目 录

[1. 共识算法 1](#_Toc38576475)

# 共识算法

Hacker Speaker

【共识算法系列】第一讲即将开始！

—————————————

主讲人：Haichao

Algorand基金会Associate Director

曾任微软、Nervos、东京大学研究员

—————————————

分享主题：共识算法入门

—————————————

分享主题大纲：

\* 什么是 BFT 共识算法

\* BFT 共识算法的分类

\* 常见 BFT 共识算法

\* 区块链的安全性

共识算法（Consensus Algorithm)是保证区块链可信度和安全性的重要技术。目前，有多种主流的共识算法，如工作量证明（Proof of Work, PoW）、权益证明（Proof of Stake, PoS）等等，各有其优、缺点与适用对象。共识算法基于分布式系统的概念，区块链系统即是分布式系统中最典型的一种模型——状态复制机。每一个区块链节点的本质就是一个状态机（State Machine），多个节点互相连接构成网络。共识算法的出现，就是为了使一个状态复制机网络能够保持正常且可靠运行。因此，在区块链技术领域，不可小觑共识算法的重要地位。

在本期Hacker Speaker中，我们邀请到Algorand副总监Haichao，为我们带来本次分享

朱海潮

Algorand 基金会的 Associate Director，主要负责开发者社区和生态系统的建设。海潮曾在 Nervos 项目任数字货币研究员和产品经理职务，并在微软和日本东京大学等研究机构担任研究员

其实咱们中文社区对于共识算法其实一直都有很深的误解。这主要也是由于前两年各种群魔乱舞的 XPoX 系列（见下图），都宣称自己是这个星球上最先进最好的共识算法，搞得大家非常 confused。



而这些 XPoX 显然并不是共识算法。

如果严格来说的话，大家熟知的 PoS 其实也不是共识算法，甚至连 PoW 都不能算是共识算法。

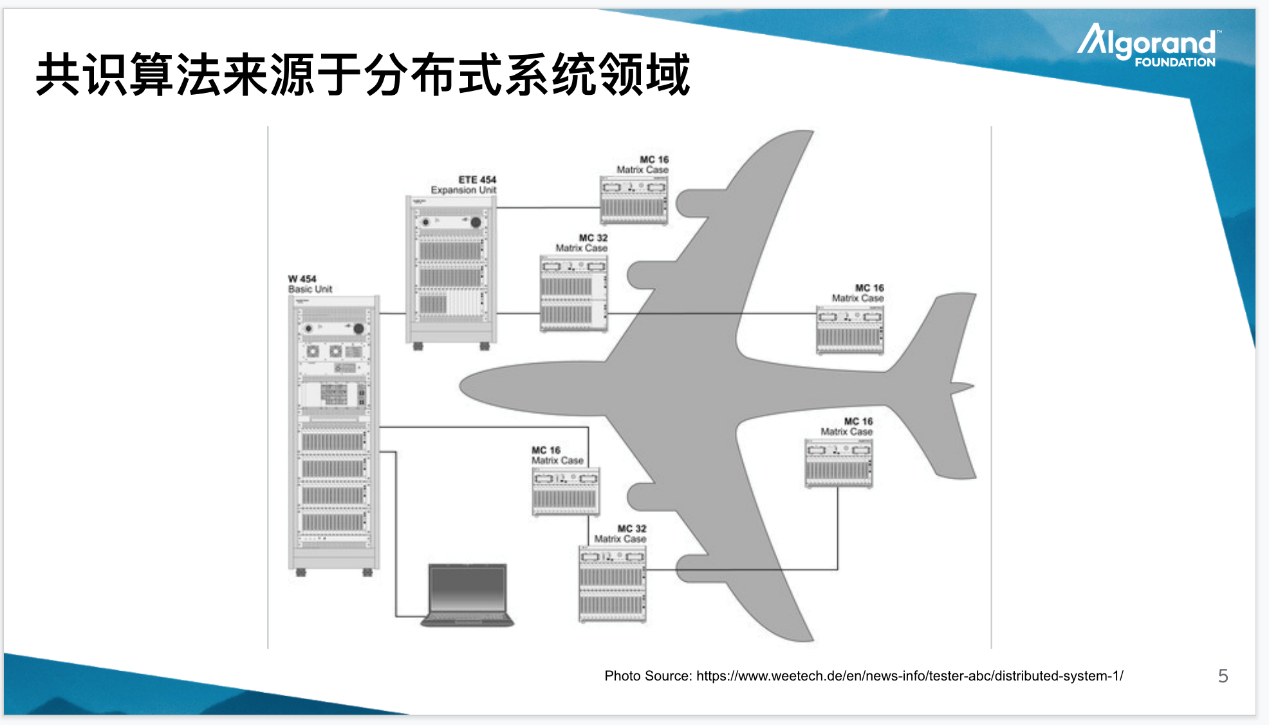
那究竟什么才是共识算法呢？我们今天就来好好的聊一下这个话题。

最早使用共识算法这个名词的其实是来自分布式系统领域的科学家们。

所以这里我们首先需要理解下分布式系统这个概念。

简单来说，分布式系统指的是一组计算机同时工作，并且之间互相通信，完成同一个计算目的。虽然这会是一个由多台计算机组成的系统，但对于使用它的用户来说，它用起来就像是一台计算机一样。

一个典型的例子就是飞机系统。



一架飞机中有着多台计算机系统，他们互相通信协作，共同完成带你飞来飞去这一伟大目标。

在一个分布式系统中，我们通常会将其中的一个个计算机称为节点（node）。

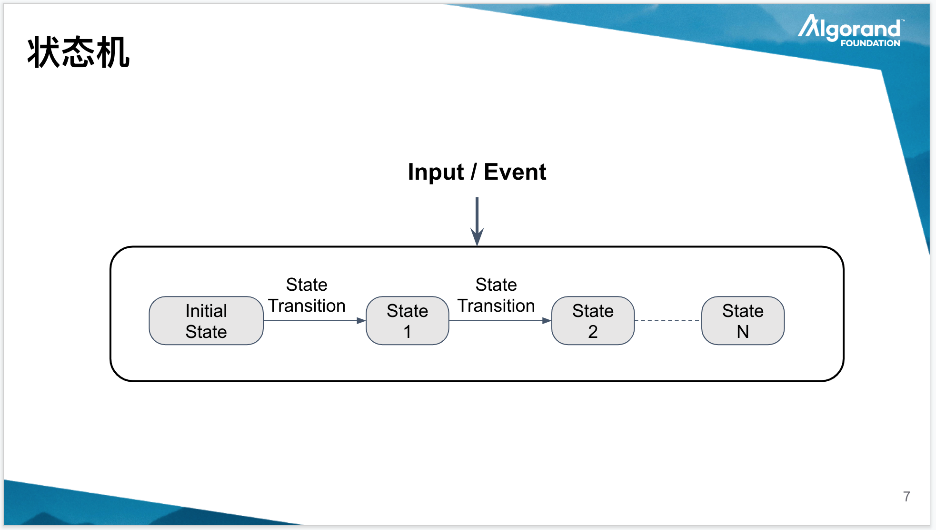
节点之间会互相通过发送消息（message）来进行通信，消息中会包含需要节点执行的事务（event），节点在收到事务后会独立进行所需的计算，并更新本地保存的结果，或将结果发送给客户端（Client），这就构成了一个分布式系统。

很明显，区块链系统也是一种分布式系统。

区块链系统本身使用的模型，也是分布式系统中最典型的一种模型 - 状态复制机（Replicated State Machine）

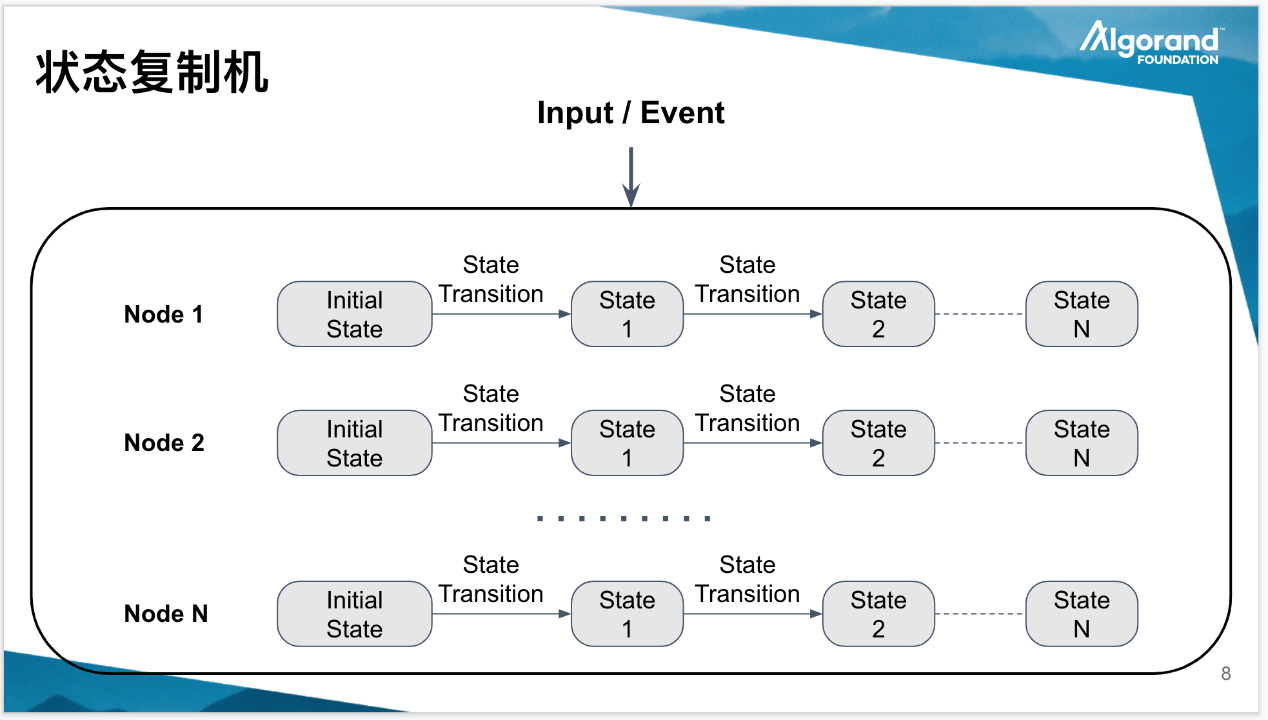
先给非 CS 背景的朋友们简单讲下什么是状态机（State Machine）。

状态机是一种抽象后的计算机模型，它指的是一台能够储存一组状态（State）的计算机，在收到来自外部的输入（Input）或者事件（Event）时，它会发生状态的转变（State Transition），如上图所示。



还是用区块链举个例子，每一个区块链节点本质上就是一个状态机，它保存了当前所有账户的余额等信息，而这些信息就可以被看作是状态；当节点收到外界发来的交易（Transaction）时，它就会根据交易信息去计算每个账户的新余额；计算完成后，节点会将新的账户余额存储下来，成为新的状态，这也就完成了状态的转变。

而由多个这样的节点互相连接构成的网络就叫做状态复制机。



在正常情况下，当收到外界发来的交易时，状态复制机网络中的所有节点都会按照交易信息独立计算，并存储新的状态；并且多个节点的最终状态会保持一致，同时对外提供相同的输出；虽然网络中存在多个状态机，但从外界来看，就像是在使用一个单独的状态机一样。

而为了使一个状态复制机网络能保持正常并且可靠运行的算法，就叫做共识算法（Consensus Algorithm）。

背景知识介绍到这里，我们就可以给共识算法下个定义了：

共识算法需要使状态复制机网络中的能够达到以下状态：

（1）一致性/安全性（Agreement/Consistency）：所有非错误的节点能够对于事件（交易）的顺序达成共识，并对外提供相同的状态输出

（2）可终止性/活性（Termination/Liveness）：所有非错误节点最终都能完成计算并输出一个值。

这样看起来，要想实现这么一个算法好像并没有那么困难

但实际上在通往共识的道路上我们需要跨过三座大山：时钟，容错和网络。

首先我们先聊一下时钟。

状态复制机中的事件是有先后顺序的，而我们需要解决的第一个问题，就是如何知道这些事件的先后顺序。

这个问题最简单的解决方法是找一个统一的全球时钟，让所有的节点都按照这个时钟来标记事件发生的时间。

然而这样的一个时钟在分布式系统中是不存在的，即使我们选中世界上某处的一个时钟作为时间源，不同节点访问该时钟所需要的时间都会不一样，这就依然会导致各节点对于事件发生的先后顺序产生不一样的观察结果。

分布式系统领域的计算科学家 Leslie Lamport 在他 1978 年的论文中提出了一种基于消息到达的先后顺序来判断事件顺序的方法。然而这种方法依然需要依赖各节点都拥有“较为一致”的物理时钟这一前提假设。

而在现实情况中这一假设并不稳定存在，由于物理时钟的细微差别，再经过一段时间的运行后，各时钟之间都难免会出现一些偏差；而且最初的时间调整工作，也是一个比较困难的社会工程学问题。

所以 Lamport 大神的这篇工作的主要贡献，还是在于为我们清晰的指出了状态复制机模型中，缺少统一的时钟是一个重大问题，也是需要共识算法来解决的问题之一。

接下来第二座山是容错

状态复制机中的每个节点都是独立的，而节点出现错误的概率也是独立的。

这些错误的原因可能是节点的网络出现问题，也可能是硬件问题，或者是什么停电了地震了之类的，都可能导致节点出错。

分布式系统领域对于错误有一个统一的定义：只要一个节点不能按照预先订好的协议进行正常的计算和消息的收发，那么它就算是出现了错误（Fault）。

而共识算法的另一个设计目的，就是容错（Fault Tolerance）：即让一个状态复制机系统能够在有节点出现错误的情况下，依然能够对状态达成共识，并对外提供统一的输出。

除了 时钟 和 容错，还有一点就是 网络

状态复制机中的各个节点需要依赖网络来互相进行通信，发送消息；它们可以选择依赖现有的如 HTTP 或 RPC 协议进行通信，也可以选择自定义的协议进行。

但无论选择哪种网络协议，现实世界中的网络并不总是牢靠的。发送出去的消息可能会出现延迟，丢失，传输可能中断。

分布式系统中对于网络条件的定义可以简单的分为同步和异步两种。

同步指的是网络中的消息总是可以在某一个常数时间 t 内被传送到；而异步指的是网络中的传输时延，并不存在一个这样的上限。（理解同步和异步的概念很重要，接下来我们会经常用到它。）

而这也是共识算法需要解决的问题，它需要网络的不稳定性考虑在内，并且保证系统依然能够达成状态共识。

稍微总结一下

共识算法是能使状态复制机网络中的各非错误节点对于交易的顺序达成共识（一致性），并总能在规定时间内对外提供输出（可终止性）的算法；并且它需要能保持系统在下列条件下依然能正常可靠的工作

1. 不存在全球统一的时钟

2. 各节点可能独立出错

3. 网络中传送的消息并不总是可靠

ok 共识算法的定义就讲完了，大家要有啥问题可以在小本本上记一下，最后一起提问

理解了什么是共识算法，也明白了共识算法需要解决的问题。我们还需要知道在共识算法领域一个非常重要的定理：FLP 不可解定理。

这个定理在 1985 年发布的论文 Impossibility of Distributed Consensus with One Faulty Process 中被首次提出，它的作者是 Fischer, Lynch, 和 Paterson 三人，所以也就被称为了 FLP 定理。

这定理其实很简单，简单到让人绝望：

“在这篇论文中我们证明了，即使只有一个错误节点的存在，任何共识算法都不可能在异步网络条件下实现可终止性。”

这是一个经过严格证明和同行审议的结论，事实上至今为止的许多尝试也都证明了这个理论的牢不可破。

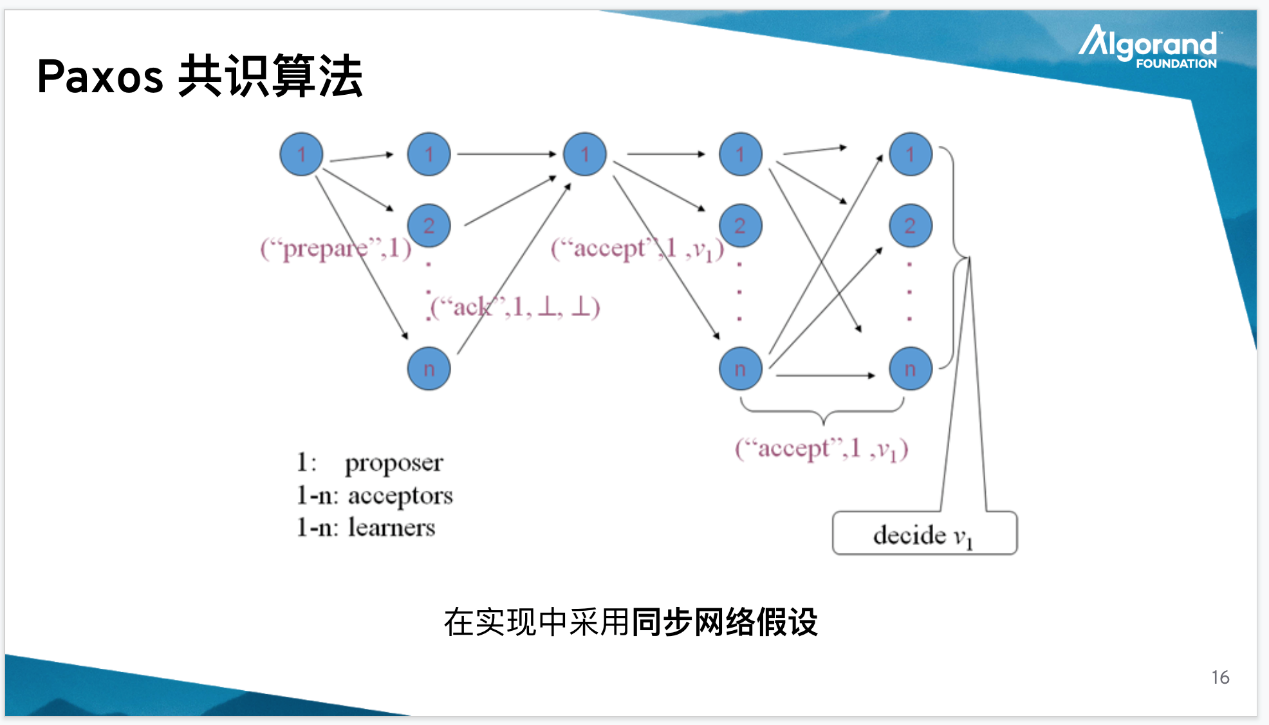
而这句话中的“任何”两字也好像已经为共识算法的大航海时代画上了一个句号，但显然坚韧的科学家们并不打算认输。

实际上，这个理论可以从另一个角度来理解：

若是想要实现一个共识算法，我们只能在“异步网络条件”和“可终止性”这两者中进行权衡，二选一。

所以如果这样理解的话，这个定理其实是为后来出现的许许多多共识算法指明了方向

而这其中最早出现的，也是最经典的一个就是 Paxos。



Paxos 共识算法的首篇论文发表于 1989年，主要贡献者还是 Leslie Lamport 大神。

Paxos 算法在之后的发展中衍生出了很多不同的改进版本，成为了一个协议族，大家比较耳熟的 Raft 也是其中之一。实际上直到今天 Google 和 Amazon 的系统里都仍然在使用这类算法。

算法的细节这里我就先不介绍了，我们着重理解一下 Paxos 是怎么绕过上述的 FLP 不可能定理的。

对于大多数系统来说可终止性是不可妥协，否则对于用户来说整个系统根本就无法实现其功能。

所以 Paxos 的选择在论文中假设网络条件为同步网络，即舍弃异步网络这一条件，从而保证可终止性。

这就意味着它会在算法的实现中规定一个超时时间，一旦超过该时间没有收到消息，则继续进行下一步骤。

实际上与 Paxos 一样，后来的所有共识算法基本上都是选择去妥协网络条件而保有可终止性。

不好意思 这里的 可终止性 怎么理解。。

不可终止指的就是，这个状态复制机一直在进行共识，但始终没有返回一个值给用户

可终止指的就是它可以在某个固定的时间内，确定能返回一个值给用户

有点像是死机的程序一样

不过需要注意的是 Paxos 共识算法是不能用于区块链中的，而其中的原因正是因为在区块链中，我们还需要考虑有名的拜占庭将军问题。

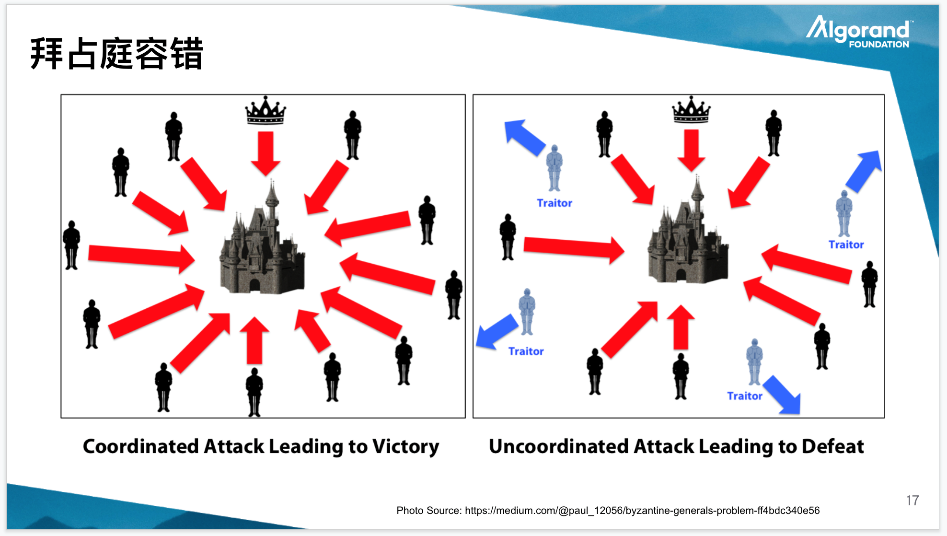
上面在介绍共识算法需要解决的问题时提到了容错，里面提到了两种节点：能正常收发消息的节点，和不能正常发消息的错误节点.

但在更加实际的环境中，尤其是在区块链中，节点可能会做出比离线更过分的事情：

它可能故意违反共识算法的规定发送错误消息。

这类节点也许会给不同的节点发送不同的消息，或者是几个节点联合起来共同发送错误的信息，尝试欺骗正常节点。

在分布式系统中这个问题被叫做拜占庭将军问题（Byzantine Generals Problem），在 1982 年被正式提出。



这份学术工作依然是由 Leslie Lamport 领衔主演。

在这篇论文中，这些会作恶的节点被称为拜占庭节点（Byzantine Node）。

而一个能够在拥有拜占庭节点的网络中依然保证网络能达成共识的共识算法，就叫做拜占庭容错共识算法，也简称叫 BFT 共识算法（Byzantine-Fault Tolerance Consensus Algorithm）。

上面提到的 Paxos 并不是 BFT 共识算法（划重点），即它只能在非拜占庭网络中正常工作；但在区块链中需要的显然必须是 BFT 共识算法。

接下来，我们就可以结束背景故事的章节，正式开始介绍一些 BFT 共识算法了。

上面也提到了，大部分的共识算法都是通过对网络条件做妥协，通过假设不同的网络条件来绕过 FLP 定理的。

这些网络假设，大致可以分成三类：

异步模型（Asynchronous Model） - 即网络中的消息延迟可以无限大

同步模型（Synchronous Model） - 即网络中的消息延迟小于某个确定的值

部分异步模型（Partial Asynchronous Model） - 即网络在“好的一天”是同步的，在“不好的一天”是异步的

有时候也有人叫它部分同步模型，不过其实是一个意思

而且论文里应该是叫部分异步

那接下来就分别介绍下这三个模型：

首先异步模型

使用异步模型的代表作是 HoneyBadgerBFT，于1983年被提出。

使用异步模型就意味着它将需要对可终止性做出妥协，即它将无法在确定的时间内终止，而是有一个概率性的终止时间，即随着时间推移它会逐渐收敛，趋向于终止。

这个算法理论上是 work 的，但理论上它的终止时间是指数型的，而且消息复杂度是 o(n^3)，即每轮共识过程需要发送节点数量的三次方次的交易。

这就意味着它将占用大量的带宽资源，而且终止时间可能会非常长。

所以异步模型本身更偏向于理论一些，有着学术贡献，但在实战中基本很难应用。

其次是 同步模型

使用同步模型的代表是 LSP， 也是由 Lamport 大神在1983年提出的。

同步模型的网络中存在着一个最大消息时延上限 t，通过充分运用这个 t，它可以在实现拜占庭容错的同时，获得共识的可终止性。

比特币中使用的中本聪共识本质上也算是同步网络模型，它通过工作量证明限制了每一次共识的时间，并且可以容忍至少一半的拜占庭节点。

同步模型也有着一些限制。由于每轮同步都需要等待固定的时延，所以响应度（Responsiveness）会比较低：即共识的速度不会随着网络状态变好或变坏而调整，而是有着固定的共识时间。

不过这方面在区块链里其实问题不大

最后是 部分异步模型

部分异步模型是最实用的网络假设模型了。大部分我们现在用的共识算法都是基于这个模型的，包括 PBFT，Tendermint，Hotstuf 等等。

部分异步模型的一个特点是它通常会有个提议者（Leader），由 Leader 来在每轮共识开始时提出（propose）一个值，然后接着由其他节点针对这个值进行共识。

使用部分异步模型的算法都会比较复杂，需要由严格的数学证明才能证明它的共识算法的特性（一致性和可终止性）。

也正是因为这个原因，市场上的大多数共识算法其实都可以一眼就看出来是假的：们要么没有严谨的数学证明，要么压根就没有解决共识算法该解决的问题，而仅仅沦落为散币的套路。

网络假设就先介绍到这

设计 BFT 共识算法时需要做的另一种假设，就是安全性假设。

所有的 BFT 共识算法都会对拜占庭节点的数量进行假设，即要求拜占庭节点的数量占总节点的数量需要低于某一个值，否则算法就不能实现其功能。

最常见的是 1/3 拜占庭容错，即算法可以容忍不多于三分之一的拜占庭节点。PBFT，Hotstuff 和 Algorand 等都采用了这一安全假设。

这个分析仅限于使用投票（Quorum-based）的共识算法，有一些特殊的算法是无法用这种方法来描述安全性的，比如比特币中的中本聪共识。

需要注意的是，这里讨论的仅仅是 BFT 共识算法自身对于作恶节点的容忍度，并不代表区块链系统的安全性。

而要实现区块链系统的安全性，我们还需要另外的机制，关于这个，我们将在最后一个章节中对此进行讨论。

接下来介绍几个经典的 BFT 共识算法

首先是 DLS

DLS 这篇论文是第一个使用部分异步网络假设的 BFT 共识算法。

我大概解释一下这个算法大致是如何工作的，让大家有个概念：

这块可能比较绕，大家先自己朗读一遍

1. 首先每一轮都得有个 Leader，每一轮开始后，各个节点首先进行一轮通信，将自己认为正确的 value 发布出去。

2. 如果 Leader 收到了 n-f 个相同的 value，它就再把这个 value 给所有的节点广播一遍。

3. 这时当各个节点收到这个 value 以后，它要将这个值“锁定”，然后再给所有的其他节点广播一遍。

4. 当 Leader 收到了来自 f+1 个节点的锁定后的 value 后，它就会将这个 value 作为最终值。

这里面的 n 是总节点数量，f 是拜占庭节点的数量

value 可以是任何一个值

一般在区块链环境里就是区块或者交易的哈希

其实从这个过程还是可以比较直观的看出它是如何满足一致性的条件的，即至少有 n-f 个节点可以对于一个最终值达成一致。

除此之外一个共识算法还需要满足可终止性，这里就需要用到网络假设了：

即部分异步网络中存在一个最大时延（在好的一天），而如果某个节点在这个时延时间内没有回复，我们就可以算它是掉线了，从而使网络整体的可终止性得到了保证。

接下来是大家都听说过的 PBFT

PBFT 这篇文章认为之前的工作都过于理论而无法在实践中应用，所以提出了一种新的更实际的基于部分异步网络假设的算法。

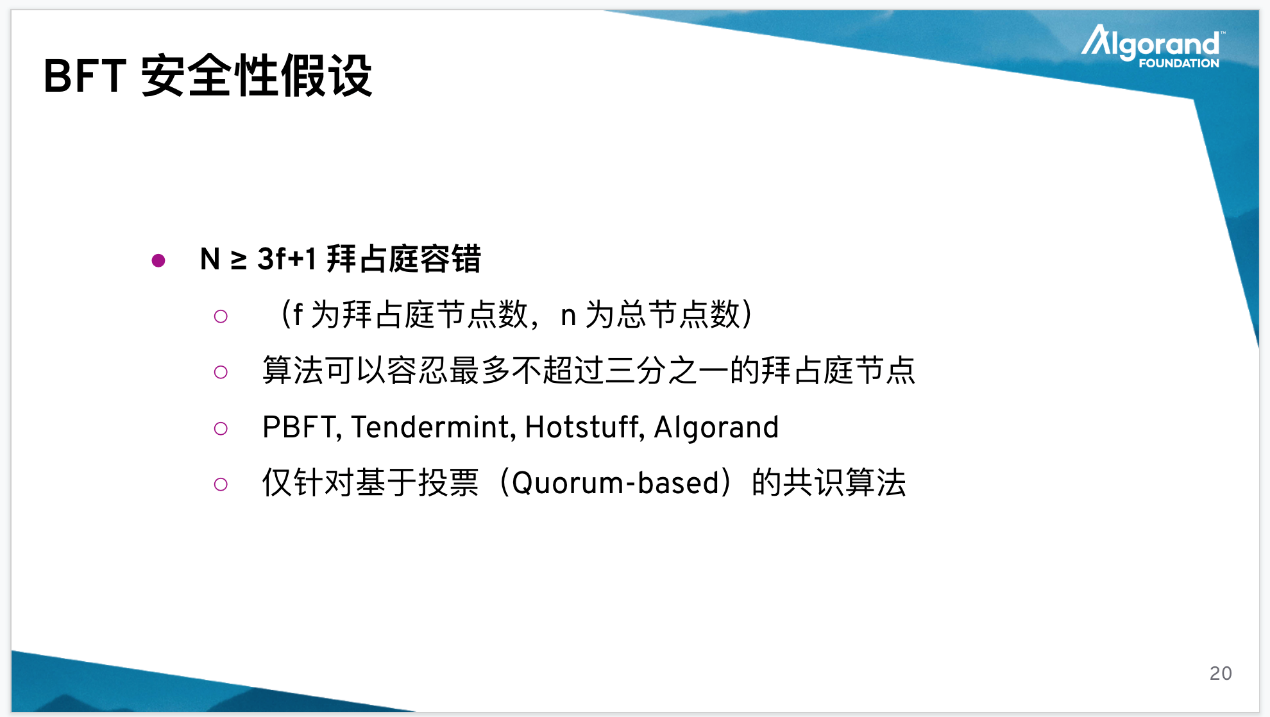
PBFT 可以在最多 (n-1)/3 的节点为错误节点（包括拜占庭节点）的条件下同时实现安全性和活性。

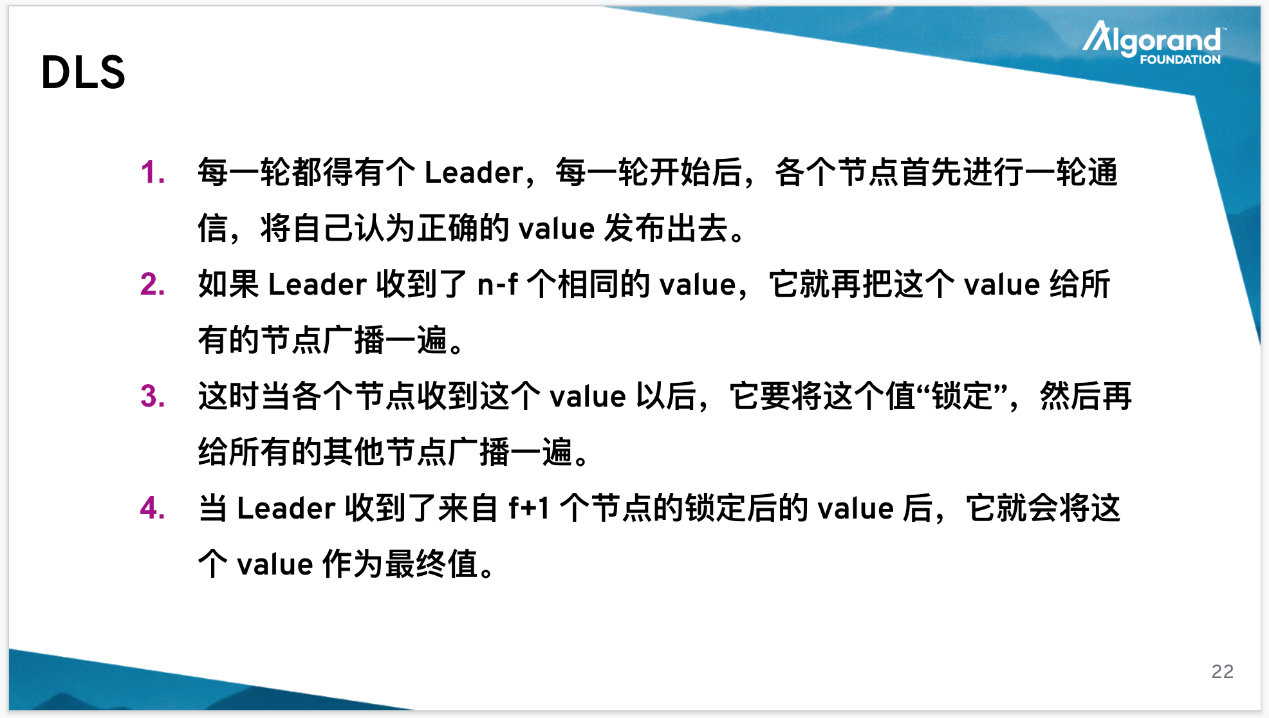
安全性就是一致性

活性就是可终止性

它大致上是这么工作的：

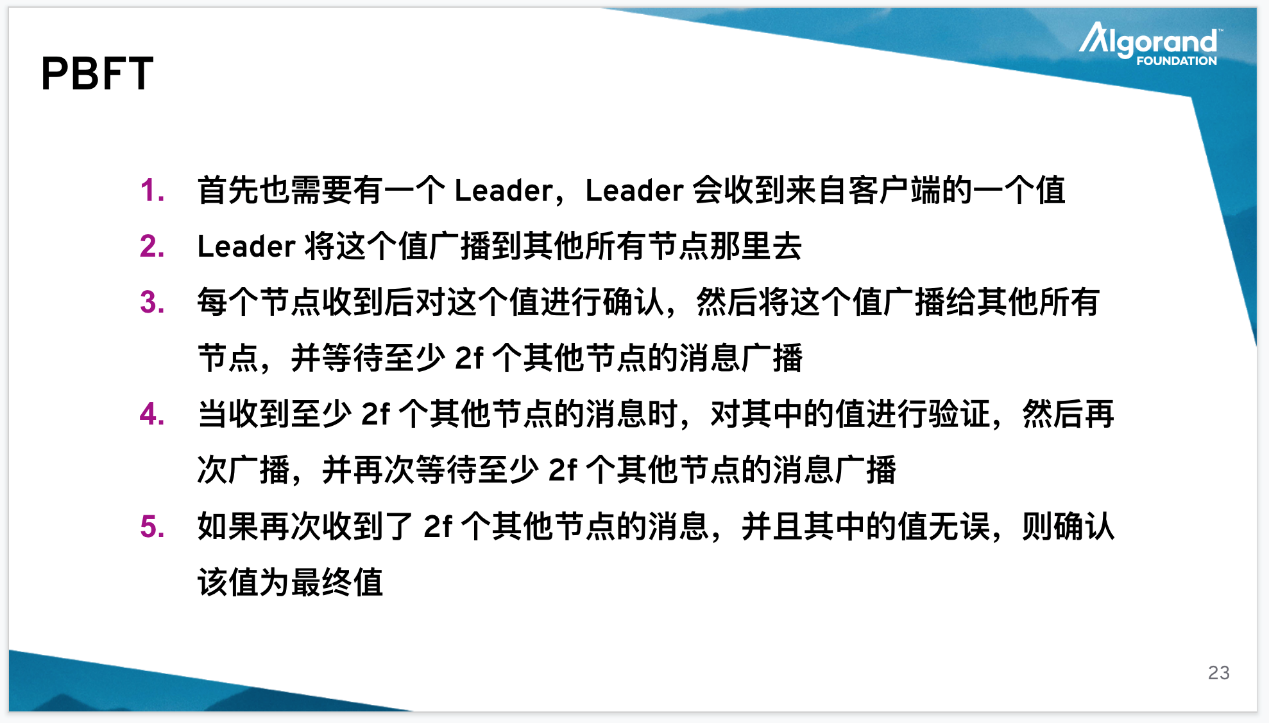
我就不发文字了，大家自己点开图片朗读一下





以上这个过程，在 Leader 没问题的情况下的消息复杂度是 O(n^2)，而当需要进行换届时则需要发送O(n^3)的消息。

同样 PBFT 这个算法也可以通过数学证明来证明共识过程结束之后，至少有 n-f 个节点可以对于一个值达成共识，即实现了共识的一致性。



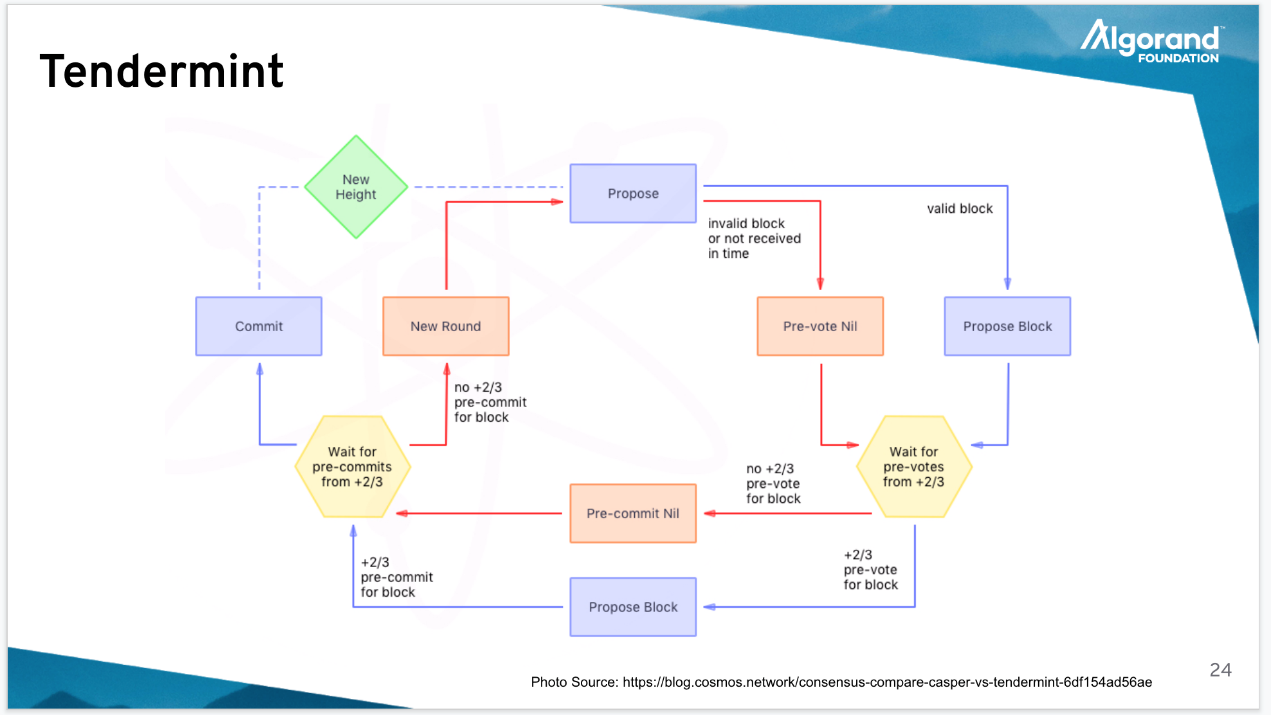
这里还可以提一下 SBFT，也许有人听过

SBFT 主要是在 PBFT 的基础上做出了一些优化：主要是通过使用门限签名（Threshold Signatures）来减少了一层消息复杂度。

然后是 Tendermint

Tendermint 是 Cosmos 提出的一个 BFT 共识算法，它的特点是它在 Leader Failure 的情况下依然是 O(n^2) 的消息复杂度。

这个图没法朗读，大家可以大概看一下



可以大概看出来它是一个循环

所以无论 leader 是好是坏，它都是用同一种办法来共识

再具体的工作流程我这里就先省去了。

再介绍一个 Hotstuff

所谓的 尤物算法

这篇论文的一作是我的好基友 Ted

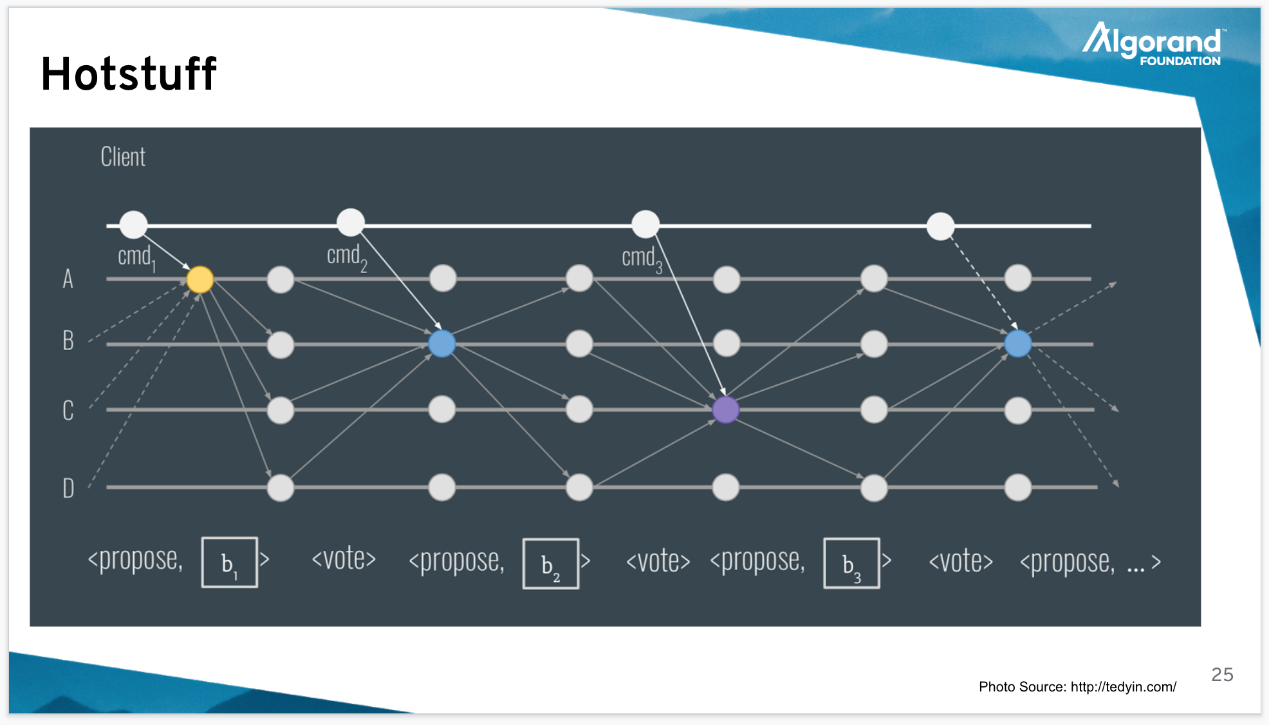
这个算法可能有人听过

Libra 里面用的就是这个共识算法

之前跟 Ted 一起去某政府部门演讲这个算法的时候，他当着各位领导的面说出尤物算法这个词的时候

我都惊了

Hotstuff 是一个最新提出的 BFT 共识算法，这份工作对之前 BFT 共识算法中的投票过程进行了抽象，从而得出了一个更加精简的共识算法。



Hotstuff 中使用了一个叫做 Quorum Certificate （QC）的概念，在某个区块拥有了足够数量的 QC 以后，该区块则可被判定为最终确定；同时算法规定了一个选择 QC 的逻辑，从而保证了共识算法的一致性；另外还设计了一个 Pacemaker 模块，保证了算法的可终止性。

这个算法拥有常数的消息复杂度；同时其精简的特性也使得其工程实现变得非常简单。

最后压轴的就是 Algorand 算法啦

Algorand 的创新之处在于它将密码学抽样的方法应用到了 BFT 共识算法中，使得在有大量节点参与的网络中也能实现快速共识。

Algorand 使用了 VRF （Verifiable Random Function）这一工具，从所有节点中随机抽样选择一部分节点参与到共识过程中，从而让真正参与到共识中的节点数大幅降低。

它的共识过程大致分为两轮：

首先各节点在本地通过运行 VRF 函数来计算自己是否可以成为该轮共识的 Leader；

之后再通过同样的方式选出一个共识委员会（Commetee），运行一个快速的 BA （Byzatine Agreement）共识算法，来对 Leader 提出的区块进行投票共识，并将最终结果广播给网络中的所有节点。

这个过程应该还是蛮直观的哈？

Algorand 共识算法的有几个非常好的特点：

首先共识速度快：即使在有大量节点（一万起）的情况下，也能拥有极快的共识速度和极高的交易处理效率

其次是不会分叉：上面所提到的共识算法中，只有中本聪共识是可以在有大量节点的环境下使用的，但中本聪共识会不可避免的产生分叉，从而影响交易确认速度（一般需要等六个区块确认）；而 Algorand 不会产生分叉，也就是说所有交易都能在一个区块内得到确认。

最好还有一点是其他共识算法一般不会说的一点

抗网络分片（Network Partition Resilience）：

在网络中的节点收到网络层面的攻击时（比如 DDoS 拒绝服务攻击），可能会导致网络分片的产生，而一般的共识算法并不会将这种攻击考虑在其安全模型中；最新版的 Algorand 算法中则将这种情况考虑在内，并且实现了即使在网络分片情况下，也能实现共识，并且快速恢复。

这些都是在实际的分布式系统中，尤其是区块链系统中非常需要的特性。

所以可以看出来，Algorand 是一个非常实用，非常适合区块链的共识算法。

那如果有朋友想在自己的项目里使用 Algorand 算法的话

请联系我

最后还有一点

是区块链的安全

到这里我们就已经学习了什么是 BFT 共识算法，了解了它的网络假设和安全性假设，并且也已经了解了几种经典的算法。

然而，区块链还需要解决一个问题：

上面的大多数 BFT 共识协议只能保证在拜占庭节点数量小于1/3的情况下正常运转

可一条区块链，尤其是无须许可的区块链（即任何人都可以加入成为节点），恶意节点的数量可能远远大于这个比例

这种情况下要如何才能保证共识协议和区块链的安全呢？

这时候就需要一些 节点准入机制 了。

一般的节点准入机制，都会使用经济激励，来激励善良的节点加入共识过程中，并劝退那些作恶的节点，从而保证区块链网络中的节点们满足 BFT 共识算法所需要的安全性假设。

下面我们就简单介绍三个最常见的准入机制：PoW，PoC 和 PoS。

要注意的是接下来关于准入机制的讨论就已经不在共识算法的范畴内了。

他们是在区块链系统中，为了保证系统安全性，而必须配合共识算法而存在的机制。

这也是为什么本质上 PoW 和 PoS 并不是共识算法。

工作量证明机制要求节点通过进行集中的运算从而计算出一个满足特定要求的数值（称为哈希值），并且可以大致推算出节点为了计算这个数值而花费了多少计算资源。

该机制最早被应用在防止垃圾邮件这个应用上，不过它最有名的应该是在比特币中的应用了，下面我们就用比特币来举例解释 PoW 如何保证区块链的安全性。

比特币协议要求，一个节点如果想要作为节点参与到共识中去，那就必须要提交一个合格的 PoW 证明。

比特币协议保证了每个 PoW 证明的计算大致需要花费10分钟，从而满足了中本聪共识需要的同步网络假设。

这样，当一个节点想要参与到共识中时，他就必须要花费足够的计算资源才可以，在比特币中这些资源也被称为算力。

对于一个敌手（Adversary）来说，如果他想要成功攻击比特币网络，成功的在中本聪共识的限制下强行让大家对他发送的恶意信息进行共识，他就需要持续花费大量资源，去控制大量的算力，这就是俗称的51%攻击。

安全性实质上是 攻击成本 和 攻击收益 的比值。

在比特币这个例子中，根据挖矿市场算力的价格和比特币的市值，我们可以分别大致估算出51%攻击所需要的成本和成功后能获得的收益。

将两者进行对比我们就能发现，过大的攻击成本和不成比例的攻击收益，足以构成劝退攻击者的城墙。

也正是因为如此，PoW 确实能保证比特币系统的安全。

PoC 空间证明是最近一段时间算是比较网红的节点准入机制。

它的大致想法是使用硬盘的空间来替代 PoW 中的计算资源。节点可以提供一个证据来证明自己拥有多大的存储空间，而拥有较大空间的节点便可以拥有更多的参与共识的权利。

PoC 其实本质与 PoW 类似，只不过是将持续计算的成本换成了购买硬盘空间所需要的一次性成本。

这种机制的好处是硬件设备不需要那么强的散热设备了，所以发出的噪音会小很多；同时主要成本从电价变成了单位存储空间的价格，让没有电力资源的人也可以参与其中，更加去中心化一些。

不过这并不是说使用 PoC 的项目都拥有与 PoW 相同的安全性，我们还需要看其共识算法是否可靠等等。

PoS 权益证明是另一种的经典节点准入机制。

它算是将上面两种进行了进一步的抽象，直接通过节点所拥有或抵押的加密货币的数量来计算节点能加入共识权重。

比较有名的案例有 Delegated PoS 和 Bonding PoS

前者让所有持币者将币抵押给某个节点，从而使其成为21个共识节点中的一员

后者让所有节点各自为自己抵押代币，并按照抵押的代币的数量来计算加入共识的权重

Algorand 也使用了 PoS 作为节点准入机制。

其做法非常简单直观，直接根据各节点所拥有的代币数量作为其被 VRF 算法选中的权重：即账户上拥有越多的币，也就越容易被选中参与到共识中。

Algorand 的创始人 Prof. Silivo Micali 将这种机制称为 Pure PoS，因为它不需要任何人去抵押代币，所有的代币都是随时流通着的。

作为理性人的用户们总是对于流动性有着偏好，所以对于 Bonding PoS 这种机制来说，并不会有很多人拿出币来进行抵押，这也就降低了区块链网络的攻击成本，对区块链和其上的资产的安全性构成了威胁。

而 Pure PoS 这种机制，并不要求用户抵押代币，网络中流通的所有代币基本上都将构成攻击成本，这就大大提升了攻击的门槛，从而提升了 Algorand 平台和其上资产的安全性。

ok 了，最后总结一下：

今天从最基础的分布式系统这一概念出发，介绍了共识算法的定义和它需要解决的问题，并延伸到了 BFT 共识算法这一领域；同时大致讲解了几种最经典的 BFT 共识算法；最后对区块链的安全性这一话题进行了讨论。

标准的议论文结尾

在此之后我们会对非常黑科技的共识算法 Algorand 进行深入的讲解，来帮助大家进一步理解共识算法这一课题。

最后再发两个参考链接：

How Does Distributed Consensus Work?

https://medium.com/s/story/lets-take-a-crack-at-understanding-distributed-consensus-dad23d0dc95

What is the difference between PBFT, Tendermint, SBFT and HotStuff?：https://decentralizedthoughts.github.io/2019-06-23-what-is-the-difference-between/

参考阅读，有时间可以看看

结束，可以开始提问啦