区块链共识算法研究综述

靳世雄[1,2], 张潇丹[2], 葛敬国[1,2], 史洪彬[2], 孙 毅[3], 李 鸣[4],
林业明[1,2], 姚忠将[1,2]

[1]中国科学院大学网络空间安全学院 北京 中国 100049
[2]中国科学院信息工程研究所 北京 中国 100093
[3]中国科学院计算技术研究所 北京 中国 100190
[4]中国电子技术标准化研究院 北京 中国 100007

摘要 共识算法是区块链系统维护数据一致性的核心机制。本文深入调研并分析了具有代表性的共识算法及其演化历程; 基于共识过程提出共识算法的分类模型, 并对各类型中代表性的共识算法进行详细分析; 最后从去中心化、可扩展性、安全性、一致性、可用性、分区容忍性六个方面建立了一套共识算法的评价指标体系, 并对代表性的共识算法进行对比分析, 给出各类算法综合性的性能评价, 希望为共识算法的应用与创新提供参考。

关键词 区块链; 共识算法; 拜占庭容错; 评价体系
中图法分类号 TP393 **DOI 号** 10.19363/J.cnki.cn10-1380/tn.2021.03.06

Overview of blockchain consensus algorithm

JIN Shixiong[1,2], ZHANG Xiaodan[2], Ge Jingguo[1,2], SHI Hongbin[2], SUN Yi[3], Li Ming[4],
LIN Yeming[1,2], YAO Zhongjiang[1,2]

[1] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[3] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[4] China Electronics Standardization Institute, Beijing 100007, China

Abstract Consensus algorithm is a key component of the blockchain system for maintaining data consistence. In this paper, typical consensus algorithms emerged in the development of blockchain are systematically reviewed and discussed. Based on consensus mechanisms, a classification model of blockchain consensus algorithms is given. In each classification category, some typical consensus algorithms are systematically described. Towards the end, a measurement system for consensus algorithms which is based on decentralization, scalability, security and agreement, consistency, availability and partition tolerance is built. With this measurement system, a comparison of these algorithms is made and the comprehensive performance of these algorithms on evaluating indicators is given out. With this work, we aim to offer useful reference for innovation of novel consensus mechanisms and development of the blockchain technology.

Key words blockchain; consensus algorithm; byzantine fault tolerance; measurement system

1 引言

2008 年, “中本聪”发表《Bitcoin: A Peer-to-Peer Electronic Cash System》[1], 详细描述了去中心化的数字货币[2]交易账本的概念和技术细节; 2009 年, 比特币系统正式发布; 随后, 比特币系统的底层技术——区块链技术进入大众视野。

区块链是一种去中心化、不可篡改、可追溯的分布式数据库系统[3]。区块链系统中底层网络采用对等式网络(P2P 网络)组织各个独立的网络节点。P2P 网络是扁平式的拓扑结构, 网络中的每个节点地位对等, 不存在任何中心化的特殊节点和层级结构。因此区块链具备去中心化的特点, 系统中各节点相互独立, 具备相同的功能, 存储同样的信息, 相互监督;

通讯作者: 张潇丹, 博士, 副研究员, zhangxiaodan@iie.ac.cn。
本课题得到“2018 新技术新应用 XX 评估与标准体系研究”项目(No. Y8V0971105); “区块链信息服务 XXXX 系统(一期)”项目(No. Y8V1181105); 国家重点研发计划“网络空间安全”重大专项“面向互联网+的云服务系统安全防护技术”项目(No. 2017YFB0801801)。
收稿日期: 2019-04-16; 修改日期: 2019-06-21; 定稿日期: 2020-12-21

与传统分布式数据库不同, 各个节点独立存储完整的数据, 任何组织或个人无法完全控制所有数据, 只拥有本地数据的控制权, 任何单个节点对本地数据的修改不会对整个区块链产生影响, 因此区块链具备难以篡改性; 区块链把数据分成不同的区块, 每一个区块头都包含前一个区块的哈希摘要信息, 如图 1 所示, 前后顺连形成一条链, 因此区块链具备可溯性。总体来说, 区块链融合了经济学的博弈论[4]、计算机科学等多种技术, 比如 P2P 网络协议[5], 块链结构、共识算法、非对称加密[6-8]、激励机制等, 解决了数据可信问题。在无需借助可信第三方的情况下, 实现互不信任的多方对等可信的价值传输。

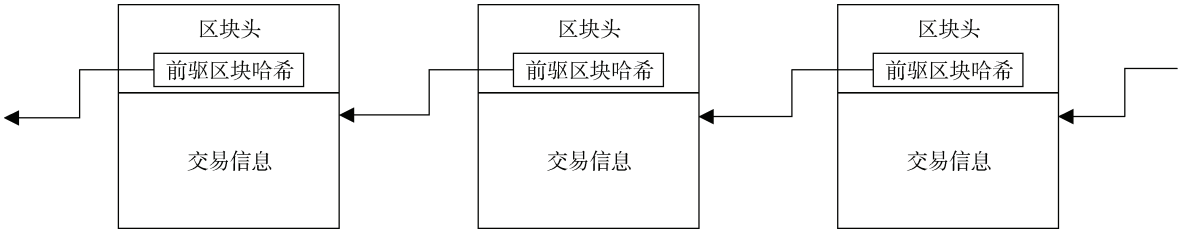


图 1 区块链示意图

Figure 1 The diagram of blockchain

共识算法是区块链系统中的核心机制, 旨在解决系统中各分布式节点数据一致性[9]的问题。在分布式对等网络中, 不同节点通过交换信息达成共识, 而网络中可能存在恶意节点篡改或伪造数据, 或者通信网络也可能导致传输信息出错, 从而影响节点间共识的达成, 破坏分布式系统的一致性。这个问题于 1982 年由 Leslie Lamport、Robert Shostak 和 Marshall Pease 正式建模为“拜占庭将军问题”[10]。传统的分布式系统共识算法大多不考虑拜占庭容错问题, 仅考虑网络延时或部分节点出现故障无法响应的情况下, 非故障节点如何实现分布式系统的数据一致性。区块链系统运行在更为开放并且缺乏信任的网络环境中, 节点数量众多且可能存在恶意节点, 因此区块链系统的共识算法设计必须考虑“拜占庭将军问题”。

1985 年, 由 Michael Fisher、Nancy Lynch 和 Michael Paterson 共同提出并证明了在分布式系统共识算法的设计中起到重要指导作用的“FLP 不可能定理”[11]。该定理指出: 在网络可靠的异步通信系统中, 当存在节点故障(即使只有一个)的情况下, 不存在协议能保证在有限时间内使系统达成一致。“FLP 不可能定理”指出了在可能存在节点失效的分布式异步通信系统中, 理论上不存在能使系统在有限时间内达成一致的共识算法。因而研究者们通过调整问题模型来规避“FLP 不可能定理”从而寻找工程上可行的共识算法, 比如比特币系统中通过假定网络为弱同步性, 即网络节点间可以快速同步, 以及矿工在一个区块上投入有限的时间等来规避“FLP 不可能定理”。

通过调整问题模型规避“FLP 不可能定理”, 使得共识算法存在“工程解”。2000 年, Eric Brewer 在一次研讨会的报告中提出了一个猜想: 分布式系统无法同时满足一致性 (Consistency)、可用性(Availability)和分区容忍性(Partition Tolerance), 最多只能同时满足其中两个特性, 如图 2 所示。该猜想于 2002 年被 Seth Gilbert 和 Nancy Lynch 在异步网络模型证明, 被命名为“CAP 定理”[12]。一致性是指分布式系统中所有节点在同一时刻持有同样的数据信息; 可用性是指系统处于服务状态, 当客户端发出请求, 服务端能在有效的时间内返回结果; 分区容忍性是指允许网络中部分节点不与其他节点通信, 即允许网络发生分区(不同区域之间的节点不能建立通信)。“CAP 定理”指出即使可以设计出工程上可行的共识算法, 这个共识算法也无法完美地做到同时满足一致性、可用性和分区容忍性。该定理的提出为共识算法的设计提供了指导性的原则, 使研究者不再追求能同时满足三个特性的共识算法。比如比特币系统的工作量证明(Proof of Work, PoW)共识算法不支持分区容忍性(要求全网节点参与竞争、验

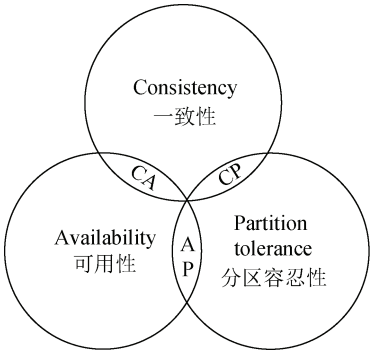


图 2 CAP 定理

Figure 2 CAP Theorem

证、最后达成共识), 同时尽可能满足一致性和可用性。比特币系统首次真正实现了互联网规模的分布式系统的拜占庭容错类共识算法, 并且该系统的稳定运行也验证了此类共识算法的可行性。

迄今为止, 研究者已经在共识领域做了大量研究工作, 从传统的分布式共识算法, 到受 PoW 共识算法启发后涌现的大量应用于区块链系统的共识算法, 这些共识算法试图从不同角度解决区块链发展中的问题。本文所做工作: 深入调研并分析区块链共识算法及演化历程; 基于区块链共识算法的共识过程, 提出共识算法的分类模型, 并对各类型中的代表性共识算法进行详细分析; 从去中心化、可扩展性、安全性、一致性、可用性、分区容忍性六个层面建立区块链共识算法的评价指标体系, 并对典型的共识算法进行对比分析, 给出各类算法综合性的性能评价。通过对共识算法进行分类以及评价指标体系的建立, 期望对未来共识算法的创新提供参考。

本文的后续章节安排如下: 第二节梳理了当前区块链共识算法的演化历程, 并展示这些共识算法的发展脉络; 第三节提出区块链共识算法的分类模型, 在此基础上, 对各类中代表性的共识算法进行详细分析; 第四节提出区块链共识算法的评价指标体系, 并对代表性的共识算法进行对比分析, 给出各类算法综合性的性能评价。最后第五节对全文进行总结。

2 共识算法的发展

分布式共识算法是分布式计算领域的核心问题, 很多研究者为促进其发展做了大量工作[13]。本节按照时间顺序给出共识算法的发展简史, 并梳理出这些共识算法的演进关系, 如图 3 所示, 箭头方向代表演进方向, 比如共识算法 *A* 指向 *B*, 则代表算法 *B* 借鉴算法 *A*; 多个算法比如 *A* 和 *B* 共同指向 *C*, 则代表算法 *C* 同时借鉴 *A* 和 *B*。

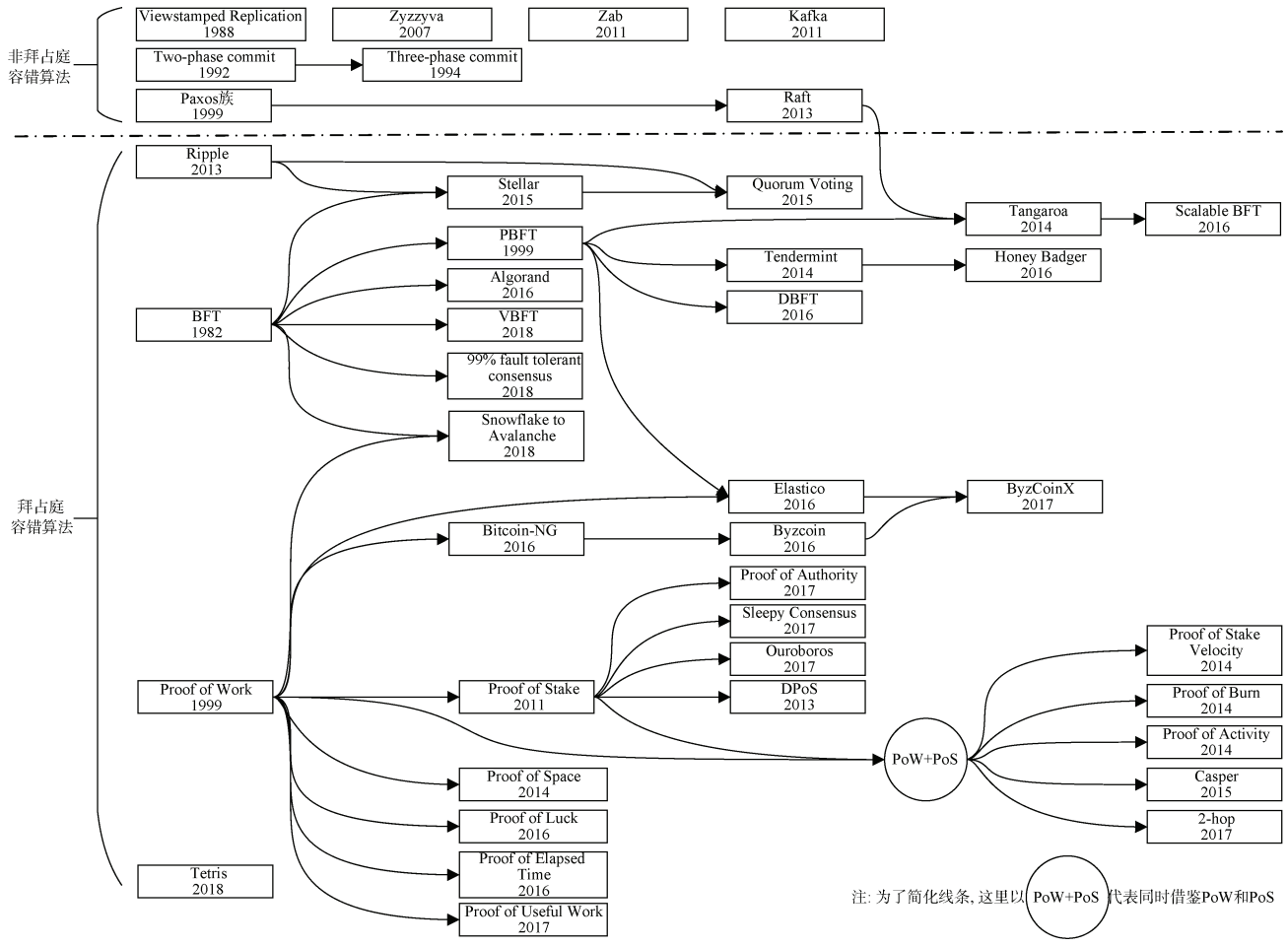


图 3 共识算法汇总

Figure 3 Summary of consensus algorithms

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, 开创了拜占庭容错类算法的先河。从此, 分布式系统共识算法被分为拜占庭容错类算法和非拜占庭容错类算法。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[15]和 1989 年提出的 Paxos 算法[16-18]开创了非拜占庭容错类算法的先河; 基于 Paxos 算法, 之后出现了多种 Paxos 变种算法[19-21]; 除此之外, 还有两阶段提交算法(Two-Phase Commit)[22]、三阶段提交算法 (Three-Phase Commit)[23]、Zyzzyva[24]、Zab[25]、Kafka[26] 以及 2013 年提出的 Paxos 算法简化版——Raft 算法[27]。以上这些算法都是非拜占庭容错类算法。

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[28-30]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。PoW 的核心思想是通过算力竞争达成共识保证数据的一致性, 同时保证了比特币系统的安全性和去中心化程度, 但同时也带来了电力资源的大量消耗; 2011 年权益证明机制(Proof of Stake, PoS)[31]被提出, 一定程度上解决了 PoW 电力资源消耗的问题; 2013 年提出瑞波共识算法[32], 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; 之后基于 Ripple 协议, 结合 BFT 算法提出的恒星共识协议[33]是第一个可证明安全的共识协议; 同年超级账本锯齿湖项目(Hyperledger Sawtooth)中提出的法定人数投票共识算法(Quorum Voting)旨在确定立即交易的状态; 授权股份证明机制(Delegated Proof-of-Stake, DPoS)[34]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, 但 21 个超级节点的引入一定程度上降低了去中心化程度; 除此之外, Sleepy Consenus[35]、Ouroboros[36]等共识算法中也应用了节点的权益; 2014 年提出的结合 PoS 和 PBFT 的 Tendermint[37]共识算法以及之后基于 Tendermint 的改进方案 HoneyBadger[38], 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[39]结合 PBFT 对 Raft 算法进行改进, 继承了 Raft 算法的简洁易理解特点, 同时具有拜占庭容错能力; 之后受 Tangaroa 启发演变而来的 Scalable BFT 算法[40]相比 Tangaroa 具有更好的性能; 2014—2017 年还相继出现了权益流通证明(Proof of Stake Velocity, PoSV)[41]、空间证明(Proof of Space, PoS)[42]、燃烧证明(Proof of Burn, PoB)[43]、活动证明(Proof of Activity, PoA)[44]、消逝时间证明(Proof of Elapsed Time, PoET)[45]、幸运证明(Proof of Luck, PoL)[46]、有用工作证明(Proof of Useful Work, PoUW)[47]等、权威证明(Proof of Authority, PoA)[48]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[49], 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[51], 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[52], 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[53], 基于 Bitcoin-NG 的改进方案 Byzcoin[54]、基于分片技术的共识协议 Elastico[55]、以及基于 Elastico 和 Byzcoin 提出的 ByzCoinX 方案[56], 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[57]中应用的可验证拜占庭容错算法(Verifiable Byzantine Fault Tolerance, VBFT)共识算法[58], 结合了 PoS、BFT 和可验证的随机函数(Verifiable Random Function, VRF)[59]; 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[61], 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[62]发布的基于知识推理的 Tetris 共识算法[62], 是一种具备最终一致性的高效共识算法。

本节按照时间顺序列出各共识算法, 这些共识算法包括传统的分布式共识算法和区块链中应用的共识算法, 并梳理出这些共识算法之间的演进关系。第三节将对在区块链中应用的共识算法建立分类模

型并描述每类中代表性的算法。

3 共识算法分类

为了对现有的共识算法有更清晰的理解, 本节对共识算法的共识过程进行建模, 并从共识过程的角度对当前主流的算法进行分类, 并详细阐述各类型中的典型共识算法。

算法的共识过程总体上分为三个阶段, 如图 4 所示: 创建区块、验证区块, 提交区块。根据共识协议, 从参与共识过程的节点中选择主节点创建新区块(定义主节点为每轮共识过程中产生有效区块的节点); 然后新区块交由其他参与共识的节点进行独立验证; 通过验证的新区块会被节点提交到本地区块链中, 达成对最新高度区块的共识; 至此完成共识, 开启下一轮共识过程。

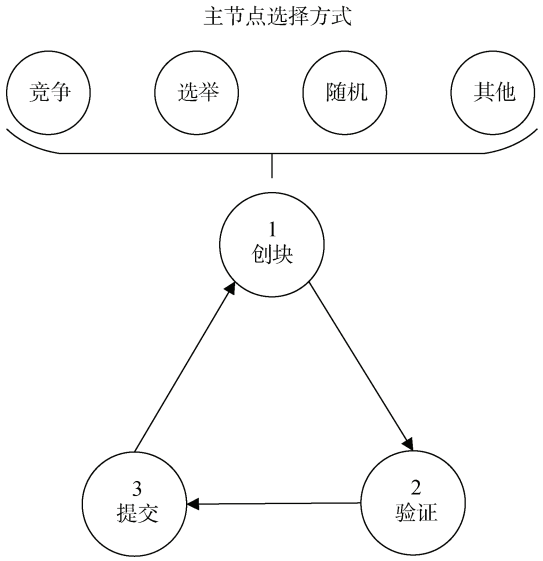


图 4 共识过程及分类模型

Figure 4 The basic consensus process and classification model

共识算法的分类方式有很多, 比如在性能层面: 从数据一致性的角度将共识算法分为强一致性共识算法、弱一致性(最终一致性)共识算法; 从拜占庭容错的角度将共识算法分为拜占庭容错共识算法、非拜占庭容错共识算法。在应用层面: 可以将共识算法分为适用于公链、联盟链、私链的共识算法。本节从共识过程出发, 按照主节点的产生方式将共识算法分为竞争类、选举类、随机类以及其他类型。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。竞争类共识算法以 PoET、Algorand、Ouroboros 为代表。以下部分, 从共识过程角度对各类中的代表算法进行详细分析。

3.1 竞争类

PoW 共识算法在比特币项目中的成功运用, 激发了一系列通过资源、能力、名誉、权益等证明方式来竞争成为主节点的共识算法。比如 Peercoin 项[63]目中的权益证明(Proof of Stake, PoS)、Parity 项目[64]中的权威证明(Proof of Authority, PoA)、Burstcoin 项目[65]中的空间证明(Proof of Sapce, PoSpace)、GoChain 项目[66]中的信誉证明(Proof of Reputation, PoR)等[67]。从命名方式的角度, 这些共识算法应该被称为证明类算法, 但从主节点的选择方式角度, 相同时间内完成证明最多的节点才成为主节点, 因此把这些共识算法归纳为竞争类更为合理。竞争类共识算法在每一轮主节点的选择中, 会设置一个竞争成功的标准, 最先达到标准的共识节点成为主节点。竞争类算法的共识过程如图 5 所示。本小节对典型的 PoW、PoS、PoSpace 进行详细分析, 帮助理解竞争类共识算法。

3.1.1 PoW

PoW 共识算法中, 所有参与共识的节点通过消耗算力解决数学难题来竞争成为主节点, 由最先解决数学难题的节点成为主节点。在比特币系统中, 中本聪采用安全散列算法, 将数学难题设计为对区块头信息的双 SHA256 哈希运算, 即:

$$F(BlockHeader \mid Nonce) < T$$

其中 F 为双 SHA256 哈希运算函数[68], T 是目标哈希值, *BlockHeader* 为区块头信息, 其中包含 *Nonce* 字段, 求解合适的 *Nonce* 字段使得对完整 *BlockHeader* 的哈希运算结果满足目标值要求。

共识过程如图 5 所示, 第一步全网节点接收并转发交易信息, 将接收到的交易信息打包进新区块, 然后全网所有节点基于本节点打包的新区块设置区块头中的 *Nonce* 字段, 计算整个区块头信息的哈希值, 使其满足目标值的要求; 率先完成计算工作的节点成为主节点, 并将自己的区块广播给其余节点; 第二步, 未竞争成功的其余节点在接收到新区块后, 放弃计算本地新区块, 并对接收到的新区块进行验证; 第三步, 节点在验证通过新区块后, 将其添加到本地区块链, 所有节点达成对最新高度区块的共识。

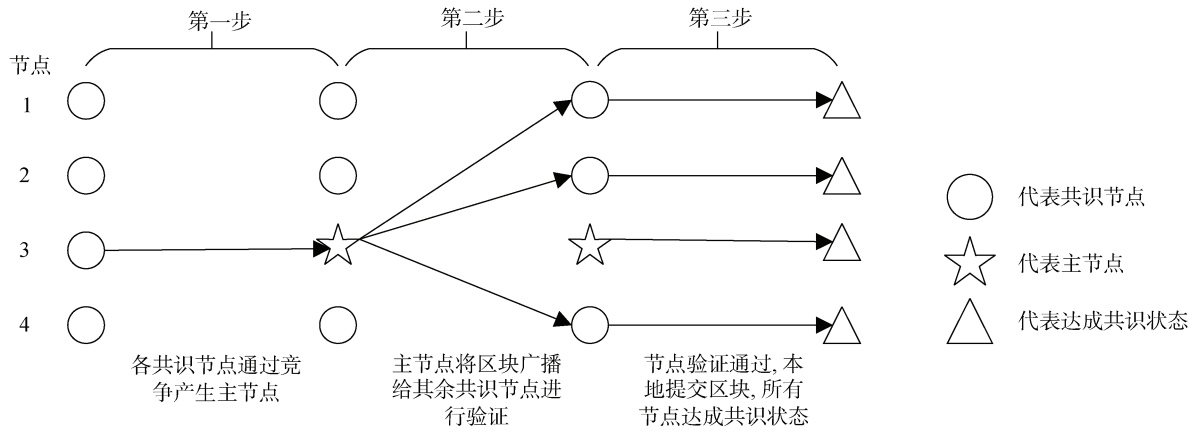


图 5　竞争类算法共识过程

Figure 5　The consensus process of competition classification algorithms

由于安全散列算法的单向不可逆性, 使得 PoW 算法难计算易验证。从 CPU 到 GPU 再到 ASIC 矿机, 随着算力不断增强, 越能快速求解满足条件的 *Nonce* 字段值, 竞争为主节点的概率也就越大。

PoW 计算密集型属性防止了恶意节点冒充多个节点来实施恶意攻击篡改交易数据等, 保障了区块链的安全; 同时带来的一个明显缺点就是耗费大量资源; 另外随着整个网络新区块产生难度的增加, 出于成本和收益的博弈, 部分亏损的小算力节点会退出比特币系统, 整个网络的新区块的生成趋向于算力高的节点, 中心化程度增加。Eyal 等[69]提出 PoW 共识还存在自私挖矿攻击的风险, 自私挖矿的收益比例大于其在全网的算力比例, 促使理性的节点加入自私挖矿矿池, 算力集中之后进一步增加 51%攻击的风险。

3.1.2　PoS

Peercoin 项目[63]中实现的 PoS 共识算法是为了降低算力的消耗。PoS 延续了 PoW 算法的竞争理念, 只不过相对于 PoW 中 *Nonce* 字段的大搜索空间而言, PoS 将搜索空间限制在一个计算量可接受的范围; 除此外, PoS 中还引入了“币龄”作为权益, 即:

$$Coinage = Coin \times Age$$

其中, *Coinage* 是“币龄”, *Coin* 为持有的货币, *Age* 为持续持有的时间。通过减小搜索空间以及引入“币龄”, PoS 将数学问题设计为:

$$F(BlockHeader \,|\, Timestamp) < T \times Weight$$

其中 F 为双 SHA256 哈希运算函数; *BlockHeader* 为区块头信息, 其中包含 *Timestamp* 字段, 取值范围是上一个区块时间和当前时间之间, 远小于 PoW 中 *Nonce* 字段的搜索空间大小; *Weight* 是用于竞争所消耗的“币龄”权重, T 是目标哈希值, 与 PoW 中的 T 相同。

PoS 的共识过程与 PoW 一致, 唯一不同是解决的数学问题不同, 其余过程均如图 5 所示。

与 PoW 相比, PoS 的数学问题中自变量的搜索空间减小, 同时不等式右侧引入“币龄”权重, 对于同一 T, 在每轮竞争中所投入的“币龄”越多, 权重越大, 竞争中获胜的概率也越大。

PoS 将算力竞争转化为权益竞争, 不仅节约算力, 权益的引入也能够防止节点发动恶意攻击; 同时促使所有节点有责任维护区块链的安全稳定运行以保障自身权益; PoS 虽然降低了算力资源的消耗, 但是没有解决中心化程度增强的问题, 新区块的生成趋向于权益高的节点。PoS 中需要拥有超全网一半的权益发动 51%攻击, 但其成本高于拥有超全网一半的算力[70], 另外创建区块需要消耗权益, 使得 PoS 持续进行 51%攻击的难度增加, 一定程度上降低了安全风险[31]。

3.1.3　PoSpace

Burstcoin 项目[65]中的共识算法为 PoSpace, 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。

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。普通节点根据“挑战”信息, 生成对应组合数据的哈希摘要, 由校验节点验证哈希值是否正确, 验证通过则说明普通节点确实存储了相应的数据。PoSpace 中定义了一个质量函数作为竞争指标:

$$Quality(hash,S)=(hash)^{1/S}$$

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 的共识过程第一步通过占用存储空间加入网络，然后对“挑战”得到的哈希结果计算质量函数，质量函数结果最小的节点成为主节点，将区块广播给其余未竞争成功的节点；第二步其余节点对接收到的新区块进行验证，验证通过后进入第三步将新区块提交到本地区块链，达成对最新高度区块的共识。从质量函数的定义可以看出，数据占用空间 S 越大，质量函数值越小，获胜的概率越大。

PoSpace 共识算法使用存储空间代替计算，节约电力资源；同时该共识协议下，节点初次接入网络时确定存储空间大小，之后不能扩容，防止了 PoW 共识算法中“矿池”(将不同节点的算力集合成一个大的算力节点)的出现，一定程度上避免了中心化程度增强的问题，同时降低了安全风险。

3.2 选举类

除了竞争方式选择主节点外，也可以通过选举的方式选择主节点。所有节点投票选择，根据投票结果，有的共识算法是产生一个主节点集合，由这些节点依次轮流成为主节点；有的共识算法是只产生一个主节点负责创建区块，下一轮重新投票。选举类的共识过程如图 6 所示，第一步是选出信任节点集合，信任节点依次轮流作为主节点；第二步是区块验证，区块验证方式分类两大类，一类与竞争类算法相似，主节点广播区块到其余节点进行验证，一类是主节点的区块经过三轮 BFT 类的投票过程进行验证；第三步是节点提交通过验证的区块，达成共识状态。本节选择授权股份证明算法(DPoS)、实用拜占庭容错算法(PBFT)、授权拜占庭容错算法(DBFT)进行描述，帮助理解选举类的共识算法。

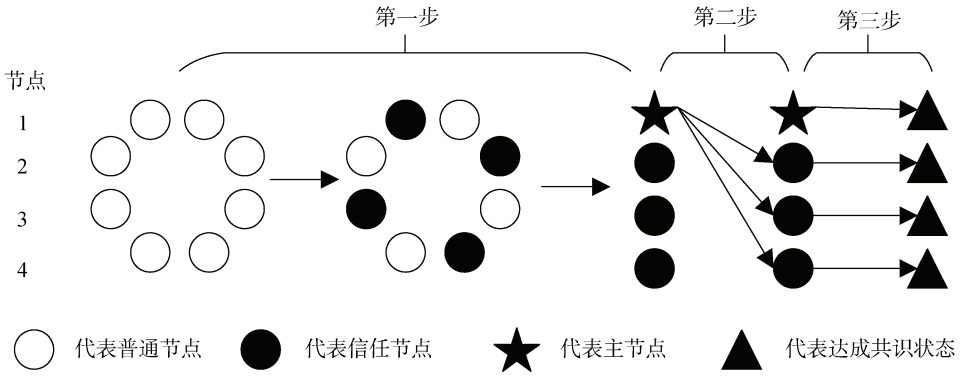


图 6 选举类算法共识过程

Figure 6 The consensus process of delegation classification algorithms

3.2.1 DPoS

EOS 项目[71]中实现的 DPoS 共识算法旨在解决 PoS 中主节点的产生趋向于高权益节点的问题，同时提高了 PoS 的共识效率。

DPoS 在 PoS 中引入选举机制。DPoS 将节点分为普通节点和信任节点两种角色。普通节点可以投票选择信任节点或者被投票成为信任节点。在 DPoS 中，节点消耗权益作为投票权，根据权益的加权结果产生最受信任的 N 个节点成为信任节点集合 $D\{D_1,\cdots,D_N\}$，每个信任节点 $D_{i,i=1\to N}$ 依次被赋予固定的期限(比如 2 秒)成为主节点，授权时间结束后，创建权依次交由下一个信任节点。信任节点集合 D 循环结束之后，重新选举调整 D，恶意节点将在下轮选举中失去其他节点的信任。

共识过程仅由信任节点集合 D 中所有节点参与，当下被授权的信任节点 D_i 创建新区块后广播到其余信任节点 $D_{j\neq i}$ 进行验证，在验证成功后将区块添加到本地，达成对最新高度区块的共识。

通过基于权益的民主投票产生信任节点集合，每个节点都可以参与到信任节点的选择上，D_i 轮流作为主节点，避免了主节点的产生趋向于高权益节点的问题，但选举的信任节点全权负责创建区块，这在一定程度上降低了去中心化程度；同时仅由 $D\{D_1,\cdots,D_N\}$ 进行验证，使得共识效率得到提升；另外，区块的产生不需要消耗算力，相对于 PoS 更加节省能耗。

3.2.2 PBFT

Fabric v0.6.0[72]中实现了 PBFT 共识算法。该共识算法从全网节点选择一个主节点负责创建区块，

然后经过三阶段投票达成共识: 预准备阶段, 准备阶段, 提交阶段。如图 7 所示, 主要过程如下:

预准备阶段: 从全网选择一个主节点; 每个节点把客户端发来的交易信息向全网广播, 主节点收集所有交易信息, 创建新区块并向全网广播;

准备阶段: 每个节点在收到主节点发送的区块信息后, 从预准备阶段进入到准备阶段, 节点对区块进行验证, 验证通过后向其他节点广播一条准备消息;

提交阶段: 节点在向全网广播准备消息之后进入提交阶段, 如果节点收到超过 2/3 节点的准备消息, 就向全网广播一条提交消息; 如果一个节点收到超过 2/3 节点的提交消息, 即可提交新区块到本地区块链, 达成对最新高度区块的共识。

PBFT 中无权益的抵押或竞争资源的消耗, 使得恶意节点的成本降低, PBFT 通过三阶段的投票机制排除恶意节点对共识的影响, 提供不超过 1/3 总节点数量的容错能力, 但 $O(N^2)$ 的通信复杂度使得系统性能随着网络规模的增加而下降。

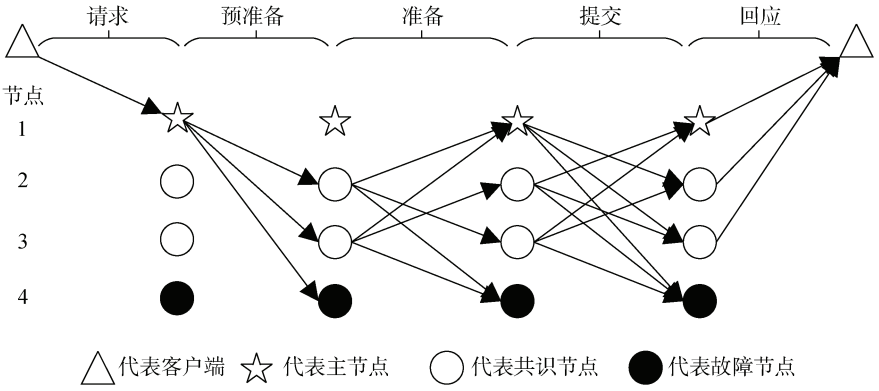


图 7 PBFT 三阶段示意图

Figure 7 The three-phase process diagram of PBFT

3.2.3 DBFT

NEO 项目[50]中提出 DBFT 算法。DBFT 改进自 PBFT 算法, PBFT 算法共识过程需要全部节点的参与, 且需要通过 3 轮网络请求确认来达成共识, 网络通信复杂度为 $O(N^2)$, N 为全网节点的数量, 可扩展性差, 随着网络规模增加, 难以快速达成一致。NEO 的解决方案是投票选取出部分节点参与共识, 由此减少网络通信消耗、提高可扩展性、同时提升交易处理速度。

DBFT 共识过程: 所有节点投票选举记账节点集合(记账节点即信任节点, 此处为了遵从原算法的描述), 记账节点集合完成共识过程, 其余节点为非记账节点, 不参与共识过程, 只采纳最终的共识结果。接下来由记账节点集合中选举出议长(议长即负责创建区块的主节点), 其他记账节点为议员; 在每轮共识中, 议长提出新区块, 由议员进行确认并广播确认结果; 当节点收到超过 2/3 记账人节点发送的确认消息后, 则在本地添加区块, 即对最新高度的区块达成共识; 之后重新选择议长, 开启新一轮共识过程。

DBFT 将共识过程的参与范围缩小为选举出来的记账节点集合, 虽然通信复杂度依然为 $O(N^2)$, 但是缩小了共识节点网络规模, 减少达成共识的通信成本。安全性上, DBFT 被证明在恶意节点少于总节点数量 1/3 的情况下也可能导致网络分叉[73]。DBFT 2.0 中[74]通过增加 commit 阶段, 避免出现分叉, 并增加两种状态恢复机制, 使得离线共识节点可以更快地从异常情况中恢复, 虽然提升了 DBFT 算法的可靠性, 但安全性依然没有得到彻底解决, 目前仍在完善。

3.3 随机类

除竞争、选举外, 也可以通过随机方式选择主节点。对于随机方式, 可以设计随机函数使得每个节点成为主节点的概率与节点所持有的某种资源成比例, 也可以设计随机函数使得每个节点成为主节点的概率相等。随机类的共识过程与竞争类共识过程相似, 如图 5 所示, 只是第一步通过随机算法随机选择主节点。本节描述衔尾蛇(Ouroboros)、消逝时间证明(PoET)两个共识算法, 帮助理解随机类的共识算法。

3.3.1 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 8 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

证明是不可被预知的。

$$F(S,\rho,sl_j)\to S_i$$

其中，S 为权益持有者候选人集合，ρ 为随机种子，sl_j 表示该 *epoch* 中第 j 个 *slot*，$S_i\in S$ 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。

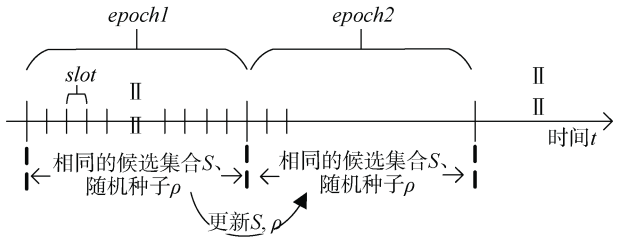


图 8　Ouroboros 示意图

Figure 8　The diagram of Ouroboros

每个节点根据当前 *epoch* 的种子 ρ，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。

与选举类相似，随机产生主节点的方式也会减少资源的消耗；但不同于选举类中主节点的产生需要节点之间网络通信进行投票，随机类使用随机算法产生主节点的方式无需网络通信，缩短了共识时间，提高了共识效率，增强可扩展性。Aggelos Kiayias 等[76]对 Ouroboros 的安全性给出了严密的证明，并提出了全新的奖励机制防止自私挖矿攻击。

3.3.2　PoET

PoET 虽然从名字看像是竞争类的共识算法，但本质上是通过随机时间长短来选择主节点。

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。第一，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；第二，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。

3.4　其他类

3.4.1　Ripple

Ripple 联盟链[77]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。

RPCA 共识算法能够容忍不超过 20%数量的拜占庭错误节点即可保证区块链系统的可用性；对于一致性而言，该共识算法对节点的 *UNL* 选择有如下要求：

$$\left|UNL_i\cap UNL_j\right|\geqslant\frac{1}{5}\max\left(\left|UNL_i\right|,\left|UNL_j\right|\right)\ \forall i,j$$

其中，$\left|UNL\right|$ 代表 *UNL* 中的节点数量。这个对 *UNL* 的要求称为连接性要求。举例说明该公式，如图 9 所示，图中不同形状代表产生分叉，最容易产生分叉的是这样一种网络连接：如图 9(a)所示，两个独立的子网，子网中的每个节点的 *UNL* 由子网的其余节点组成，两个子网之间没有连接。对于这种情况，任何一笔交易均可被子网 A 和子网 B 独立接受或拒绝，则会产生分叉；当 *UNL* 超过连接性要求时，即连接性大于 20%时，如图 9(c)所示，任何一笔交易要么都被

接受要么都被拒绝, 不会产生分叉。对于任何其他类型的网络连接, 在满足连接性要求的情况下更不会产生分叉。

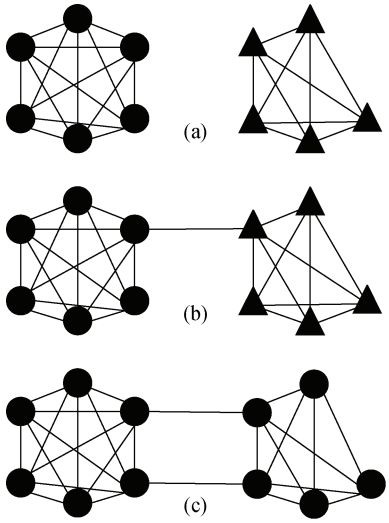


图 9 连接性示意图

Figure 9 The diagram of connectivity

3.4.2 分片共识算法

当前区块链面临的一大问题是交易性能低。竞争类和随机类共识算法中系统性能等价于单节点性能, 选举类共识算法由于 $O(n^2)$ 的通信复杂度会使得随着节点数量的增加整体的交易性能下降。分片技术中, 全网节点划分为不同的分组(委员会), 每个分组并行处理不同的交易集。其关键技术在于设计合理的分片方式以及解决分片交易的原子性问题。Elastico[55]是第一个基于分片技术的拜占庭容错公链方案, 之后出现众多改进方案, 比如 OmniLedger[56]、RapidChain[78]等。

以 Elastico 为例说明分片技术方案。Elastico 周期性地共识区块, 每个周期内, 节点执行以下操作:

1. 身份获取和委员会分配: 每个节点产生一个身份信息, 身份信息由公钥、IP 地址和一个 PoW 难题解组成, PoW 难题解是为了防止恶意节点伪造多个身份信息, 之后根据身份信息将节点划分到不同的委员会;

2. 委员会内部节点发现: Elastico 通过建立“目录服务委员会”, 节点可以高效地与同一委员会内部的其他成员建立通信连接;

3. 委员会内部共识: 委员会内部运行 PBFT 共识算法对交易集达成共识, 将共识结果发送到“共识委员会”;

4. 最终共识: “共识委员会”接受其他委员会的共识结果并重新打包交易, 运行 PBFT 共识算法达成最终共识并向全网广播;

5. 产生随机数: “共识委员会”产生一组随机数, 用于下一周期中节点重新构建身份信息。

分片技术通过并行处理提升系统性能, 使其随着全网节点数量的增多而线性提升; 另外每个节点只存储部分数据, 减少了节点数据存储压力。Elastico 中也存在一些问题, 比如没有解决原子性交换问题, 当分片拒绝某笔跨片交易后, 原分片中的交易资产被永远锁定。

3.4.3 基于 DAG 共识算法

除了分片技术外, 也可以通过有向无环图(Directed Acyclic Graph, DAG)的结构提升区块链系统的交易性能, 基于 DAG 的加密货币项目有 Obyte[79](Byteball)、DagCoin[80]、IOTA[81]等。DAG 中的每个单元代表用户发起的一笔交易, 每一条从子单元指向父单元的有向边代表一种验证关系, 即用户创建交易的时候需要对先前的交易单元进行验证, 将验证有效的交易单元的哈希值包含到自己的交易数据结构中, 则当前的交易单元称为子交易, 被验证的交易称为当前交易的父交易, 每笔交易可以有多个父交易, 每个交易也可以有多个子交易, 这样交易单元构成有向无环图的结构, 如图 10 所示。

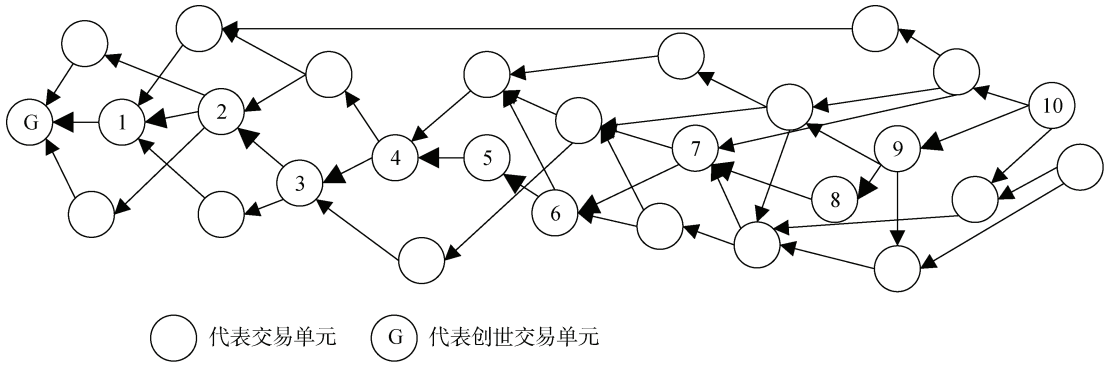


图 10 基于 DAG 共识数据存储图示

Figure 10 The diagram of data unit stored based on DAG

以 Obyte 为例展示基于 DAG 模式的交易过程:

1. 创建交易: 当用户需要发起交易时, 创建一个交易单元, 按照交易单元数据结构填充对应的内容, 包括消息类型、签名、以及一个或多个先前交易单元的哈希值;

2. 广播交易: 交易创建完成后, 将交易单元向全网广播;

3. 验证交易: 其他节点接收交易, 之后发起交易时会选择验证当前交易是否有效。

Obyte 系统中选择 12 个可信的“证人”节点, 这些节点发起的交易按照先后顺序连接成为一条主链, 如图 10 中按顺序编号的链为主链, 出现双花交易时, 根据交易单元与主链中交易单元的关系选择有效交易[80]。

从以上的过程中, 可以看出 DAG 模式中, 没有矿工和区块, 节省了交易费以及产生主节点打包区块的时间, 交易得到及时确认。这种异步并发处理交易的模式使得 DAG 具有高可扩展性, 但高并发情形下可能短时间内出现数据不一致的问题, 不适合强一致性的场景。

本节建立了共识算法的分类模型, 将共识算法分为四类: 竞争类、选举类、随机类和其他类型; 每种类型中描述典型的算法共识过程, 帮助更好地理解各共识算法及分类模型。在此基础上, 第四节从六个维度建立共识算法的评价指标, 并对各类型算法进行对比分析, 给出各类算法在不同性能上的表现。

4 评价体系

以太坊项目中, 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。区块链的去中心化是指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[82], 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。区块链系统中面临的安全性问题主要有密码学安全性、网络安全性、数据安全性、共识算法安全性等, 比如比特币系统的 51%算力攻击, 即恶意节点的算力超过全网一半的算力时, 这些恶意节点就可以联合起来对比特币系统强行分叉, 通过删除或修改交易信息构造新区块取代原来的正常交易数据, 实现双重支付等, 破坏比特币系统的安全性。区块链的可扩展性指区块链系统处理高业务量的能力。区块链可扩展性主要面临以下三点挑战[83]: (1)分布式系统的网络延迟, 区块链系统节点间网络距离以及网络环境无法控制, 所以在网络延迟上比传统的分布式系统受到更大的限制。(2)区块链的一致性, 区块链要维护节点间的一致性需要大量节点的确认。一般来说, 越是强的一致性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 这类共识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。同时随着区块链数据量的不断累加, 要想达到较高的交易率, 极大的数据量对于节点的吞吐量有很高的需求。

在比特币和以太坊中, 出于去中心化和安全性的考虑, 系统中每笔交易都需要全网节点广播, 全网节点验证, 并且每个新区块的创建都需要全网节点共同竞争, 提升去中心化程度和安全性; 共识过程需要全网节点的参与, 使得共识过程中网络通信复杂度增高, 系统处理性能低效, 限制了系统的可扩展性。EOS 项目中采用的 DPoS 共识算法, 采用 21 个超级节点参与共识过程, 这些被认为可信的超级节点按照顺序创建区块, 提升了区块链系统的性能, 可扩展性得到提升, 但是降低了区块链去中心化程度。

DSS 猜想仅是 Vitalik Buterin 用来解构区块链性能问题的方式, 理论上并不严谨和完整, 但依然给区块链共识算法的研究者们提供了一个性能研究的切入点。

本节将从去中心化、可扩展性、安全性、一致性、可用性、分区容忍性六个方面建立共识算法的评价指标体系。去中心化要求所有的节点具有相等的地位, 在共识算法中即要求所有节点能够参与共识过程且每个用户出块的权利越平均, 网络的控制权越分散, 去中心化程度越高。进一步将去中心化细化为共识节点数量、主节点选择方式、共识节点权重三个指标, 综合这三个指标建立去中心化程度指标。共识节点数量描述的是全网节点都参与共识过程还是部分节点参与共识过程; 主节点的选择方式描述主节点是通过竞争、选举或随机的方式产生; 共识节点权重描述在每轮共识过程中, 共识节点成为主节点的概率是否相等。可扩展性方面, 系统处理高业务量的能力越强, 可扩展性越强。处理高业务量的能力主要受到单个节点的性能和网络通信成本等因素的影响。进一步将可扩展性分为资源消耗、通信复杂度两个指标, 综合这两个指标对可扩展性程度进行评价。资源消耗是指共识过程对计算、存储、

网络等资源的消耗程度, 资源消耗限制单个节点的性能; 通信复杂度是指共识算法的网络通信复杂度, 旨在刻画网络通信成本(N 代表共识节点数量)。共识算法的安全性主要受到容错能力、对恶意节点的处理能力、潜在的攻击方式种类及数量、攻击难易程度、以及遭受安全攻击后的恢复能力等因素的影响。进一步将安全性细化为容错度、节点可控性、攻击多样性、攻击成本、安全恢复性, 综合这五个指标评估共识算法安全性。容错度是指共识算法保证区块链系统达成共识所能接受的拜占庭恶意节点(或算力等)在全网中的最大占比; 节点可控性是指共识算法是否具有排除恶意节点参与共识过程的能力; 攻击多样性是指共识算法可能面临的攻击方式的种类及数量, 种类及数量越多, 则潜在的安全风险越高, 目前区块链共识算法可能面临的安全攻击主要包括: 针对安全性假设的攻击、影响可扩展性的攻击和针对激励机制的攻击等; 攻击成本是指恶意节点发动安全攻击所消耗的代价, 代价越高, 发动攻击的难度越大; 安全恢复性是指区块链受到安全攻击后, 使数据恢复到攻击发生之前安全状态的能力, 进行硬分叉的可能性越高, 安全恢复性越强。去中心化程度、可扩展性和安全性是从 DSS 猜想的三个特性评价共识算法, 接下来从“CAP 定理”的一致性、可用性和分区容忍性三个特性评价共识算法。进一步将一致性层面分为分叉可能性、最终一致性。分叉可能性是指区块链产生分叉的概率大小; 最终一致性是指各节点的数据是否在有限时间内达成一致, 综合这两个角度刻画共识算法的一致性程度。可用性指标描述系统是否一直处于服务状态; 分区容忍性指系统是否允许部分节点不与其他节点通信。

以下基于评价指标体系, 从 6 个方面对各类算法进行对比分析, 如图 11 所示。去中心化程度方面, 竞争类全网节点全部参与共识过程, 但是每个节点成为主节点的概率不等, 与节点拥有的竞争资源有关, 这就造成在实际区块链系统中部分拥有大量竞争资源的节点垄断了主节点。选举类算法各节点投票选择主节点, 之后仅由选定的节点创建区块, 比如 EOS 中选出 21 个超级节点轮流负责创建区块, 这在一定程度上降低了去中心化程度。Ripple 算法每个节点维护一个信任节点列表, 各节点地位相等, 去中心化程度高。可扩展性方面, 竞争类共识算法属于资源消耗型, 单个节点的性能受到很大限制, 因此性能效率偏低, 处理大规模业务量能力差, 可扩展性程度低; 选举类共识算法虽然共识过程不需要消耗大量资源, 单个节点性能得到提升, 但是通信复杂度高, 且随着网络规模增加, 通信成本会限制可扩展性的提升, 但相比竞争类共识算法, 可扩展性得到提升; 随机类共识算法共识过程即不需要消耗大量资源, 网络通信复杂度也没有选举类算法高, 可扩展性比选举类更好; Ripple 算法通过将全网节点分为多个共识组提高了可扩展性。安全性方面, 竞争类共识算法容错度高于部分其他类算法, 选举类算法对节点的可控性强, 发现恶意节点之后可以在下一轮主节点的选举过程中将其排除, 竞争类算法面临的攻击多样性高, 以 PoW 为例, 安全性假设层面存在 51%攻击、日蚀攻击[84]等, 可扩展性层面存在粉尘攻击、空块攻击等; 激励层面存在自私挖矿攻击[69]、扣块攻击[85]等, 三个层面均面临潜在的安全攻击, 攻击多样性相对较高。选举类和随机类算法也存在一定的安全攻击, 但针对这两类算法的安全攻击方式相对较少, 相比于以 PoW 为代表的竞争类算法, 其攻击多样性相对较低。从攻击成本角度, 相比选举类和随机类, 竞争类算法中由恶意节点发动较强破坏性安全攻击的成本较高, 比如 PoS 中发动 51%攻击, 需要控制超过全网一半的权益, 发动安全攻击的难度较大。从安全恢复性角度, 选举类算法由选择的少数节点维护共识, 通过硬分叉进行安全恢复的能力强于竞争类和随机类算法。实际中, 采用 PoW 算法的比特币已经发生过数次大规模的硬分叉, 说明竞争类算法也具有一定的安全恢复能力。一致性方面, 竞争类算法由于网络延时的存在, 可能出现在短时间间隔内出现多个达到竞争标准的节点, 因此有分叉的可能性, 不具有最终一致性; 对于选举类和随机类, 算法在同一时间只产生一个主节点, 因此没有分叉的可能性, 具有最终一致性; Ripple 算法在满足连接性要求的情况下, 也不会产生分叉。一致性方面, 与竞争类共识算法相比, 选举类、随机类和 Ripple 算法具有强一致性。对于可用性和分区容忍性两个角度, 四大类共识算法均具有高可用性, 不支持分区容忍性, 满足“CAP 定理”。综上所述, 可以看出四大类算法在一致性、可用性和分区容忍性层面都致力于满足一致性和可用性, 牺牲了分区容忍性, 满足“CAP 定理”; 与“CAP 定理”只能满足其中两个特性不同, “DSS 猜想”是指去中心化程度、可扩展性、安全性三个层面的性能无法同时得到提升。竞争类算法具有更高的安全性, 在安全性不如竞争类的情况下, 随机类和 Ripple 算法提升了去中心化程度和可扩展性。因此, 网络环境更为复杂的公链中多采用竞争类算法, 比如规模最大的公链比特币采用 PoW, 以太坊拟采用 PoS; 而联盟链则多采用选

举类、随机类和其他类的共识算法, 比如 Hyperledger Fabric 中采用 PBFT 算法, Hyperledger Sawtooth 中采用的 PoET 算法, Ripple 联盟链中采用的 Ripple 共识算法。

评价指标 \ 算法分类		竞争类			选举类			随机类		其他类
		PoW	PoS	PoSpace	DPoS	PBFT	DBFT	Ouroboros	PoET	Ripple
去中心化	共识节点数量	全网	全网	全网	部分	全网	部分	全网	全网	全网
	主节点选择方式	竞争	竞争	竞争	选举	选举	选举	随机	随机	—
	共识节点权重	不等	不等	不等	相等	相等	相等	相等	相等	相等
	去中心化程度	高	高	高	中	中	中	高	高	高
可扩展性	资源消耗	高	中	中	低	低	低	低	低	低
	通信复杂度	O(N)	O(N)	O(N)	O(N)	O(N^2)	O(N^2)	O(N)	O(N)	O(N)
	可扩展性程度	中	中	中	高	高	高	高	高	高
安全性	容错度	<50%	<50%	<50%	<50%	<1/3	<1/3	<50%	<50%	<20%
	节点可控性	低	低	低	高	高	高	低	低	低
	攻击多样性	高	高	高	中	中	中	中	中	中
	攻击成本	高	高	高	中	中	中	中	中	中
	安全恢复性	高	中	中	高	高	高	中	中	中
	安全性程度	高	高	高	中	中	中	中	中	中
一致性	分叉可能性	有	有	有	无	无	无	无	有	无
	最终一致性	否	否	否	是	是	是	是	否	是
	一致性程度	弱	弱	弱	强	强	强	强	弱	强
可用性	可用性	高	高	高	高	高	高	高	高	高
分区容忍性	分区容忍性	否	否	否	否	否	否	否	否	否

图 11　共识算法评价指标

Figure 11　The evaluating indicators of consensus algorithms

本节从去中心化、可扩展性、安全性、一致性、可用性、分区容忍性六个方面建立了共识算法的评价体系并完成了初步的定性分析。Garay 等 [86]提出比特币骨干协议, 并对比特币的一致性(Persistence)和可用性(Liveness, 也称活性)进行定量分析。该协议中提出公共前缀和链质量两种特性, 并得出一定算力约束条件下这两个特性的概率大小, 进而量化比特币的一致性和可用性, 分析得出在网络同步速率大于出块速率且恶意节点的算力严格小于全网算力50%的条件下, 比特币具有高一致性和可用性。比特币骨干协议分析、抽象比特币协议, 提取关键元素、提出并证明新特性, 并通过重新实现比特币, 借助新特性来证明比特币的相关特性, 这种分析方法为后续对共识算法评价指标体系设计对应的定量分析方法提供了有效参考。

5　总结与展望

共识算法是区块链系统的核心机制之一, 获得了研究人员和产业界的广泛关注。近年来区块链项目中涌现出各种共识算法, 但缺乏完整技术分类和统一评价体系。本文深入分析区块链共识算法, 从共识过程本身出发, 提出了共识算法的分类模型, 按照主节点的产生方式将共识算法分为竞争类、选举类、随机类及其他类型, 并剖析了每种类型中代表性共识算法的共识过程及其优劣势; 同时从去中心化、可扩展性、安全性、一致性、可用性、分区容忍性六个方面建立了一套共识算法的评价指标体系, 在此基础上, 对各类共识算法进行初步的对比分析。

后续的研究重点将在此评价指标体系和评价分析的基础上, 结合区块链的应用场景, 借鉴比特币骨干协议等的分析方法, 针对共识算法设计对应的评测方法, 进一步开展科学化分析, 给出区块链共识算法在应用场景中的定量评价, 为不同的应用场景选择合适的共识算法提供参考。

参考文献

[1] Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://people.eecs.berkeley.edu/~raluca/cs261-f15/readings/bitcoin.pdf.2008

[2] Birch D. The Digital Money Decade[J]. *The Journal of Internet Banking and Commerce*, 2007, 12(1): 1-4.

[3] Michael M, Mills S. The Missing Links in the Chains?Mutual Distributed Ledger (aka Blockchain)Standards[J]. *Social Science Electronic Publishing,* 2016, 8(3): 11-15.

[4] Becker W E, Kreps D M. Game Theory and Economic Modelling[J]. *The Journal of Economic Education*, 1992, 23(2): 189.

[5] Fox G. Peer-to-peer Networks[J]. *Computing in Science & Engineering*, 2001, 3(3): 75-77.

[6] Diffie W, Hellman M. New Directions in Cryptography[J]. *IEEE*

Transactions on Information Theory, 2006, 22(6): 644-654.

[7] Rivest R L, Shamir A, Adleman L. A Method for Obtaining Digital Signatures and Public-key Cryptosystems[J]. *Communications of the ACM*, 1978, 21(2): 120-126.

[8] McEliece R J. A public-key cryptosystem based on algebraic coding theory[C], *DSN Progress Report*, 1978: 42-44.

[9] Rolim J, Mueller F, Zomaya Y. Parallel and Distributed Processing[J]. *Lecture Notes in Computer Science*, 1993, 5(3): 217-221.

[10] Lamport L, Shostak R, Pease M. The Byzantine Generals Problem[J]. *ACM Transactions on Programming Languages and Systems*, 1982, 4(3): 382-401.

[11] 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.

[12] Gilbert S, Lynch N. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services[J]. *ACM SIGACT News*, 2002, 33(2): 51-59.

[13] Yuan Y, Wang F Y. Blockchain: The State of the Art and Future Trends[J]. *Acta Automatica Sinica*, 2016, 42(4): 481-494.
(袁勇, 王飞跃. 区块链技术发展现状与展望[J]. *自动化学报*, 2016, 42(4): 481-494.)

[14] Castro M, Liskov B. Practical Byzantine fault tolerance[C]. *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation*(OSDI), 1999: 173−186.

[15] Oki, Brian M, Liskov B. H. Viewstamped Replication: A New Primary Copy Method to Support Highly-Available Distributed Systems[C]. *Acm Symposium on Principles of Distributed Computing ACM*(PODC'88), 1988: 8-17.

[16] Lamport L. Time, Clocks, and the Ordering of Events in a Distributed System[J]. *Communications of the ACM*, 1978, 21(7): 558-565.

[17] Lamport L. The Part-time Parliament[J]. *ACM Transactions on Computer Systems*, 1998, 16(2): 133-169.

[18] Pires M, Ravi S, Rodrigues R. Generalized Paxos Made Byzantine (and less Complex)[M]. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017: 203-218.

[19] Lampson B W. How to Build a Highly Available System Using Consensus[M]. Distributed Algorithms. Springer Berlin Heidelberg, 1996: 1-17.

[20] Chang F. W, Dean J, Ghemawat S. Bigtable: A Distributed Storage System for Structured Data[J]. *ACM Transactions on Computer Systems*, 2008, 26(2): 4-26.

[21] Gafni E, Lamport L. Disk Paxos[J]. *Distributed Computing*, 2003, 16(1): 1-20.

[22] Lampson B, Lomet D. B. Distributed transaction processing using two-phase commit protocol with presumed-commit without log force[J]. 1992.

[23] Skeen D, Stonebraker M. A Formal Model of Crash Recovery in a Distributed System[J]. *IEEE Transactions on Software Engineering*, 1983, SE-9(3): 219-228.

[24] Kotla R, Alvisi L, Dahlin M, et al. Zyzzyva: Speculative Byzantine Fault Tolerance[C]. *The twenty-first ACM SIGOPS symposium on Operating systems principles - SOSP '07,* 2007, 41(6): 45-58.

[25] Kotla R, Alvisi L, Dahlin M, et al. Zyzzyva: Speculative Byzantine Fault Tolerance[C]. *The twenty-first ACM SIGOPS symposium on Operating systems principles - SOSP '07*, 2007, 41(6): 45-58.

[26] Junqueira F, Reed B, Serafini M. Zab: High-performance broadcast for primary-backup systems[C]. *The dependable systems and networks*, 2011: 245-256.

[27] Kafka official website. https://kafka.apache.org/documentation/. Jan. 2019.

[28] Ongaro D, Ousterhout J. K. In search of an understandable consensus algorithm[C]. *The usenix annual technical conference*, 2014: 305-320.

[29] Dwork C, Naor M. Pricing via Processing or Combatting Junk Mail[M]. Advances in Cryptology — CRYPTO' 92. Berlin, Heidelberg: Springer Berlin Heidelberg, : 139-147.

[30] Back A. Hashcash - A Denial of Service Counter-Measure. http://www.hashcash.org/papers/hashcash.pdf. Jan. 2019.

[31] King S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake. https://peercoin.net/assets/paper/peercoin-paper.pdf. Nov. 2014.

[32] Schwartz D, Youngs N, Britto A. The Ripple protocol consensus algorithm. https://ripple.com/files/ripple consensus whitepaper.pdf. Jan. 2019.

[33] The stellar consensus protocol: A federated model for internet-level consensus. https://www.stellar.org/cn/papers/stellar-consensusprotocol.pdf. May. 2019.

[34] Bitshares. Delegated Proof of Stake. http://docs.bitshares.org/ bitshares/dpos.html. Jan. 2019.

[35] The sleepy model of consensus. https://eprint.iacr.org/2016/918.pdf. May. 2019.

[36] Kiayias A, Russell A, David B, et al. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol[M]. Advances in Cryptology – CRYPTO 2017. Cham: Springer International Publishing, 2017: 357-388.

[37] Tendermint: Consensus without mining. https://tendermint.com/static/docs/tendermint.pdf. May. 2019.

[38] Miller A, Xia Y, Croman K, et al. The Honey Badger of BFT Protocols[C].*The 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016: 31-42.

[39] Tangaroa: a byzantine fault tolerant raft. http://www.scs.stanford.edu/14aucs244b/labs/projects/copeland zhong.pdf. May. 2019.

[40] The first scalable. high performance private blockchain.

http://kadena.io/docs/Kadena-ConsensusWhitePaperAug2016.pdf. May. 2019.

[41] Proof of stake velocity: Building the social currency of the digital age. https://assets.coss.io/documents/whitepapers/reddcoin.pdf. May. 2018.

[42] Ateniese G, Bonacina I, Faonio A, et al. Proofs of Space: When Space is of the Essence[M]. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014: 538-557.

[43] Proof of burn. https://en.bitcoin.it/wiki/Proof of burn. Apr. 2018.

[44] Bentov I, Lee C, Mizrahi A, et at. Proof of activity: Extending bitcoins proof of work via proof of stake. http://eprint.iacr.org/2014/452. Jan. 2019.

[45] Buntinx P. What is Proof of Elapsed Time?. https://themerkle.com/what-is-proof-of-elapsedtime/. Jan. 2019.

[46] Proof of luck: an ecient blockchain consensus protocol. https://eprint.iacr.org/2017/249.pdf. May. 2019.

[47] Proofs of Useful Work. https://allquantor.at/blockchainbib/pdf/ball2017proofs.pdf. May. 2019.

[48] Proof of authority. https://wiki.parity.io/Proof-of-Authority-Chains. May. 2019

[49] Introducing Casper “the Friendly Ghost”. https://blog.ethereum.org/2015/08/01/introducing-casper-friendlyghost/. May. 2019.

[50] NEO offical website. https://neo.org/. Sept. 2019.

[51] delegated Byzantine Fault Tolerance. https://github.com/neo- project/docs/blob/master/en-us/basic/consensus/whitepaper.md. Jan. 2019.

[52] Gilad Y, Hemo R, Micali S, et at. Scaling byzantine agreements for cryptocurrencies. http://eprint.iacr.org/2017/454. Jan. 2019.

[53] Eyal I, Gencer A. E, Sirer E. G, et al. Bitcoin-NG: A scalable blockchain protocol[C]. *The 13th Usenix Conference on Networked Systems Design and Implementation*, 2016: 45−59.

[54] Kokoris-Kogias E, Jovanovic P, Gailly N, et al. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing[EB/OL]. 2016: arXiv:1602.06997[cs.CR]. https://arxiv.org/abs/1602.06997

[55] Luu L, Narayanan V, Zheng C D, et al. A Secure Sharding Protocol for Open Blockchains[C]. *The 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016: 17-30.

[56] Kokoris-Kogias E, Jovanovic P, Gasser L, et al. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding[C]. *IEEE Symposium on Security and Privacy*(SP), 2018: 583-598.

[57] Ontology. https://ontio.github.io/documentation/tutorial_for_developer_en.html. Jan. 2019.

[58] Verifiable Byzantine Fault Tolerance. https://ont.io/wp/Ontology-technology-white-paper-ZH.pdf. Jan. 2019.

[59] Micali S, Rabin M. O, Vadhan S. P. Verifiable random functions[C]. *foundations of computer science*(FOCS), 1999: 120-130.

[60] Snowflake to Avalanche: A Novel Metastable Consensus Protocol Family for Crypto. https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV. Jan. 2019.

[61] Buterin V. A Guide to 99% Fault Tolerant Consensus. https:// vitalik.ca/general/2018/08/07/99_fault_tolerant.html. Jan. 2019.

[62] YEE. http://dcdn.yeecall.com/yee/YEECO_Technical_White_Paper_ZH_v0.1_Draft.pdf. Jan. 2019.

[63] Peercoin official website. https://peercoin.net/. Jan. 2019.

[64] Parity official website. https://www.parity.io/. Jan. 2019.

[65] Burstcoin official website. https://www.burst-coin.org/. May. 2019.

[66] Gochain official website. https://gochain.io/. Jan. 2019.

[67] Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network. https://link.springer.com/content/pdf/10.1007%2F978-3-319-91458-9_41.pdf. Jan. 2019.

[68] Verification of a Cryptographic Primitive: SHA-256. http://www.cs.princeton.edu/~appel/papers/verif-sha.pdf. Jan. 2019.

[69] Eyal I, Sirer E G. Majority is not Enough: Bitcoin Mining is Vulnerable[M]. Financial Cryptography and Data Security. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014: 436-454.

[70] The Security and Development of BlockChain Consencus Protocol. http://www.btb8.com/blockchain/1903/37593.html. May. 2019.

[71] EOS officail website. https://eos.io/. Jan. 2019.

[72] Fabric official website. https://get.fabric.io/. Jan. 2019.

[73] The analysis and improvenment of consensus of NEO DBFT. http://blogs.360.cn/post/NEO_dBFT.html. May. 2019.

[74] NEO Global Development. https://neo.org/blog/details/4132. May. 2019.

[75] Cardano official website. https://www.cardano.org/en/home/. Jan. 2019.

[76] Kiayias A, Russell A, David B, et al. Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol[M]. Advances in Cryptology – CRYPTO 2017. Cham: Springer International Publishing, 2017: 357-388.

[77] Ripple official website. https://ripple.com/. Jan. 2019.

[78] Zamani M, Movahedi M, Raykova M. RapidChain: Scaling Blockchain via Full Sharding[C]. *The 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018: 931-948.

[79] Byteball official whitepaper. https://obyte.org/Byteball.pdf. Jan. 2019.

[80] DagCoin official whitepaper. https://dagcoin.org/whitepaper.pdf. Jan. 2019.

[81] IOTA official whitepaper. https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf. Jan. 2019.

[82] Pass R, Shi E. FruitChains: A Fair Blockchain[C]. *The ACM Symposium on Principles of Distributed Computing,* 2017: 315-324.

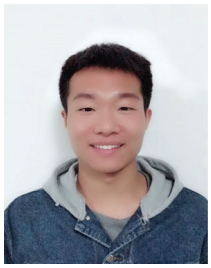
[83] The key technology: applications and challenges of blockchains.

http://www.cicpa.org.cn/Column/hyxxhckzl/xxjsyqy/qyjs/201807/W020180716535877349534.pdf. Jan. 2019.

[84] Heilman E, Kendler A, Zohar A et al. Eclipse Attacks on Bitcoin's Peer-to-Peer Network[C]. *USENIX Security Symposium*, 2015: 129-144.

[85] Courtois N T, Bahack L. On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency[EB/OL]. 2014: arXiv:1402.1718[cs.CR]. https://arxiv.org/abs/1402.1718

[86] Douceur J R. The Sybil Attack[M]. Peer-to-Peer Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002: 251-260.



靳世雄 于 2013 年在西南交通大学通信工程专业获得 学士学位。现在中国科学院信息工程研究所, 第五实验室攻读硕士学位。研究领域为网络空间安全、区块链。Email: jinshixiong@iie.ac.cn

张潇丹 于 2012 年中国科学院大学计算机系统结构专业获得 博士学位。 现任 中科院信工所副研究员, 研究领域为软件定义网络, 新型网络技术测量分析与评估, 区块链, 数据分析。Email: zhangxiaodan@iie.ac.cn

葛敬国 于 2003 年获中国科学院计算技术研究所计算机系统结构专业获得博士学位。现任中国科学院信息工程研究所研究员, 中国科学院网络安全学院教授。研究领域是计算机网络架构, 软件定义网络, 网络虚拟化, 网络安全和移动通信网络。Email: gejingguo@iie.ac.cn

史洪彬 于 2012 年北京交通大学通信与信息系统专业获得 硕士学位。 现任 中国科学院信息工程研究所工程师, 研究领域为云计算, 网络虚拟化, 区块链。Email: shihongbin@iie.ac.cn

孙毅 于 2007 年中国科学院大学计算机系统结构专业获得 博士学位。 现任 中国科学院计算技术研究所研究员, 研究领域为 区块链、分布式应用。Email: sunyi@ict.ac.cn

李鸣 于 2003 年在丹麦奥尔堡大学软件工程专业获得硕士学位。 现任中国电子技术标准化研究院区块链研究室主任。研究领域为区块链、人工智能。研究兴趣包括: IT 治理、数据治理、IT 架构。Email: 13701388822@139.com

林业明 于 2017 年在华南农业大学信息与计算专业获得 学士学位。现在中国科学院信息工程研究所攻读硕士学位。 研究领域为区块链技术、信息安全、计算机网络。研究兴趣包括:计算机应用。Email: linyeming@iie.ac.cn

姚忠将 于 2015 年在首都师范大学计算机应用技术专业获得 硕士学位。 现在中国科学院大学计算机系统结构专业攻读博士学位。 研究领域为网络安全、区块链与隐私保护。研究兴趣包括: 流量监管、共识算法等。Email: yaozhongjiang@iie.ac.cn