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使人们启发和理解，去中心化的互联网协议是可以从根本上改善社会经济基础架构的。我们已经见识过个有专长的区块链应用，诸如比特币[1]（加密货币），ZCASH [2] (隐私加密货币)，也看到了例如以太坊 [3] 的大众智能合约平台，还有无数基于 EVM (以太坊虚拟机)开发的分布式应用，例如 Augur（预测市场）和 TheDAO [4] （投资俱乐部）

然而，迄今为止，这些区块链已经暴露了各种缺陷，包括总体能效低下，性能不佳或受到限制和缺乏成熟的治理机制。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 [5]（Segregated-Witness）和 BitcoinNG [6]（一种新的可扩展协议）这样的解决方案，但这些垂直扩展解决方案仍然受到单一物理机容量的限制，以确保完整的可审计性。闪电网络 [7]可以通过部分交易完全记录在

主链账本外来扩展比特币的交易容量，这种方法十分适用于微支付和隐私保护支付通道，但是无法适用于更通用的扩展需求。

理想的解决方案是允许多个并行的区块链交互操作的同时保持其安全特性。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，允许在工作完成的同时，确保母链在子链上被重复使用，但交易必须通过每个节点依次进行验证，而且如果母链上的大多数哈希算力没有积极地对子链进行合并挖矿，那么就容易遭受到攻击。关于可替代区块链网络架构的学术回顾 将在附件中展示，我们也会在相关工作中对其他（技术）方案和缺陷进行概括。

这里我们要介绍的 Cosmos，一个全新的区块链网络架构，能够解决所有这些问题。Cosmos 是由许多被称之为“分区”的独立区块链组成的网络。分区在 Tendermint Core [8]的支持下运行，Tendermint Core 是一个类似拜占庭容错安全共识引擎，具有高性能、一致性的特性，并且在严格的分叉追责 机制下能够制止恶意破坏者的行为。Tendermint Core 的拜占庭容错共识算法十分适合用于扩展权益证明(PoS)机制下的公共区块链。使用其他共识模型的区块链，包括类似基于权益证明(PoS)的以太坊，以及比特币也能够通过使用适配分区被 Cosmos 网络连接。

Cosmos 的第一个分区称之为 Cosmos 枢纽。Cosmos 枢纽是一种多资产权益证明加密货币网络，它通过简单的治理机制能够对网络进行适配和升级。此外，Cosmos 枢纽可以通过链接其他分区来实现扩展。

Cosmos 网络的枢纽及各个分区可以通过区块链间通信（IBC）协议进行通信，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、快速地从一个分区转到其他分区，而无需在两个分区之间拥具有汇兑流动性。相反，所有跨分区的代币转移都会通过 Cosmos 枢纽，以此来追踪记录每个分区持有代币的总量。这个枢纽会将每个分区与其他故障分区隔离开。因为每个人都可以将新的分区连接到 Cosmos 枢纽，所以分区将可以向后兼容新的区块链技术。

利用 Cosmos 可以实现区块链间的互操作。这是一个具有潜力的有价值的互联网络，其中的资产由不同的验证人发布和控制，并可以在不依靠需要信任的第三方的情况下实现跨链资产无缝的转移和交易。

Tendermint

在这一部分我们将阐述 Tendermint 共识协议和用于建立其应用程序的接口。更多信息，请参见附录

验证人

在经典的拜占庭容错算法中，每个节点有相同的权重。在 Tendermint，节点有着不同数量（非负）的 *投票权*，而那些拥有相当数量投票权的节点称之为 *验证人*。验证人通过广播加密签名、投票或者对下一个区块表决同意来参与共识协议。

验证者的投票权是一开始就确定好了，或者根据应用程序由区块链来决定修改投票权。例如，在像 Cosmos 枢纽的权益证明应用里，投票权可由绑定为押金的代币数量来决定。

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权重。而>⅔ 的意思是"超过⅔ "，≥⅓则是"⅓或者更多"的意思。

共识

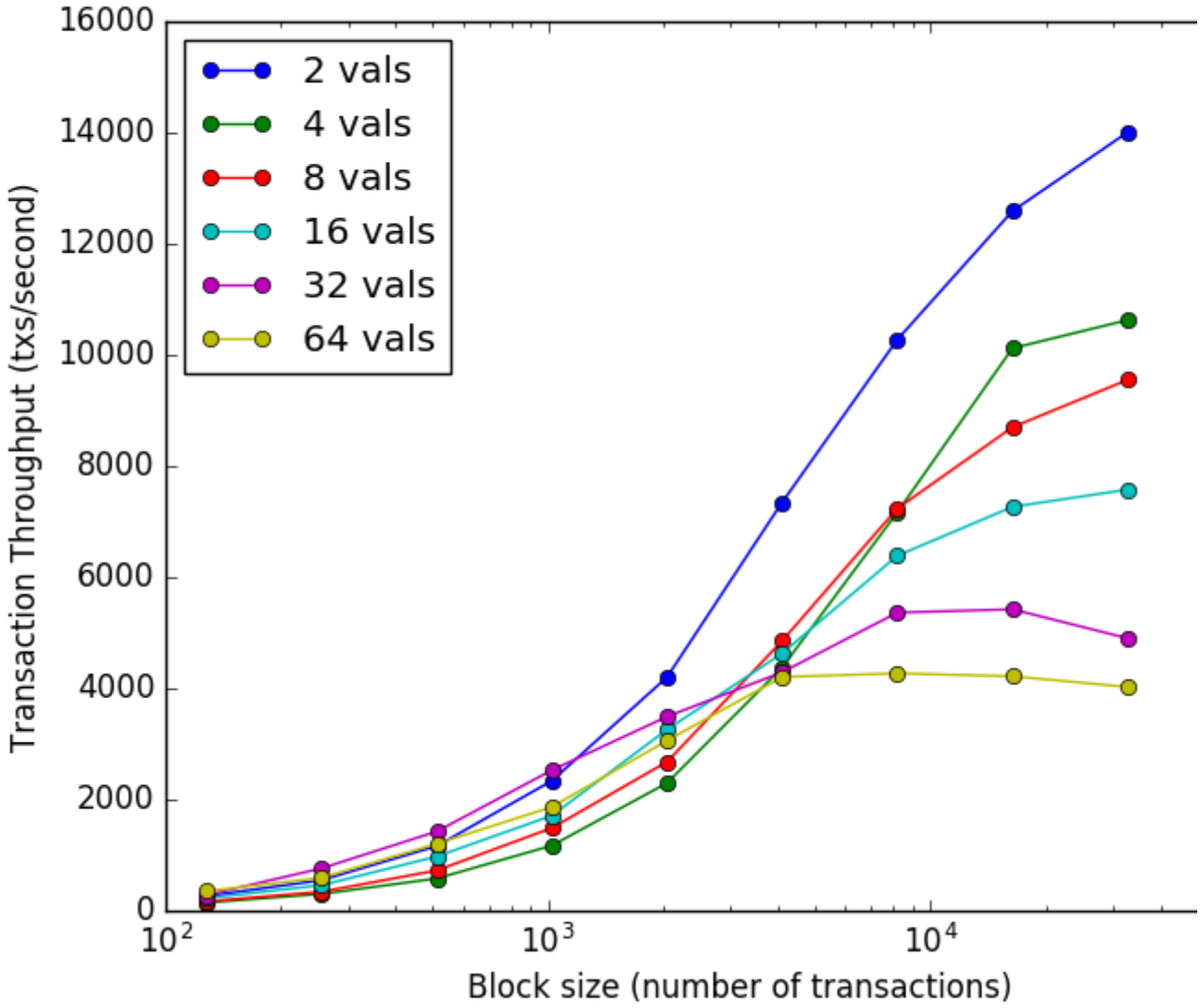
Tendermint 是部分同步运作的拜占庭容错共识协议，这种协议源自 DLS 共识算法 [20]。Tendermint 以简易性、高性能以及分叉问责制而著称。协议要求这组验证人固定且被熟知，并且每个验证人都有其公钥验证身份。这些验证人试图同时在一个区块上达成共识，这些区块是一系列的交易记录。每个区块的共识轮流进行，每一轮都会有个领头人，或者提议人，由他们来发起区块。之后验证人分阶段对是否接受该区块，或者是否进入下一轮做出投票。每轮的提议人会从验证人顺序列表中按照其投票权比例来选择确定。

更多协议的全部细节，请点击这里.

Tendermint 采用了使用大多数投票（超过三分之二）和锁定机制的最优拜占庭容错，来确保其安全性。这些能够保证：

- 蓄意破坏者想要造成安全性问题，必须有三分之一以上的投票权，并且要提交超过两份以上的值。
- 如果有一组验证人成功破坏了安全性，或者曾试图这么做，他们会被协议识别。协议包括对有冲突的区块进行投票和广播那些有问题的投票。

除了其超强的安全性外，Tendermint 还具备杰出的性能。以商用型云平台为例，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，订单提交延迟时间为 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广播恶意破坏的投票，也能维持这种每秒超过千笔交易的较高性能。详见下图。



轻客户端

Tendermint 共识算法的主要好处是具有安全简易的客户端，使其成为手机和物联网用例的理想选择。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链，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一些明显的攻击，比如远程无利害关系双花攻击及审查制度。 这些在附录中有更详细的讨论。

ABCI

Tendermint 共识算法是在叫做 Tendermint Core 的程序中实现的。这个程序是独立于应用的“共识引擎”，可以将任何已经确定的黑盒应用转变为分布式、可复制的区块链。Tendermint Core 可以通过应用区块链接口(ABCI) [17]与其他区块链应用连接。而且，应用区块链接口(ABCI) 接口允许区块链应用以任何语言编程实现，而不仅仅是写这个共识引擎所使用的语言。此外，应用区块链接口(ABCI) 也让交换任何现有区块链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。如果您想要在应用区块链接口(ABCI)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，ABCI 应用也可以做到：

- 维护 UTXO 数据库
- 验证交易的加密签名
- 防止出现不存在的余额被交易
- 允许客户访问 UTXO 数据库

Cosmos 概述

Cosmos 是一个独立平行的区块链网络，其中每条区块链通过 Tendermint1 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 枢纽。Cosmos 枢纽通过全新的区块链间通信协议来连接其他众多区块链（或将其称之为 *分区*）。Cosmos 枢纽可以追踪无数代币的种类，并且在各个连接的分区里记录各种代币总数。代币可以安全快速地从一个分区转移到另一个分区，两者之间无需体现汇兑流动性，因为所有分区之间的代币传输都会经过 Cosmos 枢纽。

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、以及可无缝升级的能力。比如，从 Bitcoind、Go-Ethereum、CryptoNote、ZCash 或其他区块链系统中衍生出来的分区，都能被锚定接入 Cosmos 枢纽。这些分区允许 Cosmos 实现无限扩展，从而满足全球交易的需求。此外，分区也完全适用于分布式交易所，反之交易所也支持分区运行。

Cosmos 不仅仅是单一的分布式账本，而 Cosmos 枢纽也不是封闭式庭院或宇宙的中心。我们正在为分布式账本的开放网络设计一套协议，这套协议将基于

密码学、稳健经济学、共识理论、透明性及可追责制的原则，成为未来金融系统的全新基础。

Tendermint-拜占庭容错

Cosmos 枢纽是 Cosmos 网络中第一条公共区块链，通过 Tendermint 的拜占庭共识算法运行。Tendermint 开源项目创立于 2014 年，旨在解决比特币工作量证明共识算法的速度、可扩展性以及造成的环境问题。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）[20]，Tendermint 成为了首个在概念论证了权益证明加密货币的团队，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”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双重支付攻击。

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 被设计为完全不分叉，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 中的验证人角色类似比特币矿工，但是他们采用加密签名来进行投票。验证人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”币）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，那么代币就会面临被惩罚（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、可量化的安全性。

治理

分布式公共账本应该要有一套章程与治理体系。比特币依靠比特币基金会以及挖矿来协作更新，但是这是一个反应缓慢的治理制度。以太坊在采用硬分叉成 ETH 和 ETC 来解决 The DAO 黑客，这主要是因为之前没有设定社会契约或机制来进行这类决定。

Cosmos 枢纽的验证人与委托人可以对提案进行投票，从而改变预先默认设置好的系统参数（比如区块转账费用限制），协作更新，并对可读性的章程进行修订投票，从而治理 Cosmos 枢纽制度。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），并得出更快更明确的解决方案。

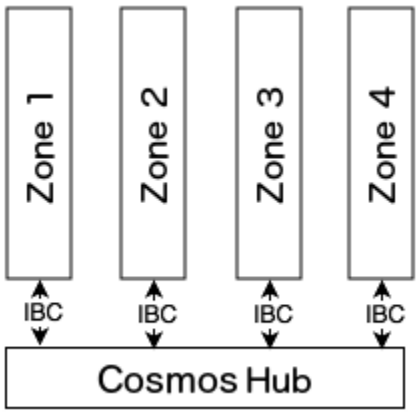
每个分区也可以制定自己的一套章程及治理机制。比如，Cosmos 枢纽的章程可以设置为强制实现枢纽的不可改变性（不能回滚，除了 Cosmos 枢纽节点产生的漏洞），而每个分区则可设置自己的回滚政策。

Cosmos 网络能够在制度不同的分区间实现互操作性，这一点给客户极高的自由度和潜力而无需许可即可实验（新技术）。

枢纽与分区

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 机制来运行众多的区块链。虽然现存提案的目标是创建一个包含全球所有交易订单的”单一区块链”，但是 Cosmos 允许众多区块链在并行运行的同时，保持可互操作性。

在这个基础上，Cosmos 枢纽负责管理称之为“分区”的众多独立区块链（有时也叫做”分片”，参考自众所周知的数据库扩展技术”分片”）。枢纽上的分片会源源不断地提交最新区块，这一点可以让枢纽同步每一个分区的状态。同样地，每个分区也会和枢纽的状态保持一致（不过分区之间不会同彼此的同步，除非间接通过枢纽来实现）。通过发布默克尔证明来证明消息被接受和发送，来让消息从一个分区传递到另一个分区。这种机制叫做”区块链间通信”，或者简称为”IBC”机制。



任何分区都可以自行成为枢纽来建立非循环图表，但为了清楚起见，我们只描述这种只有一个枢纽和许多非枢纽的分区这样简单的配置

枢纽

Cosmos 枢纽是承载多种分布式账本资产的区块链，其中代币可以由个人或分区自己持有。这些代币能够通过特殊的 IBC 数据包，即"代币数据包"（coin packet）从一个分区转移到另一个分区。枢纽负责保持各个分区中各类代币总量不变。IBC 代币数据包交易必须由发送人、枢纽及区块接受者执行。

因为 Cosmos 枢纽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。虽然每个分区可能都是一个 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 枢纽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，比如区域网络分裂或者由国家发起的攻击。

分区

Cosmos 分区是独立的区块链，能够和 Cosmos 枢纽进行 IBC 消息交换。从枢纽的角度上看，分区是一种多重资产、动态会员制的多重签名账户，可以通过 IBC 数据包用来发送和接受代币。就像加密币账户一样，分区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。分区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 枢纽的 Atom 或可作为分区验证人连接到枢纽的筹码。虽然在 Tendermint 分叉责任制下，分区出现双重支付攻击会导致 atom 数量减少，但是如果分区中有超过⅔的选票都出现拜占庭问题的话，那这个分区就可以提交无效状态。Cosmos 枢纽不会验证或执行提交到其他分区的交易，因此将代币发送到可靠的分区间就是用户的责任了。未来 Cosmos 枢纽的管理系统可能会通

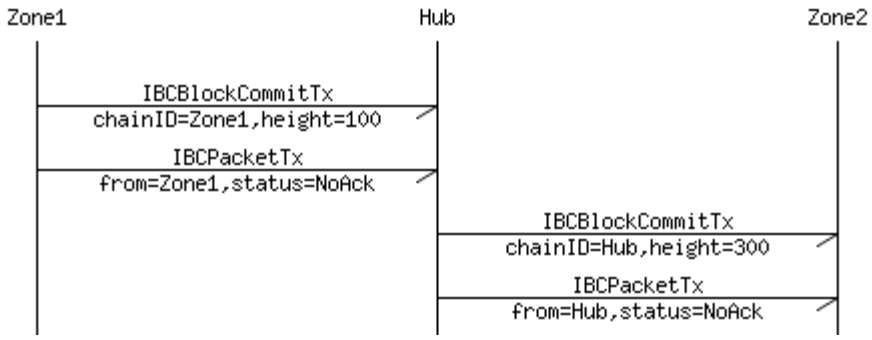
过改善提案，来解决分区故障问题。比如，在检测到袭击时，可以将有些分区（或全部分区）发起的代币转账将被暂停，实现紧急断路（即暂时中止代币转账）。

跨链通信-IBC

现在我们来介绍下枢纽与分区之间通信的方法。假如现在有三个区块链，分别是”分区 1″、“分区 2″以及”枢纽”，我们想要”分区 1″生成一个数据包，通过”枢纽”发送给”分区 2″。为了让数据包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数据包到指定目的地。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数据包确实由发送者的应用程序，通过默克尔证明机制（Merkle-proof）传送到了最新区块的哈希值上。

通过将 IBC 机制分离成两个单独的交易，即 IBCBlockCommitTx 交易与 IBCPacketTx 交易，我们可以让接收方链的本地费用市场机制，来决定承认哪个数据包，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数据包数量。



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

_关于两种 IBC 交易类型，详细请参见 IBCBlockCommitTx 和 IBCPacketTx

用例

分布式交易所

比特币借助大量复制来增加分布式账本的安全性。用类似的方式，我们可以在区块链上运行交易所，来降低其受内部及外部攻击的可能性。我们称之为去中心化交易所。

现今，加密货币社区认为的去中心化交易所基于”跨链原子事务”交易（ AXC 交易）。通过 AXC 交易，两条不同链上的两个用户可以发起两笔转账交易，交易在两个账本上要么一起提交执行，或者两个账本都不执行（即交易的原子性）。比如，两位用户可以通过 AXC 交易来实现比特币和以太币之间的交易（或是在不同账本上的任意两种代币），即使比特币和以太坊的区块链之间并

没有彼此连接。AXC 交易模式下的交易所用户双方不需要彼此信任，也不用依赖交易匹配服务。其弊端是，交易双方必须同时在线才能进行交易。

另一种去中心化交易所是进行大量复制的具有独立区块链的分布式交易所。该种交易所的用户可以提交限价订单并关闭他们的计算机，交易可以在用户离线状态下执行。区块链将会代表交易者去完成匹配和交易。

一个中心化的交易所可以构建一个有大交易量的限价交易的买卖盘账目，以此来吸引更多的交易者。在交易所领域，流动性会引发更多流动性，因此在交易所业务中，其具有的网络效应也愈发明显（或者说至少产生了”赢家通吃”效应）。目前加密货币交易所 Poloniex 以每 24 小时 2,000 万美元的交易量排名第一， Bitfinex 则每 24 小时 500 万美元的交易额位列第二。在这种强大的网络效应之下，基于 AXC 的去中心化交易所的成交量是不太可能超过中心化交易所。去中心化交易所要想和中心化交易所一争高下，就需要支持以限价订单构成的具有深度的交易买卖盘账目的运行。而只有基于区块链的去中心化交易所可以实现这一点。

Tendermint 提供的快速交易执行是另一大优势。Cosmos 的内部网络可以在不牺牲一致性的前提下优先快速的确定最终性，来实现交易的快速完成 —— 同时针对交易订单交易，以及 IBC（跨区块链通信）代币与其他网络的交易。

综上，根据现有加密货币交易所的情况，Cosmos 的一项重大应用就是去中心化交易所（称为 Cosmos DEX）。其交易吞吐能量和委托延时可以与那些中心化交易所媲美。交易者可以在各方离线的状态下提交限价订单。并且，基于

Tendermint，Cosmos 枢纽以及 IBC 的情况下，交易者可以快速地完成在交易所及其他网络的资金进出。

作为其他加密货币的纽带

特权分区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 枢纽与分区之间的关系，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。Cosmos 网络上挂钩的”桥接分区“要和中心以及其他加密币保持同步。这种间接通过”桥接分区“可以保持枢纽逻辑的简洁。并且不必要了解其他的链上共识战略，如比特币工作量证明挖矿机制。

向 Cosmos 枢纽发送代币

每个挂钩桥接分区的验证人都会在基于 Tendermint 公式的区块链之上，运行带有特殊的 ABCI 桥接应用程序，但同时也会运行一个原有区块链的“全节点”。

在原有区块链挖出新区块时，桥接分区验证人员将通过签署和分享起始点区块链的提示，各自局部视角可以达成一致。当一个桥接分区收到原有区块链的支付时（如在以太坊或比特币等 PoW 机制的链上有足够数目的确认），则在该桥接分区上创建具有该对应账户的余额。

就以太坊而言，桥接分区可以和 Cosmos 枢纽共享相同的验证人。以太坊方面（原本区块链），一个桥接合约将允许以太拥有者通过将以太币发送到以太坊的桥接分区的桥接合约上。一旦挂桥接合约接收到以太币，以太币就不能被撤

回，除非从桥接分区接收到对应的 IBC 数据包。桥接合约跟随桥接分区的验证组，它可能与 Cosmos 枢纽的验证人组相同。

就比特币而言，概念是相似，除了代替一个桥接合约，每个 UTXO 将由一个门限多重签名 P2SH 数据库限制。由于 P2SH 系统的限制，签名者不能与 Cosmos 枢纽的验证人组相同。

从 Cosmos 枢纽提出代币

桥接分区上的以太币（“桥接以太币”）可以在枢纽间转进，转出，完成传送到特定以太坊提取地址后，转出的“桥接以太币”被彻底删除。一个 IBC 消息可以证明桥接分区上的交易，这个消息将被公布到以太坊桥接合约中，以便以太币被取出。

就比特币而言，严谨的交易脚本系统让 IBC 币的镜像转换机制很难实现。每个 UTXO 都有自己的特定的脚本，所以当比特币履约签名者发生变化时，每个 UTXO 都必须迁移到新的 UTXO。一个解决方案是根据需要，压缩和解压缩 UTXO-set，以保持 UTXO 的总数量下降。

挂钩区完全责任制

这类挂钩合约存在风险的风险是，可能会出现恶意的验证人组。如果拜占庭投票权超过⅓，就会造成分叉，即从以太坊桥接合约中提取以太币的同时，还能保持桥接分区中的挂钩以太币不变。甚至，如果拜占庭投票权超过⅔，可能会

有人直接通过脱离原始桥接分区的桥接逻辑，对发送以太币发到桥接合约中的帐户下手，盗取以太币。

如果将这个桥接方法完全设计成责任制，就有可能解决这一问题。比如，枢纽及起始点的全部 IBC 包裹可能需要先通过桥接分区的认可，即让枢纽或起始点中的桥接合约对桥接分区的所有状态转换进行有效验证。枢纽及起始点要允许桥接分区的验证人提供抵押物，而侨界合约的代币转出需要延时（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任何的质询。我们会把这一系统的设计说明以及执行方式开放，作为未来 Cosmos 改善的提议，以待 Cosmos 枢纽的管理系统审批通过。

以太坊的扩展

众所周知，扩展问题是一直困扰着以太坊的问题。目前以太坊节点会处理节点上每笔交易，并且存储所有的状态参考.。

Tendermint 提交区块的速度比以太坊工作量证明要快，所以由 Tendermint 共识推动且使用桥接以太币运行的以太坊虚拟机分区能够强化太坊区块链的性能。此外，虽然 Cosmos 枢纽及 IBC 包裹机制不能实现每秒合约逻辑的执行，但是它可以用来协调不同分区中以太坊合约之间的代币流通，通过分片方式为以代币为中心的以太坊扩展奠定基础。

多用一体化

Cosmos 分区可以运行任意的应用逻辑，应用在分区创建时设定好，可通过管理者可以不断更新。这种灵活度使得 Cosmos 分区可以作为其他加密币的挂钩载体，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，利用同样的代码库，而在验证程序及初始分配有所区分。这样就允许多种现有加密币框架得以运行，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，为平台之间提供更多的交互机会。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，当然这里假设交易订单通过其他平台进行匹配。替代方案是，让分区作为分布式容错交易所（包含买卖盘账目），这算是对中心化加密币交易所之上的严格改进——现行交易所在过去时常发生被攻击的事件。

分区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 ABCI 应用在某个分区上运行，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，又不愿彻底放弃控制权给分布式第三方的人，提供绝佳的运行环境。

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，就是网络分区会导致没有任何一个分区会拥有超过⅔的投票权（比如超过⅓投票权在线下），这会中断共识。而 Cosmos 架构可以缓解这个问题，它可以使用全球中心，同时，各分区实行地区自治，然后让每个分区的投票权按照正常的地域位

置进行分配。如，一般范例就有可能是针对个别城市或地区的，让他们各自运行自己分区的同时，还能共享共同的枢纽（比如 Cosmos 枢纽），并且在临时的网络分区导致的中断期间，也可以继续维持地区自治活动。注意，这样一来在设计稳健的联邦式容错系统过程中，就可以去考虑真实的地理、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技术解决名称解析问题的区块链之一。不过，这个方案存在一些不足。

例如，我们可以通过 Namecoin 来验证@satoshi（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，该公钥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之前的所有全部下载。这一点是比特币 UTXO 交易模式中默克尔化模型的局限性导致的，这类模型中只有交易（而非可变的应用状态）会以默克尔化加入到区块哈希中。它让我们得在之后用更新证明名称的存在，而非不存在。因此，我们必须依靠全节点才能明确这个名称的最近的值，或者花费大量资源下载整个区块链。

即使在 NameCoin 上运用默克尔化的搜索树，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备份（或者至少是自其最后的名称更新的所有区块头）。这意味着带宽需要随着时间做线性的扩展。21 此外，在工作量证明制度使区块链上的名称更改需要等额外的工作量证明验证确认才能进行，它在比特币上可能要花费一个小时。

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值的默克尔证明。这点让简易、快速、安全的轻客戶端名称值验证成为可能。

在 Cosmos 中，我们可以利用这个概念并延伸。每一个在 Cosmos 上的名称注册都能有一个相关的最高级别域名（TLD），比如".com"或者".org"等，而且每个名称注册分区都有自己的管理和登记规则。

发行与激励

Atom 代币

Cosmos 枢纽是多资产分布式账本，它有自己的代币，是 Atom 。Atom 是 Cosmos 枢纽唯一的权益代币。Atom 是持有人投票、验证或委托给其他验证人的许可证明，就像以太坊上的以太币一样，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激励分给验证人及委托验证人。

`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发是 Cosmos 众售参与者持有 75%，预售参与者持有 5%，Cosmos 网络基金会持有 10%，ALL IN BITS，集团持有 10%。从创世块开始，Atom 总量的 1/3 将作为奖励发放给每年担保持有的验证人以及委托人。

更多细节见 Cosmos Plan

验证人的数量限制

与比特币或其他工作量证明区块链不同的是，由于通信的复杂性增加，Tendermint 区块链会随着验证人的增加而变慢。幸运的是，我们可以支持足够多的验证人来实现可靠的全球化分布式区块链， 并具有非常快的交易确认时间。 而且随着带宽、存储和并行计算容量的增加，我们将来能够支持更多的验证人。

在创世日，验证人的最大数量将设置为 100，这个数字将以 13% 的速度增长 10 年，最终达到 300 位。

```
1. Year 0: 100
2. Year 1: 113
3. Year 2: 127
4. Year 3: 144
5. Year 4: 163
6. Year 5: 184
7. Year 6: 208
8. Year 7: 235
9. Year 8: 265
10.  Year 9: 300
11.  Year 10: 300
12.  ...
```

成为创世日后的验证人

Atom 持有者可以通过签署和提交 `BondTx` 交易成为验证人。抵押的 atom 数量不能为零。任何人任何时候都成为验证人，除非当前验证人组的数量超过了最大值。在这种情况下，只有当持有 atom 的数量大于现有验证人中持有有效

atom 数量的最少者，该交易才有效，其中有效 atom 包括受委托的 atom。当一个新的验证人以这种方式替换现有的验证人时，现有的验证人将离线， 其所有的 atom 和受委托的 atom 进入解绑状态。

对验证人的惩罚

对于任何有意或无意的偏离认可协议的验证人，必须对其施加一定的惩罚。有些证据立即可予受理，比如在同样高度和回合的双重签名，或违反 “预投票锁定” (Tendermint 协商一致议定书的规则)。这样的证据将导致验证人失去其良好的声誉，其绑定的 atom 以及在储备池中的比例份额 – 统称为 “权益” – 将被大幅削减。

有时，由于区域网络中断、电源故障或其他原因，验证人将不可用。如果在过去任意时间点的 `ValidatorTimeoutWindow` 块中，验证人的提交投票不包括在区块链中超过 `ValidatorTimeoutMaxAbsent` 次，该验证人将离线，并减少 `ValidatorTimeoutPenalty` (默认 1%) 的权益。

一些 “恶意” 行为在区块链上并没有产生明显的证据。在这些情况下，如果存在多数的协商一致，则验证人可以在带外协调,强制将这些恶意验证人超时。

如果 Cosmos 枢纽 因为超过⅓的投票权离线而出现了中止情况，或者说超过⅓的投票权审查到进入区块链的恶意行为，这时候枢纽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 枢纽验证人可以接受任何种类的代币或组合作为处理交易的费用。每个验证人可自行设置兑换率， 并选择其想要的交易，只要不超过 `BlockGasLimit`，每隔 `ValidatorPayoutPeriod` (默认为 1 小时) 时间会根据权益相关人绑定的 Atom 比例进行分配。

在所收取的交易费用中, `ReserveTax` (默认 2%) 将存入储备池来增加储备量，增加 Cosmos 枢纽的安全性和价值。这些资金也可以按照治理系统的决策进行分配。

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 枢纽的安全取决于底层验证人的安全性和委托人的委托选择。为了鼓励发现和早期报告发现的漏洞, Cosmos 枢纽鼓励黑客通过 `ReportHackTx` 交易发布成功的漏洞，说, “这个验证人被入侵了，请把赏金发送到这个地址”。这种情况下，验证人和委托人将挂起闲置，每个人 `HackPunishmentRatio` (默认 5%) 的 atom 将被削减, `HackRewardRatio` (默认 5%) 的 atom 将发送到黑客的赏金地址作为奖励。验证人必须使用其备份密钥来恢复剩余的 atom。

为了防止这一特性被滥用于转移未授权的 atom，在 `ReportHackTx` 前后， Atom 的比例（授权的与未授权的） 将保持不变，而黑客的赏金将包括一些未授权的 atom (如果有的话)。

治理规范

Cosmos 枢纽是由一个分布式组织管理的，需要一个明确的治理机制，以协调对区块链的各种变化，如系统的参数变量，以及软件升级和宪法修订.

所有验证人负责对所有提案进行表决。如果未能及时对提案进行表决，将导致验证人被自动停用一段时间。 这段时间被称为 `AbsenteeismPenaltyPeriod` (默认 1 周)。

委托人自动继承其委托的验证人的投票权。这一投票可以被手动覆盖掉。而未绑定的 Atom 是没有投票权的。

每个提案都需要 `MinimumProposalDeposit` 代币的保证金，这可能是一个或多个代币 (包括 atom) 的组合。对于每项提案，投票者可以投票表决取走保证金。如果超过半数的投票者选择取走保证金 (例如，因为提案是垃圾信息)，那么保证金就会存入储备池，除了被燃烧的 atoms。

对于每项提案，投票人可以选择下列方案:

- 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。如果大多数人票被否决，那么他们每个人都会失

去 VetoPenaltyFeeBlocks (默认是一天的区块值 ， 税费除外) 作为惩罚，而否决大多数决定的那一方还将额外失去 VetoPenaltyAtoms 默认为 0.1%）的 Atom 作为惩罚。

参数变更提案

这里定义的任何参数都可在 ParameterChangeProposal 通过后改变。

赏金提案

通过 BountyProposal 后， Atom 可以增发和预留储备池资金作为赏金。

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 TextProposal 来协调。

路线图

详见计划.

相关工作

过去几年涌现了很多区块链共识及扩展性方面的创新。在这一部分中将挑选一些重要的创新进行简单分析。

共识系统

经典拜占庭容错

二十世纪八十年代早期就开始研究存在恶意参与者的共识机制，当时 Leslie Lamport 创造了”拜占庭容错”这个词，用来指那些图谋不轨参与者做出的恶意的行为，与”死机故障”不同，后者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，网络信息滞后有一个上限，但实际使用是在高度受控的环境下进行，比如精密飞行仪器以及使用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（ Practical Byzantine Fault Tolerance ,PBFT）[11]才作为有效的、部分同步的共识算法被逐步推广。它可以容忍⅓参与者有恶意行为。PBFT 成为标准算法，催生了各种版本，包括最近由 IBM 提出并使用于 Hyperledger 超级账本中的算法。

和 PBFT 相比，Tendermint 共识的主要好处在于其改善且简化了的底层结构，其中有些是遵循了区块链典范的结果。Tendermint 中，区块必须按顺序提交，这就消除复杂性，节省 PBFT 中状态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，那么就不能让它之后的区块 N+i（i>=1）提交。如果是通信带宽限制导致了区块 N 未提交到 Cosmos Zone 上，那么将通信带宽用于分享选票给区块 N+i 是一种浪费。如果由于网络分区或者节点掉线导致的区块 N 未提交，那么 N+i 就无论如何也不能提交。

此外，将交易打包成块可以用默克尔哈希纪录应用程序的状态，而不是用 PBFT 检查机制进行定时摘要。这可以让轻客戶端更快的提交交易证明，以及更快的跨链通信。

Tendermint Core 中也优化了很多 PBFT 特性以外的功能。比如，验证人提交的区块被分割多个部分，对其默克尔化后，然后在节点间广播。这种方式可以提

高其广播性能（具体请查看 LibSwift 19）。而且，Tendermint Core 不会对点对点连接做任何假设，只要点对点间的网络不断开，那么它就能正常运行。

BitShare 委托权益

BitShares [12]不是第一个采用权益证明机制（proof-of-stake,PoS）的区块链，但是其对 PoS 在区块链上的研究与推进做出了巨大的贡献，尤其是在 DPoS，即受委托权益证明方面。在 BitShares 中，相关方选择”见证者”负责提交交易顺序并提交；相关方选择”委托人”负责协调软件更新与参数变化。尽管 BitShare 在理想环境下能达到很高的性能：100k tx/s，1 秒的滞后。每一块只有一个单独的签名，得到交易的最终性的时间比区块时间略长。一个标准的协议仍在开发中。利益相关者可以每天去除或者替换有恶意行为的验证人，但是不同于 Tendermint PoS 的保证金机制，BitShares 没有要求验证人或者代理人的提交押金，如果发生双花攻击的话，押金不会被削减。

Stellar

Stellar [13]是以 Ripple 推行的解决方案为基础，它优化了联邦拜占庭协议模型，其中参与共识过程的并不构成一个固定的全局过程。 相反，每个进程节点组织一个或多个“仲裁片”，每个“仲裁片”构成一组可信进程。 Stellar 中的“法定人数”被定义为一组包含至少个个节点的一个仲裁片。从而可达成一致。

恒星机制的安全性依赖于任何两个仲裁的交集都是非空的假设，同时，节点的可用性要求至少一个“仲裁片”完全由诚实节点组成。这就需要在“法定人数”的大小上作出妥协：人数过少难以获得共识，人数过多则难以信任所有人。可能

难以平衡而不对信任做出重大假设。 除此之外，节点必须维护一定数量的仲裁片以获得足够的容错能力（或者任何“完整节点”，大部分结果的依赖），并且提供一种层级化的配置策略，类似于边界网关协议（Border Gateway Protocol,BGP）。BGP 被互联网服务供应商（Internet Service Provider，ISP）用来建立全球路由表，也被浏览器用来管理传输层安全协议（Transport Layer Security，TLS）证书。他们都因为不安全而臭名昭着。

对于 Stellar 论文中基于 Tendermint 的 PoS 批评可以通过本文描述的代币策略来缓解。本文提出了名为 *atom* 的新的代币，它代表未来交易过程中产生的费用和奖励。 基于 Tendermint PoS 的优势在于其原理相对简单，同时仍可充分保证和证明安全性。

BitcoinNG

BitcoinNG 是对比特币的改进，允许垂直扩展，比如增加块的大小而避免带来负面的经济后果，例如对矿工造成的影响严重不成比例。 这种改进是通过将 leader 选举与交易广播分开来实现的：leader 首先通过“微块 micro-blocks”的 PoW 选举，然后 leader 能够广播交易直至下一个新的“微块”。 这减少了赢得 PoW 比赛所需的带宽要求，使矿主竞争更公平，并通过允许最后一名矿工提交微块以加快交易提交频率。

Casper

Casper [16]是以太坊提出的 PoS 共识算法。 它的主要操作模式是“预测押注”的一致。 通过让验证者基于他们目前看到的其他投注来迭代地下注他们认为哪个块将提交入区块链。最终性可以立即实现。

链接. 这是 Casper 团队研究的一个热点领域。 挑战在于构建一个可以证明是一个演化稳定策略的投注机制。 与 Tendermint 相比，Casper 的主要优势可能在于提供“可用性超越一致性”—达成共识不需要超过 50％的投票权—可能是以提交速度或实现复杂性为代价的。

水平扩展

Interledger 协议

Interledger 协议（The Interledger Protocol，ILP）[14]不是一种严格的扩展方案。 它通过一个松散耦合的双边关系网络，提供一种指定的跨不同账本系统交互操作。像闪电网络一样，ILP 的目的是实现支付，但是它特别关注跨账本类型的支付，并扩展原子事务的处理机制，使得事务的处理不仅支持哈希锁，而且还包括法定人数的公证人（称为原子运输协议）。 后者在账本间交易中实施原子性的机制与 Tendermint 的轻客户 SPV 机制类似，因此比较 ILP 和 Cosmos / IBC 之间的区别是有必要的，具体见下文。

1.ILP 不支持连接器公证员的变更，也不允许公证员之间有灵活的权重。 另一方面，IBC 是专门为区块链设计的，验证人可以拥有不同的权重，并且随着区块链的发展，成员可以随时更改。

2.与闪电网络一样，ILP 中的接收人必须在线才能向发起人发送确认。 在 IBC 代币传输中，接收者所在区块链的验证人集合负责提供确认，而不是接收用户本人。

3.最大的不同在于 ILP 连接器不需要负责对支付状态保持权威性，然而在 Cosmos 中，hub 的验证人负责 IBC 代币传输状态和每个 zone 内持有代币数量的权威性。允许从 zone 之间的安全地不对称地交换代币是本质的创新。在 cosmos 中的 ILP 连接器可以看作是一个持久和最安全的区块链分类账：cosmos hub。

4.ILP 内的跨账本支付需要一个交易所的指令集的支持。因为不存在从一个分类账到另一个分类账不对称的代币转移，只能做到市场等价物的转移。

侧链

Sidechains [15]是一种通过使用与比特币区块链“双向挂钩”替代区块链来扩展比特币网络性能的机制。（双向挂钩相当于桥接，在 cosmos 中被称为“桥接”与市场挂钩区分）。侧链使得比特币可以方便的在比特币区块链和侧链间移动，并允许在侧链上实验新功能。在 Cosmos Hub 中，侧链和比特币是彼此的轻客户端，在比特币区块链和侧链间移动时使用 SPV 证明。当然，由于比特币使用 PoW，以比特币为中心的侧链遭受许多由于 PoW 作为共识机制的引起的问题和风险。而且，这是一个比特币利益最大化的解决方案，并不像 Cosmos 那样本地支持各种代币和 zone 间网络拓扑结构。但是，双向挂钩的核心机制原则上与 Cosmos 所采用的机制相同。

以太坊扩展性的努力

以太坊目前正在研究许多不同的战略，将以太坊区块链的状态分区化，以解决可扩展性的需求。 这些努力的目标是在共享状态空间之上，维持当前以太坊虚拟机提供抽象层。 目前，多项研究工作正在进行。[18] [22]

Cosmos vs Ethereum 2.0 Mauve

Cosmos 和 Ethereum 2.0 Mauve [[22]][22 有不同的设计理念。

- Cosmos 是针对代币儿 Mauve 是关于扩大计算能力。
- Cosmos 不仅限于 EVM, 所以即使不同的 VM 也可以交互。
- Cosmos 让 zone 的创建者决定验证人。
- 任何人都可以在 Cosmos 中建立新的 zone（除非管理者另做决定）。
- hub 与 zone 的失效隔离，所以全局的代币不变量可以保持。

普遍扩展

闪电网络

闪电网络被设计成一种代币传输网络，在比特币区块链（及其他公有区块链）上一层运行，通过把大部分交易从共识分类账之外转移到所谓的“ 付款渠道“。这通过链上加密货币脚本来实现，这些脚本使双方能够进入双方持有的状态化合同，通过共享数字签名来更新状态，并且在合同结束后最终通过在区块链上发布证据，这种机制首先受到跨链原子互换交易的欢迎。 通过与多方开通支付渠道，闪电网络的参与者可以成为集中点，为其他人的支付提供路由，从而导致支付渠道网络的完全联通，其代价是绑定在支付渠道上的资金。

虽然闪电网络也可以轻松地跨越多个独立的区块链，并借助交易市场实现价值转移，但它不能实现从一个区块链到另一个区块链的非对称代币交易。 这里描述的 Cosmos 网络的主要优点是实现直接的代币交换。 也就是说，我们希望支

付渠道和闪电网络将会与我们的代币传输机制一起被广泛采用，从而节省成本和保护隐私。

隔离验证人

隔离见证是一个比特币改进建议 BIP，链接 旨在将每块交易吞吐量提高 2 倍或 3 倍，同时使新节点能更快同步区块。 这个解决方案的亮点在于它如何在比特币当前协议的局限下，允许软分叉升级（例如，具有旧版本软件的客户端将在升级后仍可继续运行）。 Tendermint 作为一个新的协议没有设计的限制，所以它具有不同的扩展优先级。 TendermintBFT 的循环算法，主要基于加密签名而不是采矿，这种算法允许通过多个并行区块链进行水平扩展，而更常规、更频繁的区块提交也允许垂直扩展。

附录

分叉问责制

经过精心设计的共识协议应该能够在超出容错能力或者共识出错的情况下为系统提供一定的保障。这在能够通过拜占庭行为获取实质经济回报的金融系统中显得尤为必要。*分叉问责制*就属于这种非常重要的保障机制，这种制度会使一个导致共识出错的进程（例如使协议客户端开始接受不同值——即分叉）被识别出来，并且根据协议规则对其进行惩罚，甚至将其移送至司法系统处置。但当司法系统不可靠或者诉讼费极其昂贵时，为了让验证人们都参与到这个机制当中，该制度会强制要求他们建立安全保证金，一旦检测到恶意行为，那么这些保证金将会被罚没或者削减[10].

注意这与比特币有所不同，由于网络非同步与局部哈希碰撞的概率特性，比特币的分叉是定期出现的。因为在很多情况下，恶意分叉与非同步引起的分叉是无法分辨的，除非让矿工为挖到的孤立区块支付隐性的机会成本，否则比特币就很难准确地执行分叉问责制。

Tendermint 共识

我们把投票阶段分为 *预投票* 和 *预提交* 两个阶段。一个投票可以是用于特定区块，也可以用于 *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*。如果在同一轮中对 Nil 的预提交超过⅔，他们就进入到下一轮中。

注意到协议中的严格的确定性会引发一个弱同步假设，因为错误的发起者必须被检测到并将其略过。验证人们在为 Nil 预投票之前会等待一段时间，称之为超时提议，这个超时提议的 hi 等待时间也会随着每一轮的进行而递增。每一轮的进行都是完全不同步的，在这个过程中，只有当验证人收听到超过⅔的网络投票才能进入到下一轮中。实际上，它需要极其强大的阻碍才能阻挠这个弱同步假设（导致无达成共识，无法提交区块），而且通过每个验证人超时提议（TimeoutPropose）的随机值还可更加提高这么做的难度。

另一个附加的约束条件，或者叫锁定条例，它能确保网络最终在每个高度只提交一个区块。任何试图在给定高度提交超过一个区块的恶意行为都会被识别出来。首先，一个区块的预提交必须被认为是正当的，并且以 Polka 的形式提交。如果验证人已经准备在 R_1 轮中预提交一个区块，我们称他们锁定了这个

区块，并且然后用于验证 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。其次，验证人们必须为他们锁定的区块提议并且（或者）预投票。这两个条件共同作用，确保了验证人在对其正当性没有充分论证的情况下不能进行预提交操作，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投票。这样不但可以保证共识算法的安全性，还能保证它的活跃度。

关于这一协议的全部细节参考这里.

Tendermint 轻客户端

因为生成侧链（一个分叉）意味着至少⅓的担保权益被罚没，Tendermint 权益证明（Tendermint-PoS）取消了同步所有区块头的要求。当然，罚没保证金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。另外，轻客戶端可以定期地与验证人组的改变保持同步，以避免出现远程攻击（但是其他解决方案也具有可能性）。

与以太坊类似，Tendermint 能够让应用程序在每个区块中嵌入一个全局默克尔根的哈希值，可以简单方便的验证的状态查询，比如查询账户余额、在智能合约中的值，或者未使用交易输出（UTXO）的存在，具体由应用程序的特性决定。

远程攻击的防御

假设有一个足够有弹性的广播网络采集与一组静态验证人组存在，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罚没。这个由 Vitalik Buterin 在 2014 年首次提出的新方法，解决了其他权益证明加密货币的”没有任何相关权益”问题（详见 相关工作部分）。但由于验证人组必须是能够变化的，在较长的时间段内最初的一些验证人会解除 j 押金绑定，这就使他们可以自由地从创世区块中创建新链，并且因为他们不再有被锁定的保证金，他们将不需要为这个行为支付任何费用。这类攻击被称为远程攻击（LRA），与短程攻击相比，后者对处在押金绑定中的验证人发起的分叉是可以对其进行惩罚的（假设有类似 Tendermint 共识这样的分叉问责制拜占庭容错算法）。所以远程攻击经常被认为是对权益证明机制的一种危险打击。

幸运的是，远程攻击（LRA）可以用以下的途径来缓解。第一，对于解除绑定的验证人而言（取回抵押保证金并且不再从参与共识中获取费用），保证金在一定时间内不能转移，也可以称其为”解绑周期”，这个周期可能长达数周或数月。第二，对于轻客户端的安全性而言，其首次连接到网络时必须根据可信源验证最新的一个区块哈希或者多个最好的区块哈希。这种情况有时被称为”弱主观性”。最后，为了保证安全，必须与最新的验证人组进行频繁的同步，时长与解绑周期一致。这样就确保了轻客戶端在因验证人解绑资金而失去任何权益之前，知道验证人组的变化情况，否则解绑的验证人就会在其绑定的高度后面开始创建新区块来实施远程攻击，以此来欺骗客户端（假设它可以控制足够多的早期私钥）。

注意到用这样的方式来对抗远程攻击（LRA）需要对工作量证明（proof-of-work）的原始安全模块进行彻底检查。在工作量证明中（PoW），一个轻客户端可以通过在每一个区块头中运行工作量证明，以此简便地在任何时候与可信任的创始区块的当前高度进行同步。但是，为了对抗远程攻击（LRA），我们需要轻客戶端定期上线追踪验证人组的变动，其中在首次上线时必须格外仔细地根据可靠源来验证从网络采集的信息。诚然，后面这个要求和比特币类似，其协议和软件也必须从可靠源获得。

以上这些为了防止远程攻击的方法，比较好地适用于由 Tendermint 驱动下的区块链验证人节点以及全节点，因为这些节点需要保持与网络的连接。这些方法同样适用于希望频繁地与网络同步的轻客户端。但是，对于那些不希望频繁接入互联网或者区块链网络的轻客户端来说，还有另一种方法可以解决远程攻击的问题。非验证人节点可以在很长的解绑期内（比如比验证人的解绑期更久）使用代币作为保证金，并且为轻客戶端提供二级证明当前有效性以及过去区块哈希的解决方案。虽然这些代币对于区块链共识的安全性并没有价值，不过他们还是可以为轻客戶端提供强大的保障。如果在以太坊中支持历史区块哈希查询，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。

克服分叉与审查攻击

由于提交区块流程的定义，任何联合后不少于⅓的投票权的节点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些交易的区块来审查特定的交易，尽管这将导致大多数区块提案被拒绝，致使区块

提交速率减缓，降低了它的实用性与价值。恶意的联合或许仍然会陆陆续续地广播选票，用阻挠区块链的区块提交来将其逼停，或者使用任何这些攻击的组合攻击。最终，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作恶者也参与进来，就会用可能出现错误的验证组人子集导致速度降低的方法来分割网络。这不只是 Tendermint 面临的局限性，更确切地说是所有被活跃敌对者控制了网络的共识协议所面临的局限性 1。

对于这些类型的攻击，验证人的子集应该通过外部的方式进行协调，以签署选择一个分叉（及关系到的所有证据）与带有签名的验证人的初始子集的重组提案。签署了这样一份重组提案的验证者，将放弃在所有其他分叉上属于他们的保证金。客户端应在重组提案中验证签名以及任何相关的证据，并作出判断或提示终端用户作出决定。例如，一个手机钱包 app 应该在可能接受任何由一半以上的初始验证人们通过投票权利签署的重组提案时，给予用户安全警告提示。

当多于⅓的投票权益是不诚实的时候，一个非同步的拜占庭容错算法步能够达成共识，然而，分叉假设不少于⅓的投票权益已经因不正当的双重签名或者锁定改变而成为不诚实的。因此，签署重组提案是一个协调问题，任何非同步协议都无法解决这个问题（也就是自动的，并且不考虑底层网络的可靠性）。目前，我们通过互联网媒体的社会共识，把重组提案的协调问题留给了用户去协调。验证人必须在签署重组提案之前就确保没有出现网络分割的问题，以此来避免签署两个相冲突的重组提议的情况发生。

假设外部协调媒介和协议是可靠的，对于分叉的担心会比审查攻击要少许多。

除了需要大于⅓的拜占庭投票权益才能启动的分叉和审查制度以外，超过⅔的联合投票权益可能会提交任意、无效的状态。这是任何拜占庭容错算法的共识系统所特有的问题。与利用简单可验证证明来创建分叉的双重签名不同，检测无效状态的提交需要非验证节点来验证整个区块，这意味着非验证节点会保留一份本地的状态副本并执行每一笔交易，然后为他们自己独立计算出状态的根源。一旦检测出来，处理这类故障的唯一方法就是社会共识。打一个比方，在比特币出现问题的情况下，无论是由于软件漏洞造成的分叉（正如 2013 年 3 月），还是由于矿工拜占庭行为提交的无效状态（正如 2015 年 7 月），由商户、开发者、矿工和其他组织组成的联系紧密的社区所建立起来的社会共识会让他们按照分工来参与到修复网络的工作当中去。此外，由于 Tendermint 区块链的验证人身份是可识别的，那么如果需要的话，无效状态的提交实际上是可以被法律或其他外部法律体系惩治的。

ABCI 说明

ABCI 由 3 种主要的信息类型组成，这三类信息从共识引擎传递到应用程序上，然后应用程序用相应回复信息做出应答。

`AppendTx` 信息是应用程序的主要传递媒介。区块链中的每一笔交易都通过这个信息来传递。应用程序需要验证每笔交易，这将通过接收针对当前状态、应用协议和交易密码证书的 AppendTx 信息来实现。验证过的交易将需要通过添加数值到键值存储或者更新 UTXO 数据库的方式来更新应用状态。

`CheckTx` 信息与 AppendTx 信息类似，但它只是为了交易验证。Tendermint Core 的内存池会先用 CheckTx 验证交易有效性，并且只会将有效的交易传递给其他的节点。应用程序会检查交易序列号，如果序列号过期就会根据 CheckTx 返回一个错误。

`Commit` 信息是用来计算之后会存入到下一区块头中的当前应用状态的加密提交项。这具有便利的特性。状态的前后矛盾性将会像引起程序错误，从而导致区块链分叉。这也简化了安全轻客戶端的开发，因为默克尔哈希证明可以通过检查区块哈希来加以验证，而区块哈希是由规定人数的验证人们签署的（通过投票权益）。

此外，ABCI 信息允许应用程序保持追踪验证人组的改变，并让应用程序接收诸如高度和提交选票之类的区块信息。

ABCI 请求/响应是简单的 Protobuf 信息。请参考这里的模式文件。

AppendTx

- **命令行参数**:
  - `Data ([]byte)`: 交易请求信息
- **Returns**:
  - `Code (uint32)`: 回复代码
  - `Data ([]byte)`: 结果字节，如果有的话
  - `Log (string)`: 错误信息
- **使用**:
  提交并执行一笔交易，如果交易有效，那返回 CodeType.OK

CheckTx

- **命令行参数**:
  - `Data ([]byte)`: 交易请求信息
- **Returns**:
  - `Code (uint32)`: 回复代码

    - `Data ([]byte)`: 结果字节，如果有的话
    - `Log (string)`: 错误信息
- **使用**:
  验证一笔交易。这个信息不应该改变应用状态。交易在广播给其他节点前，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，以允许序列执行同一区块中相关的交易。

Commit

- **返回值**:
    - `Data ([]byte)`: 默克尔根值
    - `Log (string)`: 调试或出错信息
- **使用**:
  返回当前应用状态。

Query

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

Flush

- **使用**:
  刷新回复队列。应用 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

- **返回值**:
    - `Data ([]byte)`: 信息字节串
- **使用**:
  返回关于应用程序状态的信息. 应用指定。

SetOption

- **参数**:
    - `Key (string)`: 设置参数
    - `Value (string)`: 参数值
- **返回值**:
    - `Log (string)`: Debug or error message

- **使用**:
  比如，针对内存池的连接可以将键设置为”mode”（模式），值为”mempool”（内存池）。或者针对共识连接，将键设置为”mode”，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

- **参数**:
  - `Validators ([]Validator)`：初始化创世验证人
- **使用**:
  在创世区块创建时调用

BeginBlock

- **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。在附加交易（AppendTxs）前进行调用。

EndBlock

- **参数**:
  - `Height (uint64)`：结束时的区块高度
- **返回值**:
  - `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。

See the ABCI repository for more details

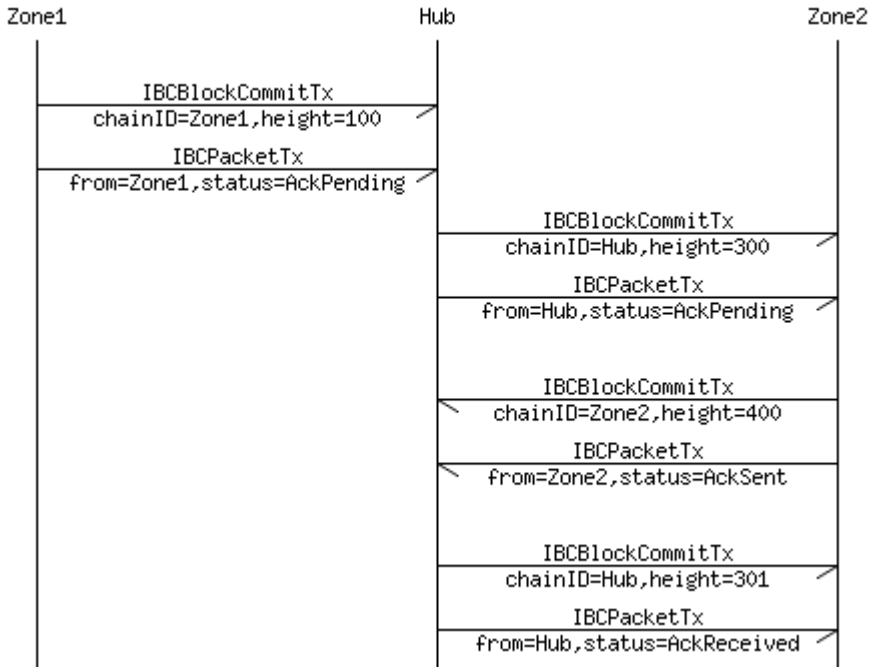
更多细节请参考 ABCI 知识库。

IBC 数据包交付确认

发送者有很多需要接收链提供数据包交付确认的原因。比如，如果预计目的链会出错，那发送者就可能无法了解目的链的状态。或者，当目的链可能遇到因

接收数据包猛烈增多而形成的拒绝服务攻击时，发送者会想要设定数据包超时时限（借助 MaxHeight 即最大值包域）。

在这些案例中，发送人可以通过在 AckPending 上设置初始数据包状态来要求提供交付确认。然后就由接收链通过包含一个简化的 IBCPacket 的应用默克尔哈希来确认交付。



首先，一个 IBCBlockCommit 和 IBCPacketTx 是被上传到“枢纽”上用来证明”分区 1″上的 IBCPacket 的存在的。假设 IBCPacketTx 的值如下：

- FromChainID: “Zone1”
- FromBlockHeight: 100 (假设)
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
    - SrcChainID: “Zone1”
    - DstChainID: “Zone2”
    - Number: 200 (say)
    - Status: AckPending
    - Type: “coin”
    - MaxHeight: 350 (假设 “枢纽” 当前高度为 300)

    - Payload: <一个"代币"的有效负荷字节>
- FromChainID: "Zone1"

- FromBlockHeight: 100 (假设)
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
    - SrcChainID: "Zone1"
    - DstChainID: "Zone2"
    - Number: 200 (假设)
    - Status: AckPending
    - Type: "coin"
    - MaxHeight: 350 (假设"枢纽"目前的高度是 300)
    - Payload: <一个"代币"的有效负荷字节>

其次，一个 IBCBlockCommit 和 IBCPacketTx 被传输都"分区 2"上用来证明 IBCPacket 在"枢纽"上的存在。假设 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: <一个"代币"相同的有效负荷字节>
- FromChainID: "Hub"

- FromBlockHeight: 300
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
    - SrcChainID: "Zone1"
    - DstChainID: "Zone2"
    - Number: 200
    - Status: AckPending
    - Type: "coin"
    - MaxHeight: 350

    - Payload: <一个"代币"相同的有效负荷字节>

接下来，"Zone2″必须将缩写的来显示 AckSent 的最新状态包添加到应用程序状态哈希中。

IBCBlockCommitand 和 IBCPacketTx 会传输到"枢纽"上来证明简化的 IBCPacket 存在于"分区 2″上。假设 IBCPacketTx 的值如下：

- FromChainID: "Zone2"
- FromBlockHeight: 400 (假设)
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
    - SrcChainID: "Zone1"
    - DstChainID: "Zone2"
    - Number: 200
    - Status: AckSent
    - Type: "coin"
    - MaxHeight: 350
    - PayloadHash: <一个"代币"相同的有效负荷字节的哈希值>
- FromChainID: "Zone2"

- FromBlockHeight: 400 (假设)
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
    - SrcChainID: "Zone1"
    - DstChainID: "Zone2"
    - Number: 200
    - Status: AckSent
    - Type: "coin"
    - MaxHeight: 350
    - PayloadHash: <一个"代币"相同的有效负荷字节的哈希值>

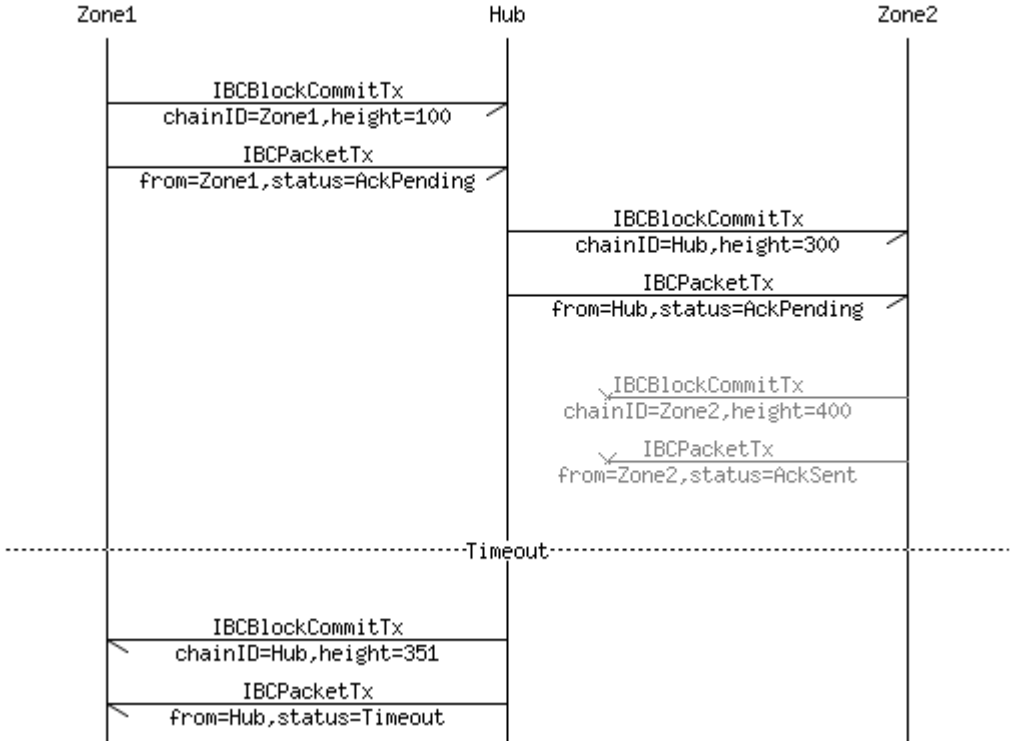
最后，"枢纽"必须更新从 AckPending 到 AckReceived 的数据包状态。这个新完成状态的证明应该返回到"分区 2"上。假设 IBCPacketTx 的值如下：

- 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>
- FromChainID: “Hub”

- FromBlockHeight: 301
- Packet: an IBCPacket:
    - Header: an IBCPacketHeader:
    - SrcChainID: “Zone1”
    - DstChainID: “Zone2”
    - Number: 200
    - Status: AckReceived
    - Type: “coin”
    - MaxHeight: 350
    - PayloadHash: <相同”代币”有效负荷的哈希字节>

与此同时，“分区 1”会假设“代币”包的交付已经成功，除非”枢纽”上有证据给出相反的证明。在上述例子中，如果”枢纽”没有从”分区 2″接收到第 350 个区块的 AckSent 状态，那么它就会自动将其设置为 Timeout（超时）。这个超时的证据可以贴回到”Zone1″上，然后所有代币都会被返还。



默克尔树及默克尔证明的说明

Tendermint/Cosmos 生态支持的两种默克尔树：简单树和 IAVL+树。

简易版默克尔树

简易版默克尔树针对基础的静态列表。如果项目的数量不是 2 的次方，那么有些树叶就会在不同的层上。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。这种默克尔树就是用于一个区块交易的默克尔化的，而顶层元素就是应用状态的根。

```
1.               *
2.              / \
3.             /   \
4.            /     \
5.           /       \
6.          *         *
7.         / \       / \
```

```
      /   \          /   \
     /     \        /     \
      *       *       *      h6
     / \     / \     / \
    h0  h1  h2  h3  h4  h5

A SimpleTree with 7 elements
```

IAVL+树

IAVL+数据结构的目的是永久储存应用状态中的密钥对，这样就可以对确定的默克尔根哈希进行高效的运算。这个树的平衡通过 AVL 算法的变体来实现，所有运行都是 O(log(n))。

在 AVL 树中，任意节点的两个子树的高度至多有一处不同。无论在什么时候这种情况都是与更新相违背的，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。在初始的 AVL 算法中，内部节点也可以保留密钥值对。AVL+算法（注意这里有个”+”号）对 AVL 算法进行了修改，来维持所有数值都在树叶节点上，同时还只需采用分支-节点来存储密钥。这样在维持较短的默克尔哈希轨迹对的同时，还简化了算法。

AVL+树类似于以太坊的帕氏树。其中也有一定的折中。密钥不需要在嵌入到 IAVL+树之前生成哈希，所以这就为密钥空间提供了较快的命令迭代，这会为很多应用程序带来好处。逻辑实现更简单，只需要内部节点和树叶节点这两种节点类型。作为一个平衡的二叉树，其默克尔证明平均更短。而另一方面，IAVL+树的默克尔根有取决于命令的更新。

我们将支持额外有效的默克尔树，比如当二元变量可用时的以太坊帕氏树。

交易类型

在标准的执行中，交易通过 ABCI 界面涌入 Cosmos Hub 的应用程序中。

Cosmos Hub 将接收几类主要的交易类型，包括

`SendTx`, `BondTx`, `UnbondTx`, `ReportHackTx`, `SlashTx`, `BurnAtomTx`, `ProposalCreateTx`，以及 `ProposalVoteTx`（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，创建提案交易、以及提案投票交易），这些都不需要加以说明，会在未来版本的文档中加以备案。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 数据包交易）

IBCBlockCommitTx

IBCBlockCommitTx 交易主要由这些组成：

- `ChainID (string)`: 区块链 ID
- `BlockHash ([]byte)`: 区块哈希字节，就是包括应用程序哈希默克尔根
- `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的包括区块提交的 Tendermint 预提交投票
- `ValidatorsHash ([]byte)`: 新验证组的默克尔树根哈希
- `ValidatorsHashProof (SimpleProof)`: 在区块哈希中证明验证人哈希的简易树默克尔证明
- `AppHash ([]byte)`: IAVL 树，应用程序状态的默克尔树根哈希
- `AppHashProof (SimpleProof)`: 在区块哈希中验证应用程序哈希的简易版默克尔树证明 `AppHash` against the `BlockHash`

IBCPacketTx

`IBCPacket` 由下列项组成：

- `Header (IBCPacketHeader)`: 数据包头

- `Payload ([]byte)`：数据包有效负荷字节。可选择。
- `PayloadHash ([]byte)`：数据包字节哈希。可选择。

有效负荷 `Payload` 或有效负荷哈希 `PayloadHash` 必须存在一个。`IBCPacket` 的哈希就是两个项的简易版默克尔根，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)`：所有数据包的唯一数字

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。

An `IBCPacketTx` transaction is composed of:

- `FromChainID (string)`: The ID of the blockchain which is providing this packet; not necessarily the source
- `FromBlockHeight (int)`: The blockchain height in which the following packet
  is included (Merkle-ized) in the block-hash of the source chain
- `Packet (IBCPacket)`: A packet of data, whose status may be one of `AckPending`, `AckSent`, `AckReceived`, `NoAck`, or `Timeout`
- `PacketProof (IAVLProof)`: A IAVLTree Merkle-proof for proving the packet’s
  hash against the `AppHash` of the source chain at given height

`IBCPacketTx` 交易有下列项组成：

– `FromChainID (string)`：提供给这个数据包的区块链 ID，不是源所必须的

– FromBlockHeight (int)：区块链高度，其中接下来的包会包含在（默克尔化的）源链的区块哈希中

– Packet (IBCPacket):数据包，其状态可能是 AckPending, AckSent, AckReceived, NoAck，或者 Timeout 其中的一个

– PacketProof (IAVLProof)：IAVL 树默克尔证明，用于验证在给定高度下的源链应用哈希中的数据包哈希

通过”Hub”，将数据包从”Zone1″发送到”Zone2″的序列，描述在{Figure X}函数中。首先，一个 IBCPacketTx 会向”Hub”证明数据包是包含在”Zone1″的应用程序状态中。然后，另一个 IBCPacketTx 会向”Zone2″证明数据包包含在”Hub”的应用程序状态中。在这个过程中，IBCPacketTx 的字段是相同的：SrcChainID 永远是”Zone1″，而 DstChainID 永远是”Zone2″。

The PacketProof must have the correct Merkle-proof path, as follows:

PacketProof 必须有正确的默克尔证明路径，如下：

1. IBC/<SrcChainID>/<DstChainID>/<Number>
2.

当“Zone1”要通过“Hub”将数据包传送到“Zone2”中，无论数据包是否在“Zone1”、“Hub”、或者“Zone2”中默克尔化了，IBCPacket 数据都是相同的。唯一易变的字段是为追踪交付的 Status。