计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.191100189

区块链共识算法对比研究

陆歌皓　谢莉红　李析禹
云南大学软件学院　昆明 650500

摘　要　共识算法是区块链系统中最核心的部分,它直接影响着区块链系统的高效性、安全性和稳定性。针对不同的业务场景,研究者、开发者如何选择或设计一种合适的共识算法,是现阶段区块链应用落地的一大难题。文中从拜占庭将军问题出发,提出了共识算法在设计时应满足的条件。然后,将共识算法按照容错类型分为 CFT 类共识算法和 BFT 类共识算法,详细阐述了 9 种共识算法的基本原理,并从容错性、性能效率、去中心化程度、资源消耗和使用规模 5 个方面对它们进行比较,总结出它们的优缺点,以期帮助研究者、开发者选择或设计共识算法,推动区块链共识算法的应用与演进。

关键词:共识算法;区块链;容错;对比;优缺点
中图法分类号　TP301

Comparative Research of Blockchain Consensus Algorithm

LU Ge-hao,XIE Li-hong and LI Xi-yu
School of Software,Yunnan University,Kunming 650500,China

Abstract　The consensus algorithm is the most important part of blockchain system,which directly affects the blockchain system's efficiency,security and stability. According to different business scenarios,how researchers and developers choose or design an appropriate consensus algorithm is a big problem for the implementation of blockchain applications at the present stage. Based on the problem of Byzantine generals,this paper proposes the conditions that the consensus algorithm should meet in the design. Then,this paper divides the consensus algorithms into CFT consensus algorithm and BFT consensus algorithm according to the fault-tolerance type,describes the basic principles of the nine consensus algorithms in detail,and compares them from five aspects:fault-tolerance,performance efficiency,degree of decentralization,resource consumption and scale of use,and summarizes their advantages and disadvantages. It is expected to help researchers and developers select or design consensus algorithms and promote the application and evolution of block chain consensus algorithms.

Keywords　Consensus algorithm,Blockchain,Fault-tolerance,Compared,Advantages and disadvantages

1　引言

Facebook 公司 Libra 项目白皮书[1]的发布导致了比特币新一轮的暴涨,吸引了许多投资者的眼球,也引起了国内外众多研究者[2-4]对区块链的广泛关注。为抢占新一轮信息革命的先机,推动产业创新发展,各国政府高度重视区块链技术[5-6],纷纷出台政策与资金支持。

区块链是由分布式存储、P2P 组网、一致性验证、共识算法和密码学等多种计算机技术共同集成的一种新型技术[7],它利用块链式数据结构来验证与存储数据,利用共识算法来生成和更新数据,利用密码学的方式保证数据传输和访问的安全,利用由自动化脚本代码组成的智能合约来编程和操作数据[8],实现了不完全可信环境下的可信数据管理。其中,共识算法是区块链中最核心的部分,它直接影响着整个系统的高效性、安全性和稳定性。现阶段,区块链加速着数字经济发展,正与实体产业深度融合,研究者、开发者却难以根据业务需求选择或设计出吞吐量高、容错性高、低资源消耗的共识算法。

本文详细阐述了 9 种共识算法的工作原理,并从容错性、性能效率、去中心化程度、资源消耗和使用规模 5 个方面对它们进行了比较和评价,总结出各算法的优缺点,以期为研究者、开发者选择或创新设计共识算法时提供参考,推动区块链共识算法的演进。

2　拜占庭将军问题与共识

拜占庭将军问题是分布式系统中节点达成共识的经典问题,最早由 Lapmport 等[9]在 1982 年提出。若分布式系统中存在恶意的计算机节点,这些节点会选择性响应某些请求或篡改系统中的数据,在不可靠的信道上,系统中所有非恶意的节点如何通过消息传递的方式达成共识?

消息在不可靠的信道上试图通过消息传递的方式达成共识是不可能的。进程间的通信是无法达到完全同步的,节点也无法保证持续在线,在网络可靠且存在节点失效(即便只有一个)的最小化异步模型系统中,也不存在一个可以解决一致性问题的确定性算法[10]。这为共识算法的设计提出了要求:工程上,节点间须采用非对称加密算法对消息进行签名保证消息可靠传递,设置消息最大时延(即在一个最大时间限制

本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:云南省重大科技专项(2019ZE005)
This work was supported by the Major Science and Technology Projects of Yunnan Province(2019ZE005).
通信作者:陆歌皓(glu@ynu.edu.cn)

T_{max}内，若消息的接收方依旧没有收到 γ 节点发送的 ξ 消息，就判定 γ 节点失效且 ξ 消息超时，系统将不再等待和处理 ξ 消息）解决 FLP 不可能问题。同时存在恶意节点的环境下要达成共识，恶意（拜占庭/非拜占庭）计算机节点的比例不可超过系统中计算机节点总数量的 1/3，才能保证系统的安全性（Safety）与活性（Liveness）。

通常，我们将选择性只响应某些节点请求、对系统中数据做出伪造或篡改等破坏性行为的计算机节点称为拜占庭错误（Byzantine Fault）节点，将发生宕机、网络异常导致消息延时、丢失或重复等现象的计算机节点称为非拜占庭错误（Crash Fault）节点。

3 共识算法分类

共识算法能够使高度分散且彼此不信任的网络环境中的节点就某个事务达成数据一致且不产生分叉。节点间的共识过程包括 4 个阶段：选主、造块、验证、上链，其核心是选主和验证[11]。

根据共识算法容错类型的不同，即先考虑节点故障不响应的非拜占庭错误（Crash Fault）情况，再考虑节点是否会伪造或篡改信息进行恶意响应的拜占庭错误（Byzantine Fault）情况，我们将共识算法分为 CFT（Crash Fault Tolerance）类共识算法和 BFT（Byzantine Fault Tolerance）类共识算法。

3.1 CFT 类共识算法

CFT 类共识算法只能保证分布式系统中节点发生宕机等非拜占庭错误时的整个分布式系统的可靠性，而当系统中节点做出篡改数据等违背共识原则的行为时，算法将无法保障系统的可靠性，因此 CFT 类共识算法主要应用于企业内部等封闭式环境。目前主流的 CFT 共识算法主要有 Paxos 算法及其衍生的 Raft 共识算法，下面对它们的基本原理进行阐述。

3.1.1 Paxos

Paxos[12]是基于消息传递且具有高度容错特性的一致性算法，主要解决分布式系统中如何就某个特定值（Value）达成一致。Paxos 算法将节点分为 Proposer，Acceptor 和 Learner 3 种角色，每个角色对应节点上的一个进程，每个节点可同时拥有多个角色。

Proposer，即提案者，负责提出一个提案（Proposal），等待结案被选定，往往是客户端担任该角色；

Acceptor，即提案的受理者，负责对提案进行投票，往往是服务器担任该角色；

Learner，即同步者，被告知结案结果，并与之统一，不参与投票，可能是客户端也可能是服务器担任该角色。

Proposal，即提案，由 Proposer 提出。一个提案由一个编号及决议值（Value）形成的键值对组成，编号保证了提案的可区分性，决议值（Value）代表提案本身的内容。

Chosen，即提案被选定，当有半数以上的 Acceptor 批准了一个 Proposal 时则认为该提案被选定（Chosen）。

除此之外，算法还需要满足 Safety 和 Liveness 两方面的约束要求：

Safety，保证决议结果是正确的，没有歧义的。

1）只有被 Proposers 提出的决议值（Value）才能被 Chosen。

2）只能 Chosen 一个决议值（Value），并且进程只能获取那些真正被 Chosen 的决议值（Value）。

Liveness，保证决议在有限时间内完成。

3）当一个决议值（Value）被 Chosen 后，Learners 会获得这个被 Chosen 的决议值（Value）。

Paxos 算法的共识过程是 Proposer 提出提案（Proposal）来争取大多数 Acceptor 的支持，当某个 Proposer 提出的提案获得了超过半数的支持时，发送结案结果给所有节点进行确认。在这个过程中，如果 Proposer 出现了故障，可以通过触发超时机制来解决。如果每次新的一轮提案的 Proposer 都恰好故障，那么系统将永远无法达成一致，这样的情况发生的概率是极小的。

Paxos 算法共识过程分为 3 个阶段。准备阶段（Prepare），Proposer 向网络中半数以上的 Acceptor 发送一个带有提案编号的〈Prepare〉请求，试探是否可获得大多数 Acceptor 的支持；Acceptor 收到提案后会时刻保存收到过的提案最大编号和接受的最大提案，当 Acceptor 收到一个〈Prepare〉请求时，它会判断当前收到的提案编号与目前保存的最大提案编号的大小，若当前收到的提案编号比目前保存的最大提案编号大，则将其已接受过的编号最大的提案（如果未接受过任何提案，则为空）作为响应信息〈Promise〉反馈给 Proposer，同时更新当前保存的最大提案编号，并承诺不再接受任何编号小于当前收到的提案编号的提案。提交阶段（Accept），如果 Proposer 收到半数以上的响应信息〈Promise〉，则发出一个提案（带有刚才提案编号和决议值）的〈Accept〉请求。注意，如果 Proposer 收到的响应信息中不带有任何提案（为空），则提案中的决议值（Value）由 Proposer 自己决定；如果 Proposer 收到的响应信息中带有提案，则替换提案中决议值（Value）为响应中提案编号最大的决议值（Value）。Acceptor 收到〈Accept〉请求后，如果发现〈Accept〉请求中的提案编号不小于该 Acceptor 承诺接受的最大提案编号，则接受（Accepted）该提案，并更新接受的最大提案。同步阶段（Learn），Acceptor 接受（Accepted）一个提案后，将该提案发送给所有 Learner 进行同步。一旦网络中的多数节点接受（Accepted）了共同的决议值（Value），则形成决议，成为最终确认的提案。

3.1.2 Raft

Raft（The Raft Consensus Algorithm）是一种比 Paxos 更易于理解和实现的算法[13]，其核心思想是一组服务器从相同的初始状态开始，以相同的顺序执行一些列指令操作，最终就会达到一致的状态。因此 Raft 是使用日志方式来进行同步的，是一种管理复制日志的一致性算法。Raft 算法将服务器划分为 3 种角色：Leader，Follower，Candidate，3 种角色之间可相互转换。

Leader，即领导者，整个集群中有且仅有一个。负责接收客户端的请求、管理复制日志以及与 Follower 保持心跳（heartbeat）以维持领导者地位。

Follower，即追随者，被动响应请求 RPC（Remote Procedure Call）。负责响应来自 Leader 的日志复制请求，响应来自 Candidate 的选举请求，若收到客户端的请求则直接转发给 Leader，初始时所有服务器均为 Follower。

Candidate，即候选者，负责发起选举投票，Raft 启动后或 Leader 宕机后，一个节点从 Follower 转为 Candidate，并发起选举，当选成功后，由 Candidate 转为 Leader。

Term，即任期，Term 表现为一个不断连续递增的编号，每一轮选举是一个 Term，一个 Term 仅选举出一个 Leader。

Raft 算法共识过程分为两个阶段。Leader 选举阶段，当

Term 增加时 Follower 转换为 Candidate，为自己拉票并给其他服务器发送〈RequestVote RPC〉拉票，等待下面 3 种情况的发生：1)赢得了选举，即自己获得了超过半数以上的服务器投票，成为 Leader；2)输掉了选举，即另一台服务器获得了超过半数以上的投票，并接收到对应的心跳(heartbeat)，成为 Follower；3)不输不赢，即选举超时没有一台服务器获得超过半数以上的投票，Term 增加时重新发起选举。日志复制阶段，Leader 接受客户端的请求，更新日志，向所有 Follower 发送心跳(heartbeat)，所有 Follower 同步 Leader 的日志。

3.2 BFT 类共识算法

分布式系统中节点发生了任意类型的错误，BFT 类共识算法都能保证系统的可靠性，只要系统中发生错误(非拜占庭/拜占庭错误)的节点少于一定比例。因此，区块链系统大多使用的是 BFT 类共识算法。下面详细阐述几种 BFT 类共识算法的基本原理。

3.2.1 PoW

Pow(Pow of Work)即工作量证明，最早是用来对抗垃圾邮件[14-15]的一种计算机技术。2008 年，中本聪在比特币白皮书[16]中创新性地设计了 PoW 算法，该算法是通过解决一个计算难题来参与共识的一种算法，核心是利用计算机算力寻找一个满足区块哈希值的 Nonce 值。

PoW 算法的共识过程是客户端发起交易广播到网络中等待确认，网络中的用户将所有等待确认的交易打包到一个区块体中，交易在区块体中以 Merkle 树的形式构造成 MerkleRoot 记录到区块头，收集和记录区块头中的其他元数据，并不断修改区块头中的 Nonce 值以使该区块头的哈希值小于一个特定的目标值，当 Nonce 值被计算出来时，向全网广播，网络中收到提案区块的节点对其进行验证，验证合法，交易被确认，该区块成功被添加到主链上，重新开始计算下一个区块。同时，比特币 PoW 算法中设定了激励机制，节点可以通过算力来竞争区块的记账权，这个过程称为“挖矿”，参与挖矿的节点被称为“矿工”，第一个正确计算出 Nonce 值并成功将区块添加到链上的矿工会获得一定数目的系统奖励，奖励来自交易发起方支付的手续费和网络中预设的固定奖励，用户为获得自身最大化利益就会积极维护系统安全。除此之外，算法还设定了主链原则即网络中最长的一条链才是正确有效的链，解决了共识过程中的分叉问题。

哈希算法能够将任意长度的输入变为固定长度的输出，特点是正向快速逆向困难，即对于特定的输入，哈希结果都是一致的，结果的验证是非常容易的，反之，要得到一个特定的哈希值，只能不断替换输入来寻找目标值。因此，比特币 PoW 算法采用了 SHA256 双哈希运算，区块头中设置 80 个字节。合格区块的判定公式如下：

$$SHA256(SHA256(nVersion, hashPrevBlock, MerkleRoot, nTime, nBits, nNonce)) < Target \quad (1)$$

其中，$nVersion$ 为 4 字节，为版本号；$hashPrevBlock$ 为 32 字节，为上一个区块的哈希值；$MerkleRoot$ 为默克尔根，由当前区块内所有交易记录构造而成；$nTime$ 为 4 字节，为时间戳；$nBits$ 为 4 字节，记录本区块的难度值；$nNonce$ 为 4 字节，为随机数；$Target$ 为当前目标值，与难度成反比。

$$Target = MaxTarget / Difficulty \quad (2)$$

MaxTarget：最大目标值，是恒定的，即为创世区块中的 Bit 值 0x1d00FFFF。

Difficulty：当前区块难度，对应 4 个字节存储在区块头的 Bit 中。比特币为保证区块生成速度保持在大约 10 min 一个，设置了难度调整机制，每新生成 2016 个区块后难度值自动调整一次，公式为：

$$NewDifficulty = OldDifficulty \times (Tactual / (2016 \times 10\,\text{min})) \quad (3)$$

其中，$Old\ Difficulty$ 表示旧难度值；$Tactual$ 表示最近产生的 2016 个区块所花费的真实时间。

当难度值越大，目标值越小，寻找 Nonce 值所需算力就越大，保证了系统的安全性与数据的不可篡改性。

3.2.2 PoS

PoS(Pow of Stake)即股权证明[17]，是为解决 PoW 算法大量浪费资源问题而提出的一种替代算法，该算法与 PoW 算法的最大不同点在于区块的记账权由权益最高的节点获得，而不是算力最高的节点。

点点币(Peercoin)[18]是第一个实现 PoS 算法的应用。点点币中权益即为币龄，币龄是节点所持币的数量与其持有时间的乘积，发起交易会消耗一定数量的币龄，每消耗 365 币龄时也将获得年利率 5%即 0.05 个币的利息。例如某人在一笔交易中持有点点币 100 个，共持有 30 天，那么币龄为3000，后来发现了一个 PoS 块，币龄被清空为 0，获得利息为 0.05 * 3000/365=0.41 币。共识过程中节点通过消耗的币龄来获取记账权，节点消耗的币龄越多，获得记账权的机会越大。算法设定的主链原则为：消耗币龄最多的链为系统中正确有效的链。点点币中 PoS 算法合格区块的判定公式为：

$$ProofHash < Target \times CoinAge \quad (4)$$

其中，$ProofHash$ 对应一组数据的哈希，此处省略；$CoinAge$ 为币龄；$Target$ 为当前目标值，同 PoW 一样，与难度成反比。

不同的是：

$$Target = PrevTarget \times (1\,007 \times 10 \times 60 + 2 \times T_{gap}) / 1\,009 \times 10 \times 60 \quad (5)$$

其中，$PrevTarget$ 为上一个区块的目标值；$2 \times T_{gap}$ 为前两个区块的间隔时间。

当前两个区块时间间隔大于 10 min 时，当前目标值相比于上一个区块目标值会提高，即当前区块难度会降低；反之，当前两个区块时间间隔小于 10 min，当前目标值相比于上一个区块目标值会降低，即当前区块难度会提高。

3.2.3 DPoS

DPoS(Delegated Proof of Stake)即股权授权证明机制[19-20]，它是 PoS 的一种衍生算法，算法的思想是系统中持有权益的节点投票选举出一部分代表，再由这些代表轮流获取区块记账权，某种程度上类似于股份制公司的“董事会”。

DPoS 算法将节点分为两部分：参与投票的选民和被选举出的代表。每个节点都可将自己持有的权益转化为选票投给自己信任的节点，所持的权益越多，选票所占的权重也就越高，获得票数最多的 n 个节点当选为见证人(Witness)即代表。见证人的名单每经过一个固定周期都将重新选举更换一次。见证人直接参与区块的共识和验证过程，在一个规定的时间内它们随机排列并轮流对交易进行打包，生成新区块连接到最长链上。每生成一个区块，见证人会获得 m%的交易手续费，m 值由见证人们提议设定并由选民们投票决议。当然，参与见证人的竞选也需缴纳大量的保证金，这样，见证人

在系统中投入的资金最大，为保证自身的利益积极地维护系统安全。

3.2.4 PBFT

PBFT(Practical Byzantine Fault Tolerance)即实用拜占庭容错算法[21]，是一种基于状态机复制(State machine replication)的一致性算法，服务作为状态机，在分布式系统的不同节点中进行副本复制，每个状态机的副本都保存了服务的状态和所实现的操作。该算法可在发生错误的节点比例不超过节点总数1/3的情况下保证系统正常运行。其思想是让每一个收到消息的节点都去询问其他节点收到的消息内容。

PBFT算法将节点分为两种类型：主节点(Pri-mary)和备份节点(Backup)。主节点负责为客户端的交易请求分配序号，并转发给备份节点；备份节点负责按照主节点分配的序号执行请求，同时检查序号合法性，并使用超时机制来监测主节点是否宕机。为描述方便，我们用集合 R 来表示系统中的所有节点，每个节点用整数{0，…，$|R|-1$}编号，同时假设 $|R|=3f+1$，f 为发生错误的节点最大个数。主节点编号为 p，并通过公式 $p=v \bmod |R|$ 来选取，主节点选取完毕后，其他节点转化为备份节点。这里 v 表示视图(View)的编号，它是从0开始连续递增的一个数字，通常一轮共识都是在同一个视图中进行的，当主节点失效或处理完一个客户端请求，视图更换(View Change)协议启动，重新选取主节点，因此一个主节点的有效时间为一个视图。

PBFT算法的共识过程分为3个阶段：预准备阶段(Pre-Prepare)、准备阶段(Prepare)和确认阶段(Commit)，如图1所示。

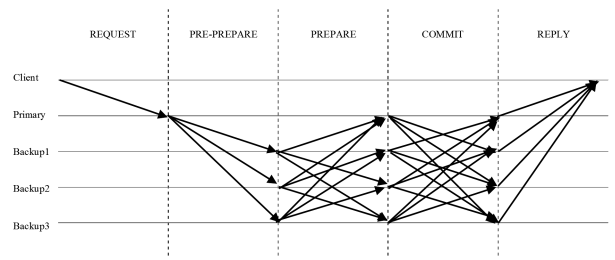

图1 PBFT算法共识过程

Fig. 1 PBFT algorithm consensus process

预准备阶段，主节点 p 收到客户端的请求 m 并给请求分配序号 n，向所有备份节点广播〈*Pre-Prepare*，v，n，$D(m)$〉消息，只要一个备份节点认可了主节点分配的序号 n 并接受该〈Pre-Prepare〉消息，那么该备份节点就会进入准备阶段。进入准备阶段的节点会向除自己以外的所有节点发送〈*Prepare*，v，n，$D(m)$，i〉消息，每个节点都将接受其他节点的Prepare消息，当收到 $2f$ 个与〈Pre-Prepare〉消息一致的Prepare消息时，请求 m 就视为已准备好(Prepared)状态。这时，备份节点会全网广播一条〈*Commit*，v，n，$D(m)$，i〉消息，当收到 $2f+1$ 个(包含自身)通过验证的〈Commit〉消息时，m 请求进入被确认(Commited)状态并被执行，执行结果将返回给客户端保存到本地状态数据库。

3.2.5 BFT-Smart

BFT-Smart是一个采用Java语言编写的状态机复制库，它同样是被设计为可容忍⌊$(n-f)/3$⌋个节点发生拜占庭错误的算法[22-23]。BFT-Smart算法中设置了状态转换(State Transfer)服务可对发生错误的节点进行修复，重新放入系统，并允许修复完成的节点访问其他节点的状态来获取最新的复制状态。同时为避免 f 个节点同时发生错误，状态转换服务将各节点已执行的操作日志存储在其他磁盘，保证系统出现这样的情况时可稳定地恢复。除此之外，通过一个可信的第三方(TTP)特殊客户端，算法可实现系统动态的添加/删除副本，即重新配置(Reconfiguration)。

BFT-Smart算法将节点分为领导节点(Leader)和副本节点(Backup)两种类型。其摄政(Regency)机制与PBFT算法的视图(View)机制相同。

BFT-Smart算法的共识过程分为提议(Propose)、写入(Write)和接受(Accept)3个阶段，如图2所示。全网节点中选举出一个领导节点(Leader)，客户端向所有节点发送〈Request〉消息，其中包含客户端序列号、数字签名以及操作请求内容等，然后等待回应。当系统处于正常阶段(Normal Phase)即系统中不存在失效或发生错误的节点时，领导节点首先对接收到的〈Request〉消息进行正确性验证，验证通过后接受它并为它分配序号，同时发送〈Propose〉消息给副本节点，只要一个副本节点接受了该消息并进行了转发，那么其他节点也将接受并向包括自己的所有节点发送〈Write〉消息。当收到 $2f$ 个〈Write〉消息时，节点再次向全网包括自己的所有节点广播一个〈Accept〉消息。当节点收到 $2f+1$ 个〈Accept〉消息时请求被执行，算法将在每个节点的日志中存储这一系列请求操作内容和加密证明等，同时将响应回复〈Accept〉给客户端。若系统中有节点发生错误(错误节点数目 $f=\lfloor(n-1)/3\rfloor$)并触发了两次超时，算法将强制跳转至同步阶段(Synchronization Phase)，同时摄政机制启动重新选举领导节点，该过程与共识过程可同步进行。第一次触发超时，〈Request〉请求会自动转发给所有节点，因为该超时的触发可能是由于错误的节点仅将其响应发送给部分节点而产生的。第二次触发超时，节点马上进入下一个摄政同时发送〈Stop〉消息通知其他节点，当一个节点收到超过 f 个〈Stop〉消息就马上启动下一个摄政。领导人选举完毕，所有节点向新的领导节点发送〈StopData〉消息，当至少有 $n-f$ 个有效的〈StopData〉消息被领导节点接受，它会发送一个〈SYNC〉消息到所有节点，接收到〈SYNC〉消息的节点会执行与领导节点相同的操作来验证领导节点是否收集并发送了有效的信息。如果领导节点是正确的，那么所有节点都变成了相同的状态。

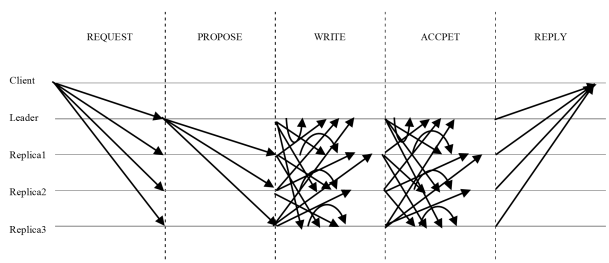

图2 BFT-Smart算法正常阶段共识过程

Fig. 2 BFT-Smart algorithm normal phase consensus process

3.2.6 DBFT

DBFT(Delegated Byzantine Fault Tolerant)即授权拜占庭容错算法[24]，它是由小蚁团队提出的一种改进的拜占庭容错算法。该算法的思想是系统中的代币持有者通过投票选举出自己所支持的共识节点，这些被选出的共识节点再通过BFT算法来达成共识并生成区块。该算法目前已经被

运用于 NEO[26]项目中。

DBFT 算法将节点分为两种类型：普通节点和共识节点。普通节点负责发起交易，并接受账本中的数据，不参与系统的共识过程；共识节点由 NEO 持有者选出，负责生成区块，并维护账本安全，同时共识节点需轮流担任议长(1 个)和议员(多个)的角色。这里，我们用 R 表示共识节点的集合，$n=|R|$ 表示参与共识过程的节点数量，那么只要共识节点发生错误的数量 f 不超过$\lfloor(n-1)/3\rfloor$，算法就能保证系统正常运行。

DBFT 算法的共识过程如图 3 所示。系统中所有节点投票选出一定数量的共识节点，任意节点发起交易并附上自己的签名，收到交易信息的共识节点会将交易记录到本地存储，被确定好的议长设置视图并广播提案〈PrepareRequest〉，议员收到提案后对其进行验证，若提案被验证无效则启动视图更换(ViewChange)，重新确定议长；若提案被验证有效，议员向全网发送〈PrepareReponse〉消息，当收到 $n-f$ 条〈PrepareReponse〉消息时，议员们达成共识并发布一个新的区块〈Block〉，对新区块签名，发送给其他节点进行同步〈Synchronization〉。其他节点收到完整区块后达到相同的区块高度，同时清除本地内存中所存储的当前视图交易数据，然后开始新一轮的共识。除此之外，若在一个视图规定的时间内共识节点未能达成共识，任意共识节点在收到 $n-f$ 条视图更换(ViewChange)消息时，都会增加视图，进入下一轮共识。

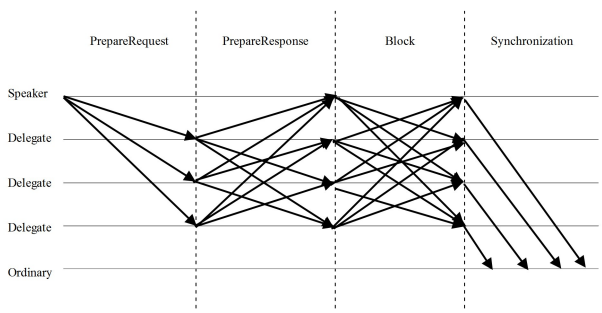

图 3 DBFT 算法共识过程

Fig. 3 DBFT algorithm consensus process

3.2.7 RPCA

RPCA(The Ripple Protocol consensus algorithm)即瑞波算法[27-28]，是 Ripple Labs 公司(原 OpenCoin)运营的瑞波币[29]系统正在使用的一种共识算法。该算法通过一些预先配置好的验证节点来完成共识，验证节点对交易进行验证和投票，经过几轮投票，若某交易持续获得超过一个阈值(一般为 80%)的选票，交易直接记录到账本中。系统中的每个节点都维护着一份自身信任的节点列表(Unique Node List，UNL)，它是验证节点的一个子集。除验证节点外，系统中还存在非验证节点，也称为追踪节点，它们主要负责转发网络中的交易信息和响应客户端的请求，不参与系统的共识过程。验证节点和追踪节点可相互切换，当追踪节点获得一定阈值的选票时可切换至验证节点，当验证节点长期不活跃则不再被节点信任，将其从 UNL 中删除，其变为追踪节点。

RPCA 算法的共识过程如图 4 所示。客户端发起交易广播至网络中，验证节点接收到交易数据，存储至本地并进行验证，验证内容不合法的交易将被舍弃，验证内容合法的交易汇集成交易候选集，其中包含之前未完成共识的交易。每个验证节点周期性地将自己的交易候选集作为交易提案发送给其他节点。验证节点收到其他节点发来的提案，检查提案的发送者是否在自己维护的 UNL 上：若不是，舍弃该提案；若是，对该提案进行存储并用自身的候选集交易与之进行比对，相同的交易可获得一个选票。在一定时间内，当交易未能达到 50%的选票时，该交易将回到交易候选集中等待下次共识重新确认；当交易成功获得 50%的选票时，进入下一轮，重新将其作为提案发送给其他节点，这时阈值也将提高。随着轮数的增加，阈值不断提高，直到交易达到 80%及以上的选票时，验证节点将其写入账本。

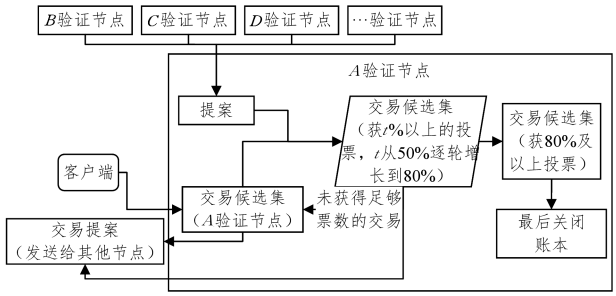

图 4 RPCA 算法共识过程

Fig. 4 RPCA algorithm consensus process

RPCA 算法中，参与共识的节点(验证节点)的身份是已知的，该算法相比于 PoW，PBFT 等算法大大降低了网络节点间的沟通成本，提高了共识效率。由于算法须达到 80%及以上的选票才能达成共识，因此想要对账本造成破坏，UNL 中的恶意节点也须达到 80%及以上才能成功，该算法很好地保证了系统的正确性。

4 9 种区块链共识算法的对比

区块链中运用不同的共识算法会导致系统的整体性能出现差异。下面，我们从去中心化程度、使用规模、容错性、性能效率和资源消耗对上述 9 种共识算法进行比较与总结。

4.1 对比指标

容错性：指共识算法对系统中发生非拜占庭错误节点的容忍值(Crash Fault Tolerance)和发生拜占庭错误的节点的容忍值(Byzantine Fault tolerance)。容错性也是算法安全性的参考之一。

性能效率：主要通过算法出块速度和每秒钟系统可处理的交易数量(TransactionPerSecond，TPS)来衡量。出块速度表现为交易打包成区块到完成共识并记录在区块链上整个过程所需要的时间。TPS 则表示交易吞吐量，由数据区块的大小和出块速度共同决定，通常等于区块内所包含的交易数量除以当前区块产生的时间长度。算法运用在实际系统中的出块速度越快，交易吞吐量越大，表示算法的性能效率越高。

去中心化程度：去中心化并不代表没有中心节点，中心节点是一个相对中立的存在。在一轮共识中，我们将记账节点看作中心节点，由于在当轮共识中，若记账节点是正确的，其他所有节点都将围绕记账节点来保持数据一致。因此，我们根据记账节点的选取规则和每轮选取的数量来比较算法去中心化程度的高低。

使用规模：指算法在系统中运用后所能承载的网络节点数量。

资源消耗：指各节点在达成共识的过程中所需消耗的计算机算力、内存、输入输出和电力等资源。

4.2 对比情况

表 1 对 9 种共识算法的容错性和性能效率进行了比较，同时收集、归纳了算法相关代表性区块链应用。表中 n 表示系统中节点总数量。

表 1 9 种共识算法容错性和性能效率比较

Table 1 Fault tolerance and performance efficiency comparison ofthe nine consensus algorithms

算法	Crash Fault Tolerance	Byzantine Fault Tolerance	出块速度（时间/块）	交易吞吐量（TPS）	代表性应用
Paxos	$n/2$	0	—	—	GoogleChubby，Zookeeper
Raft	$n/2$	0	—	—	R3 CodaQuorum，IPFS Private Cluster
PoW	$n/2$	$n/2$	10 min(bitcoin)[16]	7(Bitcoin)[16] 15(Ethereum)[31]	Bitcoin，BitcoinCash，Ethereum，Dogecoin，Litecoin，Zcash
PoS	$n/2$	$n/2$	64s(Blackcoin)[31]	—	Blackcoin，Nxt，Peercoin，Casper，ADA
DPoS	$n/2$	$n/2$	3s(Bitshares)[19] 0.5s(EOS)[34]	—	Bitshares，Steemit，EOS，Lisk，Ark，GXChain，ASCH
PBFT	$n/3$	$n/3$	秒级(Hyperledger Fabric)[40]	—	Hyperledger Fabric
DBFT	$n/3$	$n/3$	15～20 s(NEO)[39]	1000(NEO)[39]	NEO
BFT-Smart	$n/3$	$n/3$	—	8000(Symbiont)[23]	Symbiont，R3 Coda
RPCA	$n/5$	$n/5$	3 s(Ripple)	1 500(Ripple)[29]	Ripple，Stellar

本文根据各算法记账节点的选取规则和每轮选取的数量对算法的去中心化程度进行比较，可得出：Paxos（投票，1 个提案）=Raft（投票，1 个 leader）=PoW（算力竞争，1 个矿工）>PoS（权益竞争，1 个矿工）>DPoS（权益+投票，n 个记账节点）>PBFT（取模运算，1 个主节点）=BFT-Smart（取模运算，1 个主节点）>DBFT（权益+投票+取模运算，7 个共识节点）>RPCA（预先设置验证节点）。

采用 5 分到 1 分（从高到低）的方式对算法进行评分，如图 5 所示。

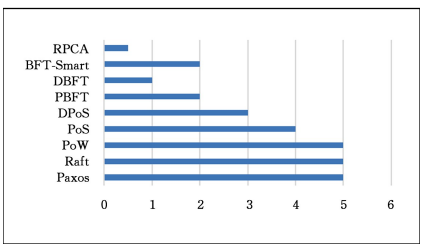

图 5 9 种共识算法去中心化程度对比图

Fig. 5 Comparison diagram of degree of decentralization of nine consensus algorithms

本文根据各算法运用于系统中导致系统所能承载的网络节点数量对算法的使用规模进行比较，可得出：PoW=PoS=DPoS>DBFT>RPCA>Paxos=Raft>PBFT=BFT-Smart。

一般地，适用于公有链的共识算法所能承载的网络节点数量较多，适用于非公有链的共识算法所能承载的网络节点数量较少。

同理，采用 5 分到 1 分（从高到低）的方式对算法进行评分，如图 6 所示。

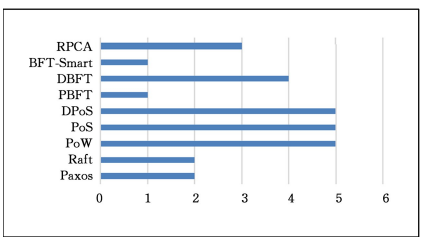

图 6 9 种共识算法使用规模对比图

Fig. 6 Comparison diagram of use scale of nine consensus algorithms

本文根据各算法在共识达成过程中对计算机算力、内存、输入输出和电力等资源的消耗大小对算法的资源消耗情况进行比较，可得出：PoW>PoS>DPoS=PBFT=DBFT>BFT-Smart=Paxos=Raft>RPCA。

PoW 与 PoS 需进行哈希计算，对计算机算力、资源消耗大。

同理，采用 5 分到 1 分（从高到低）的方式对算法进行评分，如图 7 所示。

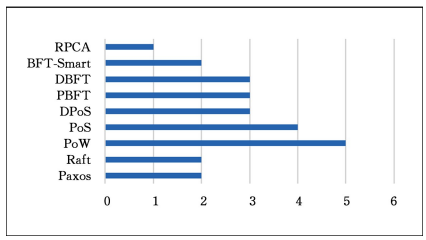

图 7 9 种共识算法资源消耗对比图

Fig. 7 Resource consumption comparison diagram of nine consensus algorithms

4.3 对比结论

Paxos 算法性能高，资源消耗低，系统可在正常节点数占比大于 1/2 的情况下达成共识，但其应用范围小，仅适用于具有较高容错性的分布式系统。谷歌 Chubby[30] 就是一个采用 Paxos 算法的典型应用，它为文件系统提供了粗粒度锁服务，用来存储大量小文件。

Raft 算法在任意一轮的共识中都有且仅有一个合法 Leader，可容忍非拜占庭错误节点数比例为 $n/2$。Raft 算法是 Paxos 算法的一种简单实现，因此其容错性、性能效率、去中心化程度、使用规模和资源消耗等方面与 Paxos 算法极相似。

PoW 算法逻辑简单，易懂易操作，容错性为 $n/2$，系统安全性有严格的数学公式证明，任何企图破坏系统的行为都将会投入大量得不偿失的成本。其缺点非常明显，挖矿消耗了大量的算力、电力和硬件等资源，数据达成一致的时间较长，系统吞吐量低，同时存在算力集中化和 51%攻击问题。比特币[16] 系统平均 10 min 产生一个区块，TPS 最高为 7 笔，以太

坊[31] TPS 最高 15 笔。

与 PoW 算法相比较，PoS 算法在一定程度上缓解了 PoW 算法资源消耗过高的问题，缩短了共识达成的时间，提高了系统性能效率。目前，运用了 PoS 算法的黑币[32]系统，出块速度可达到 64 s/个。但运用 PoS 算法的系统仍需节点进行哈希计算，在网络性能较差的情况下极易产生分叉，容易造成寡头优势，有失公平性，最重要的是其安全性未得到严格的数学论证[33]。

DPoS 算法大幅度减少了参与共识和验证的记账节点数量，降低了系统能耗，提高了出块效率，实现了秒级的共识验证。比特股(Bitshares)[19]是首个实现 DPoS 算法的应用，其选举了 101 个见证人，出块速度为 3 s/个。柚子(EOS)[34]中选举 21 个受托人，出块速度可达到 0.5 s/个。由于投票需要学习一定的技能并花费大量时间和精力，这正是投资者所缺乏的，因此造成了系统中节点投票积极性不高的问题，同时对于系统中的恶意节点，算法没有给出及时的响应措施，极易给系统造成安全隐患。现阶段已有一些研究者提出引入信用奖惩办法[35]、投票激励机制、熔断机制、信用机制[36]和备用见证人节点[37]来对 DPoS 算法进行改进。

PBFT 算法每轮记账都会向全网进行广播并由大家来共同选举主节点，可同时容忍非拜占庭错误和拜占庭错误，容忍值均为 $n/3$，具有高一致性、高可用性，抗欺诈能力强。然而随着系统中节点数量的增加，算法完成一轮共识所需要广播的消息数量呈平方级别上升，整体性能也将持续走低，因此 PBFT 算法无法承载大规模的网络节点，较适用于私有链或联盟链场景。

BFT-Smart 算法是对 PBFT 算法进行的改进，前者除执行共识外，还提供了状态转换和重新配置的服务，实现了系统中节点增删，有效提高了系统的性能效率。Symbiont[38]使用编程语言实现了 BFT-Smart 算法，在 4 节点网络集群中可达到 8000 TPS 的吞吐量，与其文献[23]中的预期性能相符合。

DBFT 算法借鉴了 DPoS 算法的思想，通过投票预先选举出部分直接参与共识的记账节点，因此我们认为此类算法是弱中心化的，有着较高的性能效率。据 NEO 白皮书[39]所述，系统每 15～20 s 可生成一个区块，交易吞吐量可达到 1000 TPS，经优化，可达到 10000 TPS。但该算法未经过大规模网络验证，系统的实际安全性未知。

RPCA 算法优点是其性能效率相对较高，瑞波币每 3 s 左右就能产生一个区块，交易吞吐量可达到 1500 TPS，而且任何时候都不会产生硬分叉。缺点是容错性较低，对非拜占庭和拜占庭错误的容忍值都仅有 $n/5$，验证节点是预先配置的，去中心化程度较低，同时验证节点的可靠性将会直接影响整个网络的正常运行。

综上，9 种共识算法的优缺点如表 2 所列。

表 2 9 种共识算法的优缺点

Table 2 Advantages and disadvantages of nine consensus algorithms

算法	优点	缺点
Paxos	算法容忍非拜占庭错误节点能力高，性能高	算法难以理解，不可容忍拜占庭错误节点
Raft	算法容忍非拜占庭错误节点能力高，强领导机制，性能高	不可容忍拜占庭错误节点
PoW	算法逻辑简单，易实现易操作，安全性高，容错性高	资源消耗过高，无意义的哈希计算工作，数据达成一致的时间较长，系统吞吐量低
PoS	一定程度上缓解了 PoW 资源浪费问题并提高了出块速度	依旧需进行无意义的哈希计算工作，币龄是时间的线性函数，易出现持币人囤币现象，造成寡头优势
DPoS	解决了 PoW 资源浪费问题，性能效率较高，出块速度较快，吞吐量较高	相比于其他算法更加趋于中心化，参与节点投票积极性不高，投票无门槛且权益越大，票数的权重越大，易造成联合选举行为，恶意节点很可能被选举为共识节点。
PBFT	无代币，性能效率高，安全性高	共识过程中，确认流程过多，通信开销大，主节点选取采用公式 $p=v \bmod \lvert R\rvert$，无法避免恶意节点担任主节点，节点不可进行动态增删，无法承受大规模节点
DBFT	借鉴了 DPoS 算法思想，参与共识节点数量较少，有效提高了性能效率	相较于其他算法更趋于中心化，系统实际安全性未知
BFT-Smart	实现节点动态增删，系统吞吐量较高	主节点选取使用公式 $p=v \bmod \lvert R\rvert$，无法避免恶意节点担任主节点，可承载节点规模未知，算法实际应用情况未知
RPCA	算法性能效率较高，安全性较高	容错性低，存在验证节点列表的正确性威胁

结束语 区块链共识算法已成为当今信息科技领域的研究热点。本文对 Paxos，Raft，PoW，PoS，DPoS，PBFT，DBFT，BFT-Smart，RPCA 9 种区块链共识算法的工作原理进行了详细阐述，并从容错性、性能效率、去中心化程度、资源消耗和使用规模 5 个方面对它们进行了比较和评价，总结出各算法的优缺点，这有助于研究者、开发者清晰认识上述 9 种共识算法，帮助研究者、开发者在针对特定业务场景展开应用时选择合适的共识算法或设计出在可扩展性、安全性、去中心化程度[41]上具有三角平衡的共识算法，推动区块链共识算法的应用与发展。

参 考 文 献

[1] AMSDEN Z, ARORA R, et al. The Libra Blockchain [EB/OL]. https://www.chainnode.com/doc/3631.

[2] LI M N. Analyzing Intellectual Structure of Related Topics to Blockchain and Bitcoin: From Co-citation Clustering and Bibliographic Coupling Perspectives [J]. Acta Automatica Sinica, 2017, 43(9): 1509-1519.

[3] ZENG S, NI X. A Bibliometric Analysis of Blockchain Research [C]// 2018 IEEE Intelligent Vehicles Symposium (IV). Changshu, 2018: 102-107.

[4] DABBAGH M, SOOKHAK M, SAFA N S. The Evolution of Blockchain: A Bibliometric Study [J]. IEEE Access, 2019, 7: 19212-19221.

[5] REN M, TANG H B, SI X M, et al. Survey of Applications Based on Blockchain in Government Department [J]. Computer Science, 2018(2): 1-7.

[6] People's Daily. During the 18th collective study session of the Political Bureau of the CPC Central Committee, xi jinping stressed that blockchain should be regarded as an important breakthrough in independent innovation of core technologies and accelerate the innovation and development of blockchain technologies and industries [EB/OL]. http://paper. people. com. cn/rmrb/html/2019-10/26/nw. D110000renmrb_20191026_2-01. html.

[7] Baidu baike. Blockchain [EB/OL]. https://baike. baidu. com/item/区块链/13465666.

[8] China blockchain technology and industrial development BBS. China's white paper on blockchain technology and application development (2016) [EB/OL]. http://www. cbdforum. cn/bcweb/index/article/rsr-6. html.

[9] LAMPORT L. The Byzantine Generals Problem[J]. Acm Transactions on Programming Languages & Systems, 1982,4(3): 382-401.

[10] FISCHER M J,LYNCH N A,PATERSON M S. Impossibility of distributed consensus with one faulty process[J]. Journal of the ACM,1985,32(2):374-382.

[11] YUAN Y,NI X C,ZENG S,et al. Blockchain Consensus Algorithms:The State of the Art and Future Trends[J]. Journal of automation,2018,44(11):93-104.

[12] LAMPORT L. Paxos made simple [J]. ACM SIGACT News, 2001,32(4):51-58.

[13] ONGARO D,OUSTERHOUTJ K. In search of an understandable consensus algorithm[C]// USENIX Annual Technical Conference. 2014:305-319.

[14] CASTRO M,LISKOV B. Practical byzantine fault tolerance and proactive recovery[J]. ACM Transactions on Computer Systems,2002,20(4):398-461.

[15] BACK A. Hashcash—A Denial of Service Counter-Measure [EB/OL]. http://www. hashcash. org/papers/hashcash. pdf, 2002-8-1.

[16] NAKAMOTO S. Bitcoin:A Peer-to-Peer Electronic Cash System [EB/OL]. https://bitcoin. org/bitcoin. pdf,2008-11-8.

[17] Wikipedia. Proof-of-stake [EB/OL]. https://en. bitcoin. it/wiki/Proof_of_Stake.

[18] KING S,NADAL S. PPCoin:Peer-to-Peer Crypto-Currency with Proof-of-Stake[EB/OL]. https://www. peercoin. net/assets/paper/peercoin-paper-nl. pdf.

[19] Delegated Proof-of-Stake Consensus [EB/OL]. https://bitshares. org/technology/delegated-proof-of-stake-consensus/.

[20] DANTHEMAN. DPOS Consensus Algorithm—The Missing White Paper[EB/OL]. https://steemit. com/dpos/@ dantheman/dpos-consensus-algorithm-this-missing-white-paper.

[21] DWORK C,NAOR M. Pricing via Processing or Combatting Junk Mail[C]// International Cryptology Conference on Advances in Cryptology. 1993:139-147.

[22] jcs47. How BFT SMaRt works. [EB/OL]. https://github. com/bft-smart/library/wiki/How-BFT-SMaRt-works,2018-10-30.

[23] BESSANI A,SOUSA J,ALCHIERI E E P. State Machine Replication for the Masses with BFT-SMART[C]// 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. Atlanta,GA,2014:355-362.

[24] SOUSA J,BESSANI A. From Byzantine Consensus to BFT State Machine Replication:A Latency-Optimal Transformation [C]// 2012 Ninth European Dependable Computing Conference. Sibiu,2012:37-48.

[25] 张铮文. 一种用于区块链的拜占庭容错算法[EB/OL]. https://docs. neo. org/zh-cn/basic/consensus/whitepaper. html.

[26] NEO[EB/OL]. https://neo. org/.

[27] SCHWARTZ D,YOUNGS N,BRITTO A. The Ripple Protocol Consensus Algorithm [EB/OL]. https://ripple. com/files/ripple_consensus_whitepaper. pdf,2013.

[28] ARMKNECHT F,KARAME G O,MANDAL A,et al. Ripple: Overview and Outlook[C]// Trust and Trustworthy Computing. 2015.

[29] Ripple[EB/OL]. https://ripple. com/.

[30] BURROWS M. The Chubby lock service for loosely-coupled distributed systems [C]// ACM OSDI '06 Proceedings of the 7th symposium on Operating systems design and implementation, 2006,335-350.

[31] Ethereum White Paper. A Next-Generation Smart Contract and Decentralized Application Platform [EB/OL]. https://github. com/ethereum/wiki/wiki/White-Paper.

[32] VASINP. BlackCoin's Proof-of-Stake Protocol v2 [EB/OL]. https://blackcoin. org/blackcoin-pos-protocol-v2-whitepaper-cn. pdf,2014.

[33] ZHOU Y F. Evolution of Blockchain Core Technology—Consensus Mechanism Evolution (1)[J]. Computer Education,268(4): 155-158.

[34] EOS. IO Technical White Paper v2[EB/OL]. https://github. com/EOSIO/Documentation/blob/master/TechnicalWhitePaper. md#consensus-algorithm-bft-dpos,2018-3-16.

[35] ZHANG Y,LI X H. The research and implementation of an improved blockchain's consensus mechanism[J]. Electronic Design Engineering,2018,26(1):38-42,47.

[36] TANG S P,YAN C. Research and improvement of block chain DPoS consensus mechanism[J]. Modern Computer (Professional Edition),2019(6):11-14.

[37] HUANG J C,XU X H,WANG S C. Improved scheme of delegated proof of stake consensus mechanism[J]. Computer Application,2019(7):2162-2167.

[38] Symbiont[EB/OL]. https://symbiont. io/technology.

[39] NEOwhitepaper[EB/OL]. https://docs. neo. org/zh-cn/whitepaper. html.

[40] Hyperledger. Hyperledger Whitepaper [EB/OL]. https://www. hyperledger. org/,2017-6-21.

[41] JIAO Z Z. All sorts of possibilities around the impossible triangle of blockchain [OL]. http://www. ccw. com. cn/channel/blockchain/2019-04-03/6967. html.

LU Ge-hao, born in 1977, Ph.D, associate professor. His main research interests include blockchain and artificial intelligence.