见计划。

相关工作

过去几年，关于区块链共识及可扩展性已经有过多次创新，这一部分将挑选一些重要的创新进行简短分析。

共识系统

经典拜占庭容错

二十世纪八十年代早期的共识机制中就已经出现了恶意参与者，当时 Leslie Lamport 杜撰了“拜占庭容错”这个词，用来指那些图谋不轨参与者做出的恣意妄为的行径，这个词和“死机故障”相对，死机故障就只是处理过程崩溃而已。早期针对同步网络也探索出了一些解决方案，其信息滞后有一个上限，尽管实际使用是在高度受控的环境下进行的（比如飞机控制器以及原子钟同步的数据中心）。直到九十年代后期，实用拜占庭容错（PBFT）才被引用，作为有效的、部

分同步的共识算法，以容忍处理过程中⅓的恣意行为。PBFT 成为标准算法，繁殖了各种衍生品，包括最近 IBM 用于超级账本中的。

和 PBFT 相比，Tendermint 共识的主要好处在于其有一套经过改善且简化了的底层结构，其中有些是拥抱区块链范例的结果。Tendermint 区块必须按顺序提交，这一点可以消除复杂性，节省与 PBFT 浏览变化相关的通信开支。在 Cosmos 和众多加密币中，如果区块 N 本身没有提交，那么就无需让区块 N+i（i>=*1*）来提交。如果是带宽导致了区块 N 未提交到 Cosmos 空间，那么它就不会帮助使用带宽将选票共享给区块 N+i。如果是网络分区或者线下节点导致的区块 N 未提交，那么 N+i 就无论如何也不会提交。

此外，区块内交易的批量处理可以对应用程序的状态进行默克尔哈希，而不是用 PBFT 检查机制进行周期消化。这可以实现轻客戶端更快的证明交易提交，以及更快的跨区块链通信。

Tendermint Core 中有很多优化项和特点都超过了 PBFT 特定的性能。比如，验证人发起的区块被分割成部分，默克尔化然后散布开来，这种方式可以提高其广播性能（关于启发请查看 LibSwift[19]）。而且，Tendermint Core 不会对点对点连接做任何假设，只要点对点网络仍有微小连接，那么它就能正常运行。

BitShare 委托权益

BitShares [12]不是第一个部署权益证明机制（PoS）的区块链，但是其对 PoS 区块链的研究与采纳做出了巨大的贡献，尤其是这些被认为是“受委托的” PoS。在 BitShares 中，股权持有者选择“见证”以及“委托”，其中“见证”负责下单并提交交易，而“委托”负责协调软件更新与参数变化。尽管 BitShare 在理想环境下的性能很高（100k tx/s，1 秒的滞后），但是它也有可能受到恶意见证施加的重复使用攻击，这类攻击会导致区块链出现分叉而不用受到任何经济惩罚——它的惩罚来自于“没有任何权益”。BitShare 试图通过允许交易查看近期区块哈希来环节这个问题。此外，权益相关者可以每天移除或替代行为不佳的见证，尽管这个对成功的重复使用攻击来说根本不算什么明确的惩罚。

Stellar

Stellar [13]是在 Ripple 的解决方案上创建的，它精炼了联邦拜占庭协议模型，其中参与共识的进程不包括固定且举世闻名的组件。相反，每个处理节点管理一个或多个“仲裁集碎片”（每

一个都组成一组可靠的进程）。Stellar 中的“仲裁集”被定义为一组节点，其中每一节点包含（是一个超集合）至少一个仲裁集碎片，这样就可以达成协议。

机制的安全性依靠假设实现，即假设任意两个仲裁集的交集并非为空，而实现节点的可用性则需要至少一个仲裁集碎片来完整组成正确节点，这个在使用或大或小的仲裁集碎片间可能会产生一组权衡，因为要在不对信任施加重要假设的情况下来进行平衡是很困难的。最终，节点必须以某种办法来选择充足的仲裁集碎片，从而保证有足够多的容错（或任何“完整无损的节点”，大部分文章里得出的结果就依靠这个来决定）。此外，唯一用来确保这类配置的战略是等级制的，且和 BGP 协议相似（边界网关协议），可用于互联网高层 ISP 来创建全球路由表，或者是用在浏览器中管理 TSL（传输层安全）证书，不过这两者都因其不安全性而臭名昭著。

Stellar 文章中对基于 Tendermint 的权益证明系统的批判可以通过以下代币战略加以缓和，其中有一种新的代币叫做“atom”，可以通过发布这一代币来代表未来的费用及奖励。所以，Tendermint 权益证明机制的好处就是其简易性，在相对简易的同时，提供足够多且可证明的安全保障。

BitcoinNG

BitcoinNG 是针对比特币提出来的一种改善，或将允许垂直扩展形式，比如增加区块容量，并且不会带来负面经济影响（一般这类变动都会带来这类影响），比如越小的矿工受到的影响越大。这一改善可以通过将是首项选择从交易广播中分离来实现，即由“微区块”中的工作量证明先选择群首，之后可以对要提交的交易进行广播，直到新的微区块被发现。这个过程会减少赢得工作量证明比赛所必须的带宽需求，让小矿工可以更公平的参与竞争，然后让最后找到微区块的矿工来定期提交交易。

Casper

Casper [16]是针对以太坊提出的一个权益证明共识算法。其最初运行模式是“赌注共识”，理念就是让验证人反复对其认为会提交到区块链的区块下赌注（根据它之前打赌的经验来），最后得出结论。链接。这是 Casper 团队活跃中的研究领域，其挑战就是搭建一套进化稳定的打赌机制。和 Tendermint 相比，Casper 主要的优势就在于提供了“优于一致性的实用性”——其共识无需超过⅔的仲裁选票——可能其代价就是速度慢以及实现复杂。

水平扩展

Interledger 协议

Interledger 协议[14]严格来说不算是可扩展解决方案。它通过一个松散耦合的双边关系网络，为不同账本系统提供了特别的互操作性。比如闪电网络，ILP 的目的就是促进支付，不过是以不同账本类型的支付为主，并且延展了原子交易机制，将哈希锁以及公证人的仲裁集（也叫做原子传输协议）都包括进去。用于维持账本间交易原子数的后面这种机制和 Tendermint 的轻客戶端 SPV 机制相似，因此就可以对 ILP 和 Cosmos/IBC 之间的区别加以解释了，具体如下：

1. 在 ILP 中连接器的公证人不支持成员变动，并且不允许在公证人间进行灵活的权重。而 IBC 是专门为区块链设计的，其验证人可以有不同的权重，而且成员也可以根据区块链进程进行变动。

2. 在闪电网络中，ILP 付款接收人必须在线来发送确认函给发送者。而在 IBC 代币传输过程中，接收人区块链的验证组会负责提供确认，而非接收人。

3. 两者最大的不同就是 ILP 的连接器不充当支付状态的权威方，而在 Cosmos 中，一个中心的验证人就是 IBC 代币传输状态以及每个空间所持代币总量（不是空间每个账户所持代币总量）的权威见证人。这是一个根本性创新，允许代币在空间里进行安全且非对称的传输，而在 Cosmos 中，充当 ILP 连接器的角色是一个持久且安全的区块链账本——Cosmos 中心（Cosmos Hub）。

4. ILP 账本间支付需要由一份交易订单簿做支持，因为其不包括代币从一个账本到另一个账本的非对称传输，而只有价值或市场等值在传输。

侧链

侧链[15]是针对比特币网络扩展提出的一个机制，通过与比特币区块链“挂钩”的可替代区块链实现。侧链可以让比特币有效从区块链转移到侧链及后台，还可以在侧链上进行新功能测试。在 Cosmos Hub 中，侧链和比特币是彼此的轻客戶端，使用 SPV（安全协议验证工具）证明来决定什么时候转移代币到侧链及后台。当然，因为比特币采用工作量证明机制，所以以比特币为中心的侧链也遇到很多工作量证明作为共识算法而带来的问题与风险。此外，这个是比特币多数派解决方案，它并不支持本地代币以及空间内网络拓扑学，而 Cosmos 可以。也就是说，这种双向挂钩的核心机制在原则上是和 Cosmos 网络运用的机制一样的。

以太坊在可扩展方面的努力

以太坊目前正在研究不同的方法来实现以太坊区块链状态的碎片化，从而解决可扩展问题。这些努力的目标就是在共享状态空间中维持当前以太坊虚拟机提供的抽象层。他们同时开展了多项研究。[18][22]

Cosmos vs 以太坊 2.0 Mauve

Cosmos 和以太坊 2.0 Mauve [22]有不同的设计目标。

- Cosmos 专注代币。而 Mauve 是和扩展普通计算相关。
- Cosmos 不一定是以太坊虚拟机，因此即使是不同的虚拟机也可以进行交互。
- Cosmos 可以让空间创建者决定谁来验证空间。
- 任何人都可以在 Cosmos 开创新空间（除非管理有其他决定）。
- 中心会隔离空间故障，这样全球代币的不变性就得到了维护。

普通扩展

闪电网络

闪电网络是一种代币传输网络，在比特币区块链（及其他公共区块链）上一层运行，能够执行众多改善交易吞吐量的指令，通过将大多数交易从共式账本移出，转入所谓的“支付信道”内。这个举措通过链上的加密货币脚本或可实现，它可以让参与者进入双方有状态的合约，其中状态可以通过共享数字签名进行更新，并且最后将证据公布到区块链后可以关闭合约，这个机制最初是通过跨链原子掉期而普及的。通过开放多方支付信道，闪电网络的参与者可以成为他人支付的路由焦点，带领全面连接的支付信道网络，代价就是绑定在支付信道上的资本。

虽然闪电网络可以轻松在多个单独区块链间延伸以通过交易市场进行价值传输，但是它无法用来进行区块链间的非对称代币传输。Cosmos 网络的主要好处就是能够进行这类直接代币传输。也就是说，我们可以期待支付信道与闪电网络在我们的代币传输机制下投入广泛应用，一方面为了节省成本，另一方面也可以保证隐私。

隔离见证

隔离见证是一项比特币改善提议（连接），其目的是将每个区块的交易吞吐量提高 2-3 倍，并且同时保证区块快速同步新节点。这个方案的出色就在于它可以在比特币当前协议的限制下运

行，并且允许进行软分叉更新（也就是其他旧版本的客户端软件在更新后可以继续运行）。Tendermint 是一种新的协议，它没有设计限制，因此可以有不同的扩展选择。首先，Tendermint 采用拜占庭容错循环制算法，这个是在加密签字而非挖矿的基础上进行，可以简单地通过多平行区块链进行水平扩展，而定期频繁的区块提交还可以进行垂直扩展。

附录

分叉问责制

如果遇到超过容错能力以及共识出错的情况下，设计周到的共识协议应该要对此提供一定的保障。这个在经济系统中尤为必要，经济系统中的拜占庭行为可以获取大量资金奖励。而针对上述情况有一个最为重要的保证就是分叉问责制，这个形式下，导致共识出错的进程（也就是导致协议客户端开始接受不同值——即分叉出现）可以被识别出并根据协议规定进行生发，或者也有可能受到法律制裁。当法律系统变得不可靠或者调用代价过大，那么验证人可能要强制支付安全保证金来参与，而一旦检测到恶意行为，那么这些保证金就会被吊销或者减少。[10]

注意这个和比特币有很大区别，由于网络同步及其发现哈希冲突的概率特性，比特币的分叉是定期出现的。在很多案例中，很难将恶意分叉与同步导致的分叉区分开来，所以比特币无法可靠地实施分叉问责制，除了让矿工为孤行区块挖矿支付隐形机会成本。

Tendermint 共识

我们将投票阶段分为预投票及预提交阶段。一个选票可以用于特定区块或 *Nil*。我们把同一轮超过⅔的单个区块的预投票总和称为 *Polka*，把同一轮超过⅔的单个区块的预提交总和称为 *Commit*。如果同一轮针对 Nil 的预提交超过⅔，那么它们就进入下一轮。

注意，协议中严格的决定论会招致较弱的同步假设，因为默认的首项必须被扣除且略过。因此，验证人在对 Nil 进行预投票前会等候一段时间（即 *TimeoutPropose* ，超时提议），而 *TimeoutPropose* 的值会随着每一轮的进行而增加。每一轮剩下的进程是完全同步的，在过程中只有验证人收到超过⅔的网络投票才会进入下一步。在实践中，这么做将需要超级强大的对手无

限阻挠较弱的同步假设（导致共识无法提交区块），而且通过使用每个验证人的 *TimeoutPropose* 随机值还可加大其难度。

另一套约束，或者锁定规则会确保最终在每个高度只提交一个区块，任何试图提交超过一个区块到指定高度的恶意行为都会被识别出来。首先，每个区块的预提交必须正当，并且以 Polka 的形式提交。如果验证人已经在 *R_1* 轮预提交了一个区块，那么我们就认为它们被锁定在了这个区块，然后用于验证 *R_2* 轮新预提交动作的 Polka 必须进入 *R_polka 轮*，其中 *R_1 < R_polka <= R_2*。第二，验证人必须提出并且/或者预投票它们被锁定的区块。这两步合起来，就可以确保验证人不会在没有充足证据证明正当性的前提下进行预提交，并且保证已经完成预提交的验证人不能再为其他东西的预提交贡献证明。这样不但可以保证安全，还能保证共识算法的活跃。

关于这一协议的全部细节参考这里。

Tendermint 轻客戶端

本来需要同步所有区块头，现在在 Tendermint-PoS 里取消了这一需求，因为我们有替代链（分叉），也就是说可以削减超过⅓的绑定权益。当然，削减也是需要有人共享分叉证据的，所以轻客戶端就要存储任何它见证的区块哈希提交。此外，轻客戶端可以定期同步验证组的变动，以避免出现远距离攻击（除了其他可能实现的解决方案）。

秉着和以太坊类似的想法，Tendermint 能够让应用程序在每个区块中嵌入全球梅克尔根哈希，可以进行简单且可验证的状态查询，比如查询账户余额、合约存放价值，或者未使用交易输入的存在等，具体由应用程序的特性决定。

预防远距离攻击

假设有一套充足的可复原的广播网络集合以及一组静态验证组，那么任何区块链分叉都可以被检测到，而且发起攻击的验证人提交的保证金会被扣除。这一创新是由 Vitalik Buterin 于 2014 年首次提出的，可以解决其他权益证明加密货币“没有任何权益”的问题（详见相关工作部分。）但是，由于验证组必须要可以更改，所以可能在很长一段时内最初的验证人可能会解除绑定，因此就可以自由从创世区块中创建新链，并且不需要任何费用，因为他们不再有锁定的保证金。这类攻击被称为远距离攻击（LRA），和短距离攻击相反，短距离攻击是当前处于绑定中的验证人

导致的分叉，并且会因此受到惩罚（假设有类似 Tendermint 共识这样的分叉问责制拜占庭容错算法）。长距离攻击经常被认为是对权益证明机制的重要打击。

所幸的是，LRA 可以通过以下方法得以缓解。第一，针对解绑（来恢复抵押的保证金并且不再挣取参与共识的费用）的验证人，保证金在一定时间内必须是不可转移的，可以称作“解绑期”，可能长达数周或数月。第二，针对轻客戶端的安全，其首次连接到网络必须根据可信源验证最近的一个或者最好是多个区块哈希。这种情况有时也被称作“薄弱主观性”。最后，为了保证安全，必须频繁同步最新验证组（每次间隔时间和解绑期一样）。这一点可以保证轻客戶端在验证人解绑资金（从而没有任何权益）前知道验证组的变化情况，否则解绑的验证人就会通过实施远距离攻击来欺骗客户端，在其绑定的高度创建新区块起始返回点（假设它可以控制足够多的早期私钥）。

注意，用这种方式解决 LRA 问题需要对工作量证明模型原始的安全问题进行彻底检查。在 PoW 中，轻客戶端被认为可以在任何时候从可靠的创世区块同步到当前高度，这个过程可以简单的在每个区块头上处理工作量证明来完成。但是，为了解决 LRA，我们需要轻客戶端上线定期追踪验证组的变动，并且其首次上线必须尤其要注意根据可靠源来验证其从网络收集的情报。当然，后面这个要求和比特币的类似，在比特币中，协议和软件也必须从可靠源获得。

上述预防 LRA 的方法正好适合 Tendermint 驱动下的区块链验证人及全部节点，因为这些节点的任务就是保持连接到网络。这个方法也适合那些想要经常进行网络同步的轻客戶端。但是，对不希望频繁接入互联网或区块链网络的轻客戶端来说，还有另一种方法可以用来解决 LRA 问题。非验证人的代币持有者额可以在很长的解绑期内（比如比验证人的解绑期久）发布代币作为抵押品，并且为轻客戶端提供第二级解决方案来证实当前及过去区块哈希的有效性。就算这些节点没有计入区块链共识的安全性时，他们还是可以为轻客戶端提供强大的保障。如果历史区块哈希查询在以太坊中得到过支持，那么任何人都可以用特定的智能合约来绑定他们的代币，并且提供付费证明服务，从而有效地针对轻客戶端 LRA 安全问题开发出一个市场。

克服分叉及审查攻击

由于区块提交的定义如此，所以任何超过⅓的投票联合都可以通过下线或者不广播选票来中止区块链运行。这样的联合也可以通过拒绝包含这类交易的区块来检查特定交易，尽管这或将导致大多数区块提案被拒绝，从而减缓区块提交速率，降低实用性及价值。恶意联合或许会陆陆续续地广播选票，以阻挠区块链的区块提交，将其逼停，或者就是参与到这些攻击的组合攻击中。最后，它会通过双重签名或者违反锁定规则来造成区块链分叉。

如果一个全球活跃的对手参与进来，那么就会使网络分割，导致验证组子集出现拉低速率。这不单单是 Tendermint 面临的限制，也是所有共识协议（其网络可能由活跃对手控制）的限制。

对这几类攻击，验证人子集应该在外部进行协调，签署重组提议来选择分叉（及牵扯到的任何证据）以及验证人的初始子集。签署这份重组提议的验证人会放弃他们在其他分叉上的所有抵押品。客户端要验证重组提议上的签名，验证任何证据并且做出判断或者给端用户提示来做决定。比如，一个手机钱包 APP 可能会提示用户安全警告，而电冰箱可能会接受签署超过½的初始验证人提出的重组提议。

任何非同步的拜占庭容错算反都不能进入共识，如果超过⅓的投票不诚实的话，而且出现分叉就说明已经有超过⅓的投票权因为无正当理由的双重签名或改锁而失信。因此，签署重组提议是一个协调性问题，无法通过任何非同步协议（即自动的，不对底层网络可靠性做假设的）来解决。目前，我们认为重组提议的协调问题，可以通过互联网媒体的社会共识来实现人为协调。验证人必须确保在签订重组提议前不会出现网络分割，以此来避免有两个相冲突的重组提议被强叔的情况出现。

加入外部条件媒介以及协议足够稳健的话，那么分叉的问题就没有检查攻击问题来的严重。

分叉和检查需要有超过⅓的拜占庭投票权，此外超过⅔的投票权联合可能会恣意提交无效状态。这个是任何拜占庭容错共识系统的特点。和双重签名不同，双重签名会利用简单的可验证的证据来创建分叉，而要检测无效状态的提交则需要非验证对等节点来验证整个区块，也就是说它们会保留一份本地状态副本，执行每笔交易，然后独立计算状态根。一旦检测出来，处理这类故障的唯一方法就是社会共识。比如，如果比特币出现问题，那么无论是软件漏洞造成的分叉，还是矿工拜占庭行为提交的无效状态（正如 2015 年 7 月），连接紧密的商户、开发者、矿工以及其他组织组成的社区按照所需分工来参与修复网络。此外，因为 Tendermint 区块链的验证人或可进行身份认证，所以如果有这个想法，那么无效状态的提交也可能会受到法律或其他外部法律体系的惩罚。

TMSP 具体说明

TMSP（Tendermint Socket 协议）由三个主要信息类型组成，这三类信息从核心传递到应用程序上，然后应用程序用相关回复信息做出应答。

AppendTx 信息是应用程序的主力。区块链上的每笔交易都通过这个信息来传递。应用程序需要借助 AppendTx 信息在当前状态、应用程序协议以及交易加密证书中进行验证，验证每笔接收的交易。验证过的交易之后需要更新应用状态——通过绑定一个值到关键价值存储库中，或者更新 UTXO 数据库。

CheckTx 信息和 AppendTx 信息类似，但是只针对交易验证。Tendermint Core 的内存池会先用 CheckTx 检查交易有效性，并且只会将有效的交易分程传递给对等节点。应用程序可能会检查交易中的递增新鲜值，如果新鲜值过期就会借助 CheckTx 返回一个错误。

Commit 信息是用来计算当前应用状态上的加密提交项的——之后会存入下一区块头。这包含一些方便的特性。状态更新的矛盾性将以区块链分叉的形式出现，从而捕捉整个阶段的程序错误。这也简化了安全轻客戶端的开发，因为梅克尔哈希证明可以通过检查区块哈希来加以验证，而区块哈希是通过验证人仲裁集加以签署的（通过投票权）。

额外的 TMSP 信息允许应用程序追踪改变验证组，并让应用程序接收高度、提交的选票之类的区块信息。

TMSP 的要求/回应是简单的 Protobuf 信息。请参考这里的模式文件。

AppendTx（附加交易）

- **命令行参数**:
  - `Data ([]byte)`: 所需交易的字节
- **返回**:
  - `Code (uint32)`: 回复代码
  - `Data ([]byte)`: 结果字节，如果有的话
  - `Log (string)`: 调试或出错信息
- **使用**:

附加并运行一笔交易。如果交易有效，那返回 CodeType.OK

CheckTx（检查交易）

- **命令行参数**:
  - `Data ([]byte)`: 所需交易的字节

- **返回**:
    - `Code (uint32)`: 回复代码
    - `Data ([]byte)`: 结果字节，如果有的话
    - `Log (string)`: 调试或出错信息
- **使用**:

验证一笔交易。这个信息不应该改变状态。交易在广播给内存池层对等节点前，首先通过 CheckTx 运行。你可以发起半状态化 CheckTx，并在 `Commit` or `BeginBlock` 上清算状态，以允许执行同一区块中相关的交易序列。

Commit（提交）

- **返回**:
    - `Data ([]byte)`: 梅克尔根哈希
    - `Log (string)`: 调试或出错信息
- **使用**:

返回应用程序状态梅克尔根哈希。

Query（查询）

- **命令行参数**:
    - `Data ([]byte)`: 查询需要的字节
- **返回**:
    - `Code (uint32)`: 回复代码
    - `Data ([]byte)`: 查询回复字节
    - `Log (string)`: 调试或出错信息

Flush（划掉）

- **使用**:

划掉回复队列。使用 types.Application 的应用程序无需实施这条信息——这个由项目进行处理。

Info（信息）

- **返回**:
  - `Data ([]byte)`: 信息的字节
- **使用**:

返回关于应用程序状态的信息。Application specific.

SetOption（设置选项）

- **命令行参数**:
  - `Key (string)`: 设置密钥
  - `Value (string)`: 密钥设置的值
- **返回**:
  - `Log (string)`: 调试或出错信息
- **使用**:

设置应用选项。比如，针对内存池的连接可以将密钥设置为“mode”（模式），价值为“mempool”（内存池）。或者针对共识连接，将密钥设置为“mode”，价值设置为“consensus”（共识）。其他选项根据可具体应用进行专门设置。

InitChain（初始链）

- **命令行参数**:
  - `Validators ([]Validator)`: 初始创世验证人
- **使用**:

在创世块生成后进行调用

BeginBlock（起始块）

- **命令行参数**:
  - `Height (uint64)`: 区块刚开始的高度

- **使用**:

  为新区块的开始提供信号。在任何附加交易（AppendTxs）前进行调用。

EndBlock（结束区块）

- **命令行参数**:
  - `Height (uint64)`: 结束时的区块高度
- **返回**:
  - `Validators ([]Validator)`: 具有新选票的变动后的验证人(归零就去除）
- **使用**:

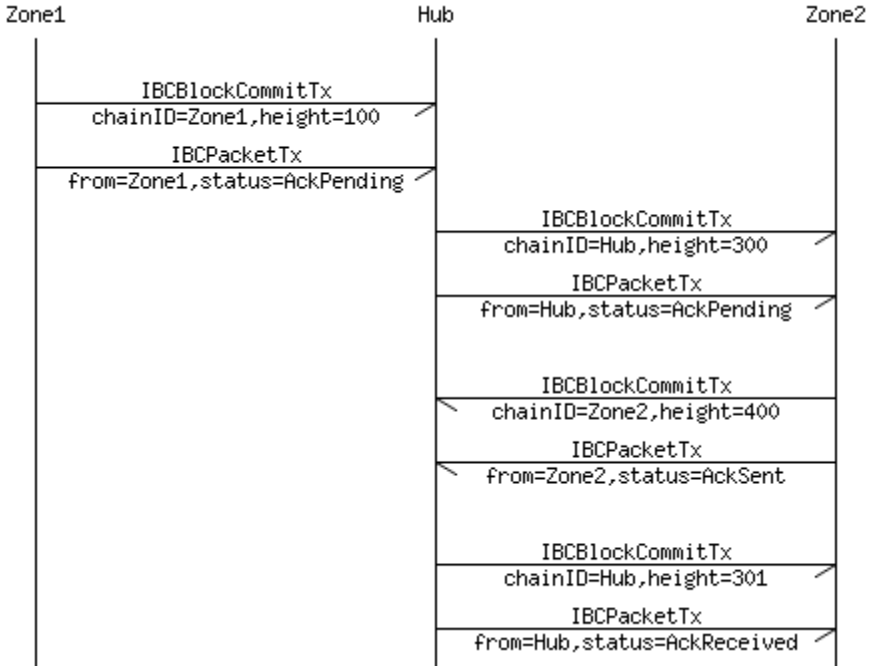
  为区块结束提供信号。在每次提交前所有交易后调用。

更多细节请参考 TMSP 知识库。

IBC 包交付确认

发送人可能会需要接收链提供包交付确认，这个原因有很多。比如，发送人可能不了解目的链的状态，如果有问题的话也不得而知。或者，发送者可能会想要向包强加一次超时（借助 `MaxHeight` 即最大值包域），而目的链可能会遇到拒绝服务攻击，比如接受包数量突然暴增。

在这些案例中，发送人可以通过在 `AckPending` 上设置初始包状态来要求提供交付确认。接着就由接收链通过在应用程序的梅克尔哈希中包括一个缩写的 `IBCPacket` 来确认交付。



首先，`IBCBlockCommit` 以及 `IBCPacketTx` 是公布在“Hub”（中心）上用来证明“Zone1”（空间 1）上的 `IBCPacket` 的存在的。假如 `IBCPacketTx` 的值如下：

- `FromChainID`: "Zone1"
- `FromBlockHeight`: 100 (假如)
- `Packet`: an `IBCPacket`:
    - `Header`: an `IBCPacketHeader`:
        - `SrcChainID`: "Zone1"
        - `DstChainID`: "Zone2"
        - `Number`: 200 (假如)
        - `Status`: `AckPending`
        - `Type`: "coin"
        - `MaxHeight`: 350 (假如“Hub”目前的高度是 300)
    - `Payload`: <The bytes of a "coin" payload>（一个“代币”的有效负荷字节）

第二步，`IBCBlockCommit` 和 `IBCPacketTx` 被公布到“Zone2”（空间 2）来证明 `IBCPacket` 存在于“Hub”上。假如 `IBCPacketTx` 的值如下：

- `FromChainID`: "Hub"

- FromBlockHeight: 300
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
        - SrcChainID: "Zone1"
        - DstChainID: "Zone2"
        - Number: 200
        - Status: AckPending
        - Type: "coin"
        - MaxHeight: 350
    - Payload: <The same bytes of a "coin" payload>（一个“代币”相同的有效负荷字节）

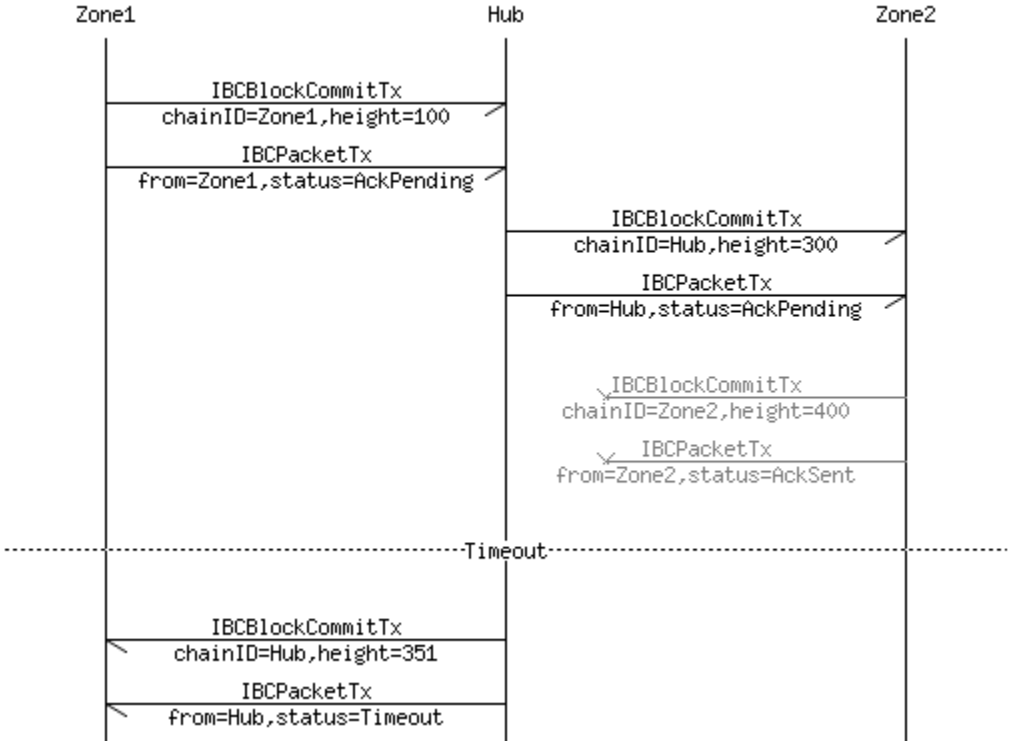
接下来，“Zone2”必须将缩写的包放入其应用程序哈希中来显示 AckSent 的最新状态。IBCBlockCommitand 和 IBCPacketTx 会公布到“Hub”上来证明缩写的 IBCPacket 存在于“Zone2”上。假如 IBCPacketTx 的值如下：

- FromChainID: "Zone2"
- FromBlockHeight: 400 (say)
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
        - SrcChainID: "Zone1"
        - DstChainID: "Zone2"
        - Number: 200
        - Status: AckSent
        - Type: "coin"
        - MaxHeight: 350
    - PayloadHash: <The hash bytes of the same "coin" payload>（相同“代币”有效负荷的哈希字节）

最后，“Hub”必须更新从 AckPending 到 AckReceived 的包的状态。这个最新状态的证据要回到“Zone2”。假如 IBCPacketTx 的值如下：

- FromChainID: "Hub"
- FromBlockHeight: 301
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
        - SrcChainID: "Zone1"
        - DstChainID: "Zone2"
        - Number: 200
        - Status: AckReceived
        - Type: "coin"
        - MaxHeight: 350
    - PayloadHash: <The hash bytes of the same "coin" payload>（相同“代币”有效负荷的哈希字节）

同时，“Zone1”可能会积极地假设“代币”包的交付是成功的，除非“Hub”上有证据给出相反的证明。在上述例子中，如果“Hub”没有从“Zone2”接收到区块 350 的 AckSent 状态，那么它就会自动将这个设置到 Timeout（超时）。这个超时证据可以贴回到“Zone1”上，然后就可以返回任意代币。

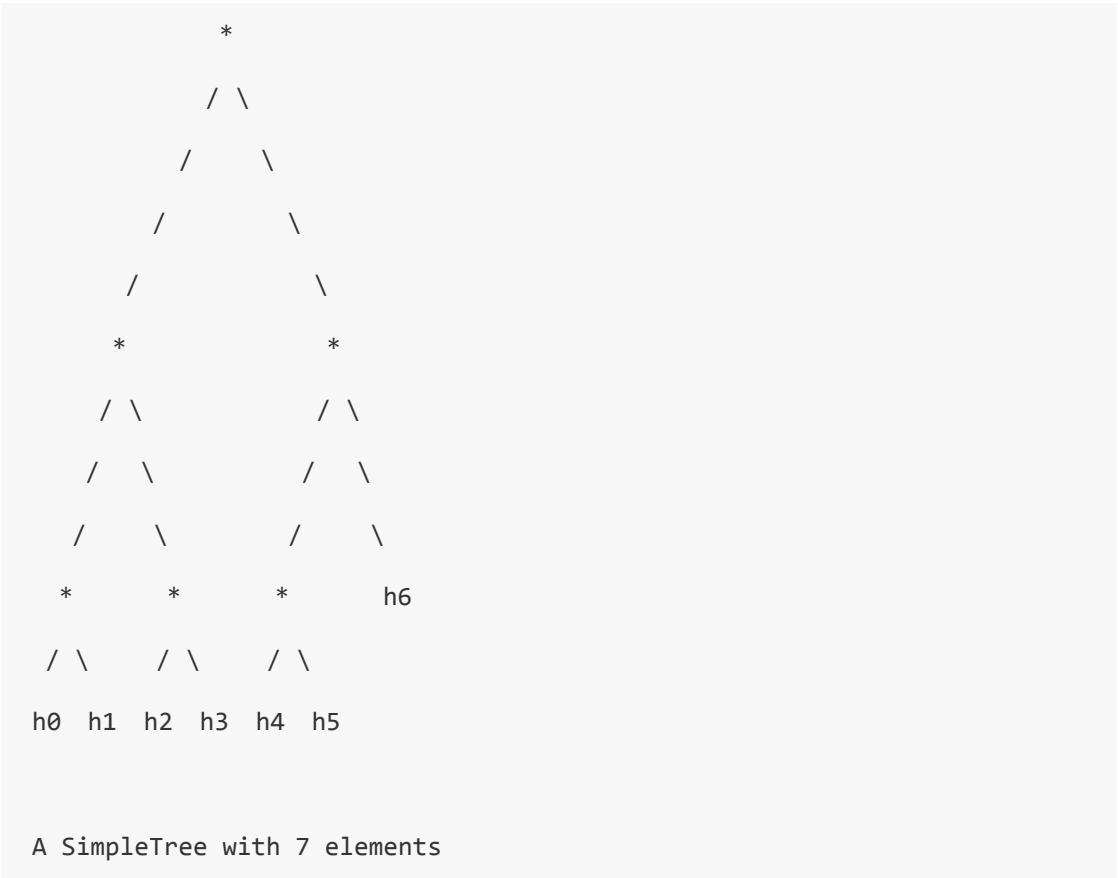


梅克尔树及梅克尔证明说明

Tendermint/Cosmos 生态支持的梅克尔树有两种：简单的和 IAVL+的。

简易版梅克尔树

简易版梅克尔树针对的元素的静态列表。如果项的数目不是 2 的次方，那么有些树叶就会在不同的层。简易树试图让树的两侧在同一高度，但是左边可能会稍大一点。这种梅克尔树就是用于一个区块交易的梅克尔化的，而顶层元素就是应用状态根。



IAVL+梅克尔树

IAVL+数据结构的目的是永久储存应用状态中的密钥-值，这样具有决定功能的梅克尔根哈希就可以有效进行运算了。这个树的平衡通过 AVL 算法的一个变量来达到，所有运行都是 O(log(n))。

在 AVL 树中，任意节点两个子树的高度至少有一处不同。无论什么时候出现更新来打破这种情况，这个树都会通过创造 O(log(n))新节点（指向旧树上未修改的节点）来再次达到平衡。在初始 AVL 算法中，内部节点也可以维持密钥-值。AVL+算法（注意这里有个“+”号）对 AVL

算法进行了修改，来维持所有树叶节点上的值，同时还只需采用分支-节点来存储密钥。这一点在简化算法的同时，还能维持较短的梅克尔哈希轨迹。

AVL+树类似于以太坊的 Patricia tries（帕氏树）。其中也有一定的折中。密钥不需要在嵌入到 IAVL+树前生成哈希，而这个就为密钥空间里提供了较快的命令迭代，或许为很多应用程序都带去了好处。逻辑实现很简单，只需要两种节点——内部节点和树叶节点。作为一个平衡的二进制树，梅克尔证明算是短的。而另一方面，IAVL+树有取决于命令的更新。

我们将支持额外有效的梅克尔树，比如以太坊的帕氏树，同时实现二进制变量的可用性。

交易类型

在标准实现中，交易通过 TMSP 界面流入 Cosmos Hub 的应用程序。

Cosmos Hub 将接受几类主要交易，包括 `SendTx`，`BondTx`，`UnbondTx`，`ReportHackTx`，`SlashTx`，`BurnAtomTx`，`ProposalCreateTx`，以及 `ProposalVoteTx`（即发送交易、绑定交易、解绑交易、攻击报告交易、削减交易、Atom 燃烧交易，创建提议交易），这些都不需要加以寿命，会在之后文章更新中进行归档。这里我们主要列举两个主要的 IBC 交易类型：`IBCBlockCommitTx` 以及 `IBCPacketTx`（即 IBC 区块提交交易以及 IBC 包交易）

IBCBlockCommitTx（IBC 区块提交交易）

`IBCBlockCommitTx` 交易主要由这些组成：

- `ChainID (string)`: 区块链 ID
- `BlockHash ([]byte)`: 区块哈希字节，就是梅克尔根（包括应用程序哈希）
- `BlockPartsHeader (PartSetHeader)`: 区块部分设置的头字节，只用于验证投票签名
- `BlockHeight (int)`: 提交高度
- `BlockRound (int)`: 提交回合
- `Commit ([]Vote)`: 超过⅔的 Tendermint 预提交投票，以组成区块提交项
- `ValidatorsHash ([]byte)`: 新验证组的梅克尔树根哈希
- `ValidatorsHashProof (SimpleProof)`: 简易版梅克尔树证明，在区块哈希中证明验证人哈希

- AppHash ([]byte): IAVL 树，应用程序状态的梅克尔树根哈希
- AppHashProof (SimpleProof): 简易版梅克尔树证明，在区块哈希中验证应用程序哈希

IBCPacketTx（IBC 包交易）

IBCPacket 由下列项组成：

- Header (IBCPacketHeader): 包头
- Payload ([]byte): 包有效负荷字节。*可选择*。
- PayloadHash ([]byte): 包字节哈希。*可选择*。

有效负荷或有效负荷哈希必须存在一个。IBCPacket 的哈希就是两个项的简易版梅克尔根，即头和有效负荷。没有完整有效负荷的 IBCPacket 被称作缩写版包。

IBCPacketHeader 由下列项组成：

- SrcChainID (string): 源区块链 ID
- DstChainID (string): 目标区块链 ID
- Number (int): 所有包特定数量
- Status (enum): 可以是 AckPending，AckSent，AckReceived，NoAck，或 Timeout 任意一个
- Type (string): 种类根据应用程序决定。Cosmos 保留“coin”（币）包种类。
- MaxHeight (int): 如果状态不是这个高度给出的 NoAckWanted 或者 AckReceived，那么状态就算超时。*可选择*。

IBCPacketTx 交易有下列项组成：

- FromChainID (string): 区块链 ID，用于提供这个包，不是必要的来源
- FromBlockHeight (int): 区块链高度，其中接下来的包会包含在源链的区块哈希中
- Packet (IBCPacket): 数据包，其状态可以是 AckPending，AckSent，AckReceived，NoAck，或 Timeout 任意一个

- `PacketProof (IAVLProof)`: IAVL 树梅克尔证明，用于一定高度的源链中的应用哈希中验证包的哈希

通过“Hub”，从“Zone1”发送到“Zone2”的包的序列会用{Figure X}函数进行描述。首先一次 `IBCPacketTx` 会向“Hub”证明包是包含在“Zone1”的应用程序状态中。然后，另一次 `IBCPacketTx` 会向“Zone2”证明包包含在“Hub”的应用程序状态中。在这个过程中，`IBCPacketTx` 的域是一样的：`SrcChainID` 永远是“Zone1”而 `DstChainID` 永远是“Zone2”。

`PacketProof` 必须有正确的梅克尔证明路径，如下：

```
IBC/<SrcChainID>/<DstChainID>/<Number>
```

当“Zone1”想要向“Zone2”通过“Hub”发送证明，那么 `IBCPacket` 的数据是相同的，无论这个包是在“Zone1”、 “Hub”还是“Zone2”上梅克尔化的。唯一可变的域只有追踪交付的 `Status` （状态），如下所示。

鸣谢

感谢朋友及同行在概念成型与检查方面提供的帮助，以及对我们同 Tendermint 及 Cosmos 工作的支持。

- SkuChain 的 Zaki Manian 在 格式和措辞方面提供了很多支持，尤其是 TMSP 部分。
- Althea and Dustin Byington 的 Jehan Tremback 在初始迭代方面帮助。
- Honey Badger 的 Andrew Miller 对共识部分给予的反馈。
- Greg Slepak 对共识及措辞给予的反馈。
- 同时还要感谢 Bill Gleim 和 Seunghwan Han 的各种支持与贡献。
- **此处还有您及您的组织对本文的贡献。**

引用

- 1 Bitcoin: https://bitcoin.org/bitcoin.pdf
- 2 ZeroCash: http://zerocash-project.org/paper
- 3 Ethereum: https://github.com/ethereum/wiki/wiki/White-Paper

- 4 TheDAO: https://download.slock.it/public/DAO/WhitePaper.pdf
- 5 Segregated Witness: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki
- 6 BitcoinNG: https://arxiv.org/pdf/1510.02037v2.pdf
- 7 Lightning Network: https://lightning.network/lightning-network-paper-DRAFT-0.5.pdf
- 8 Tendermint: https://github.com/tendermint/tendermint/wiki
- 9 FLP Impossibility: https://groups.csail.mit.edu/tds/papers/Lynch/jacm85.pdf
- 10 Slasher: https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/
- 11 PBFT: http://pmg.csail.mit.edu/papers/osdi99.pdf
- 12 BitShares: https://bitshares.org/technology/delegated-proof-of-stake-consensus/
- 13 Stellar: https://www.stellar.org/papers/stellar-consensus-protocol.pdf
- 14 Interledger: https://interledger.org/rfcs/0001-interledger-architecture/
- 15 Sidechains: https://blockstream.com/sidechains.pdf
- 16 Casper: https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/
- 17 TMSP: https://github.com/tendermint/tmsp
- 18 Ethereum Sharding: https://github.com/ethereum/EIPs/issues/53
- 19 LibSwift: http://www.ds.ewi.tudelft.nl/fileadmin/pds/papers/PerformanceAnalysisOfLibswift.pdf
- 20 DLS: http://groups.csail.mit.edu/tds/papers/Lynch/jacm88.pdf
- 21 Thin Client Security: https://en.bitcoin.it/wiki/Thin_Client_Security
- 22 Ethereum 2.0 Mauve Paper: http://vitalik.ca/files/mauve_paper.html

未分类链接

- https://www.docdroid.net/ec7xGzs/314477721-ethereum-platform-review-opportunities-and-challenges-for-private-and-consortium-blockchains.pdf.html