# 共识算法概述

共识算法：

就是解决对某一提案（目标，投票等各种协作工作），大家达成一致意见的**方式**。

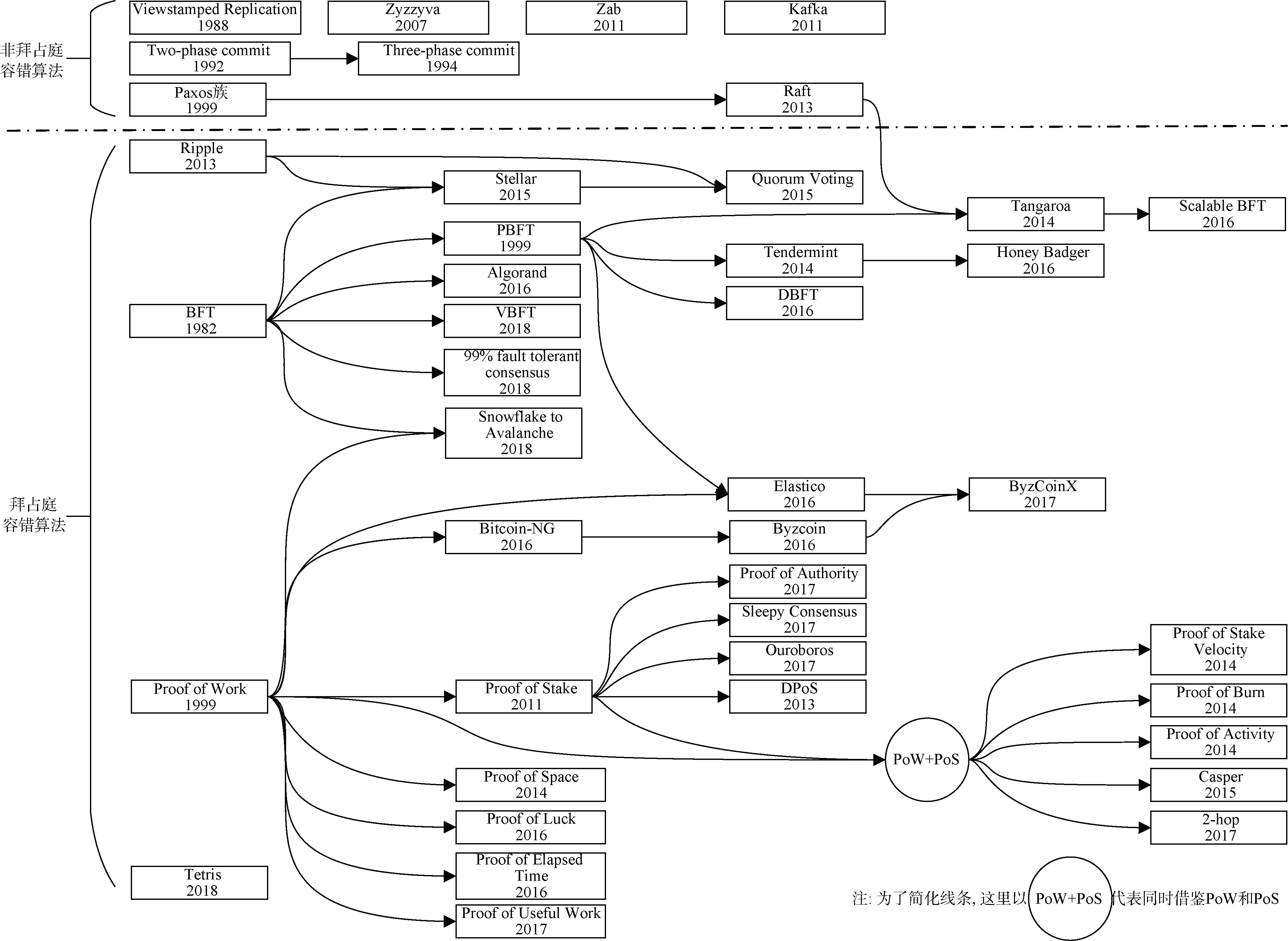
实际应用过程中的共识算法，就是**筛选**出具有**代表性**的节点（完全的“无中心”很难做到），进而达成数据的一致性。

实际问题：

在分布式系统中，特别是在区块链网络环境中，也和**拜占庭将军问题**的环境类似，有运行正常的服务器（类似忠诚的拜占庭将军），有故障的服务器，还有破坏者的服务器（类似叛变的拜占庭将军）。共识算法的核心是在正常的节点间形成对网络状态的共识。

随着节点规模的不断扩大，想要维持共识的“成本”也在不断的增加。同时，还需要在共识算法设计之初同时考虑到后续节点扩充的问题，保证算法能够持续正常的运作下去。

### 共识算法的发展



* 1982年，由Leslie Lamport、Robert Shostak、Marshall Pease 正式提出“拜占庭将军问题”，并给出了基于**口头消息**和**签名消息**的解决方案

由于拜占庭容错类算法(Byzantine Fault Tolerance,BFT)的高复杂度, BFT 类算法一直未得到实际应用,直到1999 年, Barbara Liskov 等提出了**实用拜占庭容错算法**(Practical Byzantine Fault Tolerance,PBFT)[14], 将原始拜占庭容错算法的复杂度由**指数级**降低到**多项式级**, 将拜占庭容错类算法真正引入工程领域

* 1988 年提出的Viewstamped Replication(VR)一致性算法和1989 年提出的Paxos 算法开创了非拜占庭容错类算法的先河。之后出现了多种Paxos变种算法
* 2008 年, “中本聪”发表的比特币论文开启了区块链共识算法研究的先河。比特币系统的**工作量证明共识算法**(Proof of Work, PoW)以基于工作量证明的方式开创性地解决了互联网规模的分布式系统拜占庭容错问题
* 2011 年权益证明机制(Proof of Stake, PoS)被提出, 一定程度上解决了PoW 电力资源消耗的问题
* 2013 年提出Ripple共识算法, 通过使用集体信任的子网络 (Collectively-Trusted Subnetworks), 实现了低延迟、高鲁棒性的拜占庭容错共识算法, 成功应用于基于区块链技术的金融结算网络;

**授权股份证明机制**(DelegatedProof-of-Stake, DPoS)也于2013 年提出, 引入“民主集中式”的共识准则, 提升了PoS的可扩展性, 21个超级节点的引入一定程度上降低了去中心化程度

* 2014 年提出的**结合PoS**和**PBFT** 的Tendermint共识算法以及之后基于Tendermint的改进方案HoneyBadger,旨在解决原PoS面临的“无利害关系”问题(Nothing at Stake);

同年提出的Tangaroa 共识算法结合PBFT 对Raft 算法进行改进, 继承了Raft 之后受Tangaroa 启发演变

而来的Scalable BFT 算法相比Tangaroa 具有更好的性能;算法的简洁易理解特点, 同时具有拜占庭容错能力;

* 2014—2017 年还相继出现了权益流通证明(Proof of Stake Velocity, PoSV)、空间证明(Proof of Space, PoS)、燃烧证明(Proof of Burn, PoB)、活动证明(Proof of Activity, PoA)、消逝时间证明(Proof of Elapsed Time, PoET)、幸运证明(Proof of Luck, PoL)、有用工作证明(Proof of Useful Work, PoUW)等、权威证明(Proof of Authority, PoA)等, 以不同的竞争方式替代PoW中的算力竞争以及基于PoW和PoS的二跳(2-hop)算法等解决PoW中的资源消耗和安全问题;
* 2015 年以太坊中提出Casper 协议, 在PoS中引入权益抵押;
* 2016 年, 国内NEO[50]基于PBFT 和PoS提出授权拜占庭容错(Delegated Byzantine Fault Tolerance, DBFT)共识算法, 解决了PoW和PoS中的最终一致性问题;

同年, 由图灵奖得主Micali 提出的AlgoRand 共识算法, 利用密码抽签算法随机选择部分节点参与共识过程, 被认为是适用于公链的高可扩展性共识算法;

此外, 比特币网络扩展协议Bitcoin-NG, 基于Bitcoin-NG的改进方案Byzcoin、基于分片技术的共识协议Elastico、以及基于Elastico 和Byzcoin 提出的ByzCoinX 方案, 这些方案从不同的角度提升区块链系统的可扩展性

* 2018 年也出现了一些代表性的共识算法:

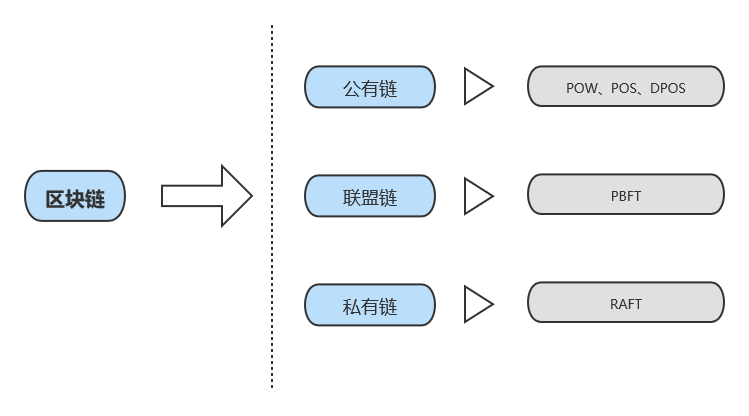
3月OCE(Ontology Consensus Engine,OCE)项目中应用的可验证拜占庭容错算法(Verifiable Byzantine Fault Tolerance, VBFT)共识算法, 结合了PoS、BFT 和可验证的随机函数(Verifiable Random Function, VRF);

5月, 康奈尔大学教授EMin Gun Sirer 发布了基于 PoW 和BFT的共识协议族Snowflake to Avalanche 共识算法[60],简单、高效且安全;

8月, 以太坊的创始人 Vitalik Buterin 发表“99% Fault Tolerant Consensus”共识算法, 仅需要1%的诚实节点就能保障区块链系统的正常运行;

11月, YEE 项目发布的基于知识推理的Tetris 共识算法, 是一种具备最终一致性的高效共识算法。

### 基本共识算法及应用场景介绍



* **公有链**：**大量**、**分散**的节点；
* **联盟链**：节点数**有限**，节点间信任等级加强；
* **私有链**：节点间默认**完全可信**，不存在“作恶节点”，只会出现节点宕机的情况；

#### POW

POW（Proof of Work），即工作量证明，简单理解就是一份证明，通过进行一定量的工作，证明节点具有一定的权益。在比特币的体系中，主要是指产生区块的权利

###### 工作量证明基本原理：

工作量证明系统主要特征是客户端需要做一定难度的工作得出一个结果，验证方却很容易通过结果来检查出客户端是不是做了相应的工作。

举个例子：给定的一个基本的字符串"Hello, World"，我们给出的工作量要求是，可以在这个字符串后面添加一个nonce值，对变更后的字符串进行SHA256运算，如果得到的哈希结果是以"0000"开头的，则验证通过。为了达到这个工作量证明的目标。我们需要不停的递增nonce值，对得到的新字符串进行SHA256哈希运算。按照这个规则，我们需要经过4251次计算才能找到恰好前4位为0的哈希散列。

由于哈希值的**伪随机特性**，要寻找4个前导0的哈希散列，预期大概要进行2^16（65536）次的尝试

|  |
| --- |
| "Hello,World1"=> e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7...  "Hello,World2"=>ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd42664...  "Hello,World4248"=>6e110d98b388e77e9c6f042ac6b497cec46660deef75a55ebc7cfdf65cc2...  "Hello,World4249"=>c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26e1...  "Hello,World4250"=>0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd... |

###### POW在比特币的应用

比特币网络中任何一个节点，如果想生成一个新的区块并写入区块链，必须解出比特币网提出的POW问题。这道题关键的3个要素是**工作量证明函数**、**区块**及**难度值**。工作量证明函数是这道题的计算方法，区块决定了这道题的输入数据，难度值决定了这道题所需要的计算量

* **区块**



* 比特币的区块由**区块头**及该区块所包含的**交易列表**组成，区块头的大小为**80字节**。
* 区块包含的交易列表则附加在区块头后面，其中的第一笔交易是Coinbase交易，这是一笔为了让矿工获得奖励及手续费的特殊交易。
* 拥有80字节固定长度的区块头，就是用于比特币工作量证明的输入字符串。因此，为了使区块头能体现区块所包含的所有交易，在区块的构造过程中，需要将该区块要包含的交易列表，通过Merkle Tree算法生成Merkle根哈希值，并以此作为交易列表的摘要存到区块头中。
* **工作量证明函数**

比特币系统中使用的工作量证明函数是SHA-256。

特点：**正向快捷**、**逆向困难**、**抗碰撞**

* **难度值**

难度值是比特币系统中的节点在生成区块时的重要参考指标，它决定了节点大约需要经过多少次哈希运算才能产生一个合法的区块。比特币的区块大约**每10分钟**生成一个，如果要在不同的全网算力条件下，新区块的产生都保持这个速率，难度值必须根据全网算力的变化进行调整。简单地说，难度值被设定在无论节点计算能力如何，新区块产生速率都保持在10分钟一个。

难度的调整是在每个完整节点中独立自动发生的。每2016个区块，所有节点都会按统一的公式自动调整难度，这个公式是由最新2016个区块的花费时长与期望时长（期望时长为20160分钟即两周，是按每10分钟一个区块的产生速率计算出的总时长）比较得出的，根据实际时长与期望时长的比值，进行相应调整（或变难或变易）。也就是说，如果区块产生的速率比10分钟快则增加难度，比10分钟慢则降低难度。

|  |
| --- |
| **公式**：新难度值 = 旧难度值 \* ( 20160 分钟/过去2016个区块花费时长 ) |

比特币工作量证明的目标值（Target）的计算公式：目标值=最大目标值/难度值。

其中最大目标值为一个恒定值：0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF。

比特币工作量证明的达成就是矿工计算出来的区块哈希值必须小于目标值

###### 工作量证明过程

1. 生成Coinbase交易，并与其他所有准备打包进区块的交易组成交易列表，通过Merkle Tree算法生成Merkle根哈希值
2. 把Merkle根哈希值及其他相关字段组装成区块头，将区块头的80字节数据作为工作量证明的输入
3. 不停的变更区块头中的随机数(nonce)的数值，并对每次变更后的的区块头做双重SHA256运算（即：**SHA256(SHA256(Block\_Header))**)，将结果值与当前网络的目标值做对比，如果小于目标值，则解题成功，工作量证明完成。

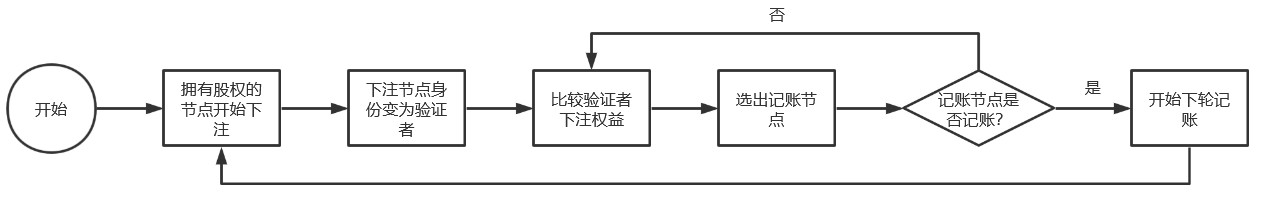
###### POW共识算法困境



不适用于交易**吞吐量大**、**交易时延要求低**的场景

#### POS

POS（Proof Of Stake），即股权证明，简单来说，就是证明你拥有某些权益。



###### POS在PeerCoin中的应用

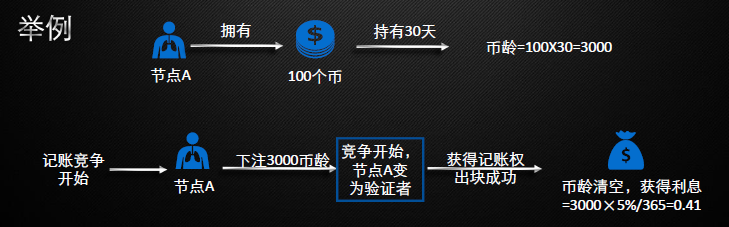
1.Peercoin采用了POS共识，它在SHA-256的哈希运算的难度方面引入了**币龄**的概念。使得**难度**与**交易输入的币龄**成**反比**。

2.在Peercoin中，币龄被定义为：**币的数量**\***币所拥有的天数**，这使得币龄能够反映交易时刻用户所拥有的货币数量。

3.未使用至少30天的币可以参与竞争下一记账权，币龄越大越有可能获得记账权。

4.获得记账权，出块成功后，币龄将清为零，这样必须等待至少30日才能签署另一区块。

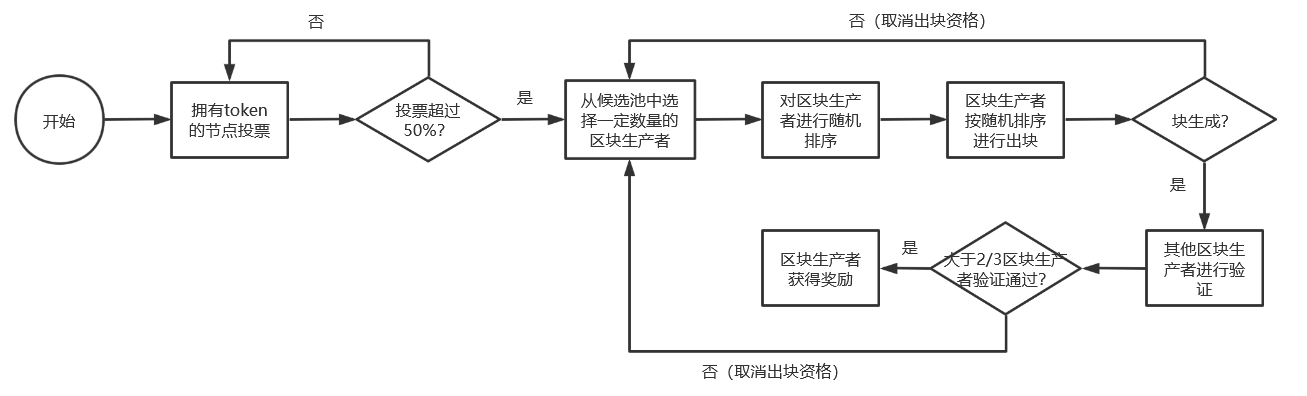
5.每被清空365币龄，你将会从区块中获得0.05个币的利息



#### DPOS

DPOS（Delegated Proof of Stake），即委托股权证明，类似投票选举，由被选举节点记账。

###### DPOS工作原理



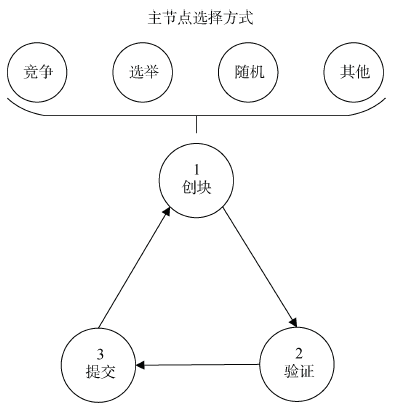
###### DPOS的应用（EOS）

EOS的DPOS分为两个步骤：1、选举区块生产者（Block Producter，简称BP）；2、达成共识，出块；

* **选举区块生产者**
* 任何持有Token的人都可以成为区块生产者，都拥有投票权。
* 投票前，候选BP可以给自己拉票（**线下方式**），每次投票超过50%的候选BP代表有效，然后生成一个BP候选池。
* 从BP候选池中选择排名靠前的21个节点作为BP，并对选出的21个BP随机排序。
* **达成共识，出块**
* 21个BP按照随机排序进行出块，在每轮出块共识的过程中，BP如果不出块或者出现恶意行为，将被其他节点举报并取消出块的权利，然后从候选池中再选择一个BP加入。
* 一个BP出块成功，并且经过至少（2/3 + 1）个BP确认，出块BP获取相应的奖励，轮流至下一个BP出块。
* 如果轮流了10次或者一天，将重新进行投票选举。

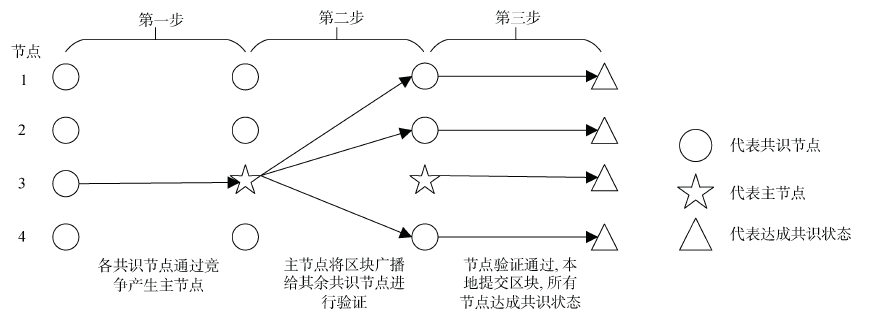
### 共识算法分类：

共识算法的核心思想之一是选择具有“代表性”的节点来维护整个系统的一致性，基于这一特点可以根据**主节点选择方式**将算法分为：**竞争类**、**选举类**、**随机类**以及其他类。算法的共识过程总体上分为三个阶段：创建区块、验证区块、提交区块，如图所示。



* **竞争类**: 所有参与共识的节点通过参与特定的竞争方式, 在竞争中获胜的节点成为主节点。竞争类共识算法以PoW为代表, 还有权益证明、空间证明等等。
* **选举类**: 所有节点通过投票的方式选择出部分节点参与共识过程, 这些被选择的节点依次轮流成为主节点。选举类共识算法以区块链中应用的传统BFT类共识算法为代表, 比如PBFT、DBFT 等。
* **随机类**:通过设计特定的随机算法从参与共识的节点中随机选择节点作为主节点。
* **其他类：**以RPCA 共识算法为例，通过引入“信任列表”，从信任列表中获取交易信息，并进行讨论判决。当获得超过一定阈值的“信任”节点的认可后，即认为达到共识。

#### 竞争类



#### PoSpace

PoSpace(即Proof of Space，空间证明)应用于Burstcoin项目中的共识算法，基于存储空间的竞争机制。在PoSpace中节点分为普通节点和校验节点两种角色：

* **普通节点**：下载**特定序列的数据块**（由节点用户公钥决定）占据硬盘空间参与主节点的竞争；
* **校验节点**：存储普通节点的**部分存储信息**以便**验证**普通节点；

###### PoSpace工作原理

* **用户接入**：

当普通初次加入网络，他需要根据选择的存储空间大小，存储一部分具有特定序列的数据（存储的数据由用户的公钥决定，因此各用户的数据并不相同）。这些数据以**有向无环图**的结构存储，而每个数据块之间的关联关系，以**Merkle树**的形式发送给校验人。

由此一来，校验人可由**公钥**知道证明人**存储的是哪些数据**、由发送的**Merkle树**知道这些数据以怎样的**结构存储**

* **验证阶段**：

1.在每轮创建新区块的竞争中，校验节点向普通节点发起“挑战”，“挑战”为普通节点所存储数据块的某种随机组合；

2.普通节点根据“挑战”信息，生成对应组合数据的哈希摘要，由于挑战是数据的一种随机组合，而略微不同的数据都将使得hash值完全不同。因此证明人必须的确存储了“挑战”所指明的数据块，才能生成正确的hash值；

3. 校验人由于存储了完整的数据库，他也可对证明人发回的hash值进行校验，验证通过则说明普通节点确实存储了相应的数据

PoSpace通过设计一个“**质量函数**”，来实现“存储空间越大的节点，越有可能获得创建新区块的权利”



其中，hash为普通节点被验证通过的哈希值，S为数据占用存储空间的大小。在每轮竞争中，质量函数结果最小的节点获胜

###### PoSpace存在的问题

* 由于矿工在一次性付出硬盘空间后，在挖矿过程中不需有后续付出，因此参加挖矿竞赛不需要付出代价，**主链分叉几乎没有成本**。我们仍需要一种规则来确定某链是唯一链，而所有用户均只记录这个唯一链，这才是真正达成共识；

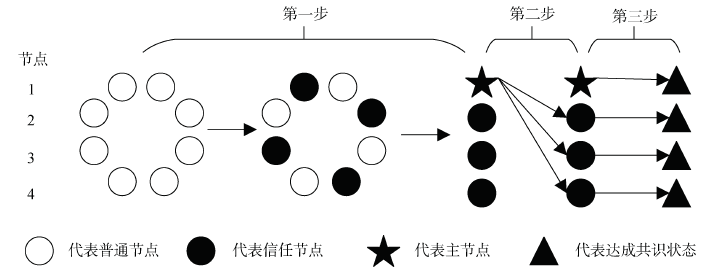
**解决思路**：由质量函数来确认唯一主链，我们设定一个数量i，规定从最新的区块往前i个区块的质量函数相加，得到链的总质量函数。总质量函数最小的那条链，可判定为主链，同时，为强调越早的区块所占的比重越高，可增加一个折扣函数，对早期的区块进行缩减（以提高其重要性）

* 引入了校验节点，增加了系统的风险；
* 如何设计和配置校验节点，需要进一步思考；
* 通过存储空间为证明，存在中心化的风险，如果少部分人可以通过巨大财力购置大量硬盘空间，持续垄断挖矿，造成类“51%攻击”

###### PoSpace的优化思路

PoSpace中硬盘所存储的空间，若能从无意义的字节，转换为有意义的内容（如媒体资源等），能够更大程度上吸引用户的参与，同时能够有效解决现有网络面临的海量数据存储的问题

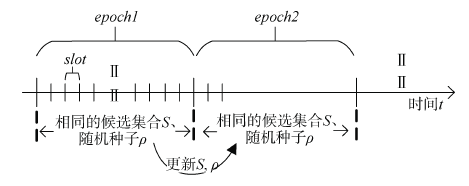
#### 选举类



#### 随机类

###### Ouroboros

Cardano 项目[75]中Ouroboros 共识算法的核心思想是：根据**节点权益大小**, **随机地**选出主节点,节点权益越大被选中的概率越大。如图所示：



Ouroboros 共识算法将时间分段划分, 一个时间段称为一个**epoch**; 以epoch为周期, 每个epoch 中划分多个时间**slot**; 每个slot 时间期限内,由**随机算法**随机选择主节点, 该随机算法在数学上证明是不可被预知。



其中, S为权益持有者候选人集合,ρ为随机种子,slj表示该 epoch 中第 j个 slot, Si∈S表示主节点。该随机算法从候选人集合S 为每个slot 选择主节点。在整个epoch 周期中, S 和ρ不变,直到周期结束之后,产生新的ρ, 更新S,开启下一个epoch 周期。

每个节点根据当前epoch 的种子, 执行随机函数F 获得当前slot 主节点; 若是自己, 则打包交易,创建区块; 否则等待主节点出块并广播, 对接收到的区块进行验证, 如果长时间未收到(超出一个slot的时间)则认为当前slot 无区块产生; 在当前epoch中不断重复这个过程直到这个epoch 中的所有slot结束。

###### PoET

基于Intel CPU 的**SGX(Software Guard Extensions)技术**, Intel 在超级账本的锯齿湖项目中提出并实现了PoET 共识算法。在这种机制下, 每个节点等待一定的随机时间, 该随机时间满足预定的概率分布函数F , **最先结束等待时间的节点成为主节点**。

其他节点进行验证, 通过两种方式验证节点确实等待了一定的随机时间：

1）产生区块的同时在SGX 的协助下产生一个等待时间的证明 , 随同区块一起广播给其他节点进行区块验证以及等待时间验证；

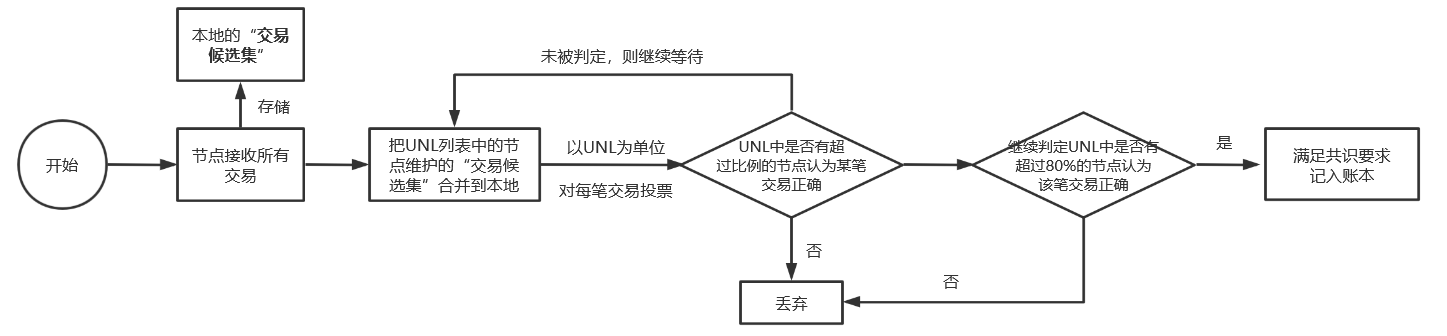
2）应用概率统计测试来检验节点的等待时间是否符合预定的概率分布F；

PoET 共识算法将**信任依托于硬件**, 使得区块链系统不必耗费大量算力, 实现了“**一CPU一票**”的公平性。每个节点成为主节点的概率相等, 但不能排除用户通过增加CPU 资源提高出块概率, 从而使中心化程度增强。PoET 面临硬件安全造成的单点故障风险, 恶意节点只需攻破一台SGX就可以控制出块权。

#### 其他类（Ripple）

Ripple 联盟链项目中提出并实现了**RPCA 共识算法**。RPCA 中引入了一个新的概念: **独立节点列表(Unique Node List, UNL)**。这是一份“信任”列表,这里的“信任”指的是节点相信这个列表里的节点不会联合起来作恶, 不需要相信这个列表里的每一个节点。每个节点维护自己的UNL, 各个节点的UNL相互独立, 可能相同也可能不同。

共识过程如下：



1.共识开始之前，每个节点接受所有的交易，放到本地的“交易候选集”；

2.每个节点把自己UNL列表中的节点各自维护的“交易候选集”合并到自己本地, 然后以UNL 为单位对每笔交易进行投票；

3.当UNL中超过一定数量比例的节点都认为某笔交易正确的时候, 这笔交易进入第4 步; 没有达到要求的交易要么被丢弃, 要么继续被留在“交易候选集”中等待下一轮共识过程开始后重新被投票；

4.共识过程的最后一步, 对于第3 步中通过的交易, 如果在这一步UNL 中超过80%的节点都认为该笔交易正确, 则这笔交易满足共识要求。所有满足这一要求的交易被添加进账本。

### 共识算法评价体系

Vitalik Buterin 提出区块链DSS猜想[61]: 区块链系统中不能从**去中心化(Decentralization)** 、**安全性(Security)** 和**可扩展性(Scalability)**三个方面同时做到提升。

* **去中心化**：网络的**控制权**分散到整个网络, 而不是被少数用户集中控制。既要保证用户公平地参与网络（即使得参与用户的收益比例与资源投入比例尽可能相近）, 同时防止网络的控制权被少数用户控制, 网络的控制权越分散, 即各用户创建区块的权利越平均, 网络的去中心化程度越高
* **安全性**：密码学安全性、网络安全性、数据安全性、共识算法安全性等；
* **可扩展性**：区块链（去中心化网络）系统处理高业务量的能力。其中分布式系统可扩展性主要面临以下三点挑战：

1. 分布式系统的网络延迟，节点间网络距离以及网络环境无法控制；
2. 节点间的数据一致性，要维护节点间的一致性需要大量节点的确认信息（越是强的一致性算法确认计算开销也会越大，同时系统规模越大达成一致的成本也越高）
3. 节点性能限制，比如PoW这种公式算法本身就会消耗大量的计算资源，在效率上存在一定限制

同时，在分布式系统中存在着“CAP定理”（**C**：**Consistency，一致性**；**A：Availability，可用性**；**P：Partition Tolerance，分区容错**）：

* **一致性**：在分布式系统中的所有数据备份，在经过有限长时间后，具有同样的值
* **可用性**：在集群中一部分节点故障后，集群整体任能响应客户端的读写请求
* **分区容错**：当集群因网路问题或节点故障问题，被切割成两个(或以上) 不完整的sub cluster 时，系统整体还能正常运作

注：在CAP定理中无法同时满足三个特性，最多只能对其中两个特性进行优化

基于“DSS猜想”以及“CAP定理”，可以得到各种公式算法的一个评价体系。基于不同的应用场景，可以进行灵活的选择。

