计算机科学
COMPUTER SCIENCE

区块链共识算法综述

谭朋柳，王润庶，曾文豪，王诗堃，邹雯诗

引用本文

谭朋柳，王润庶，曾文豪，王诗堃，邹雯诗区块链共识算法综述[J]．计算机科学, 2023, 50(6A): 220400200-12.

TAN Pengliu, WANG Runshu, ZENG Wenhao, WANG Shikun, ZOU Wenshi. Overview of Blockchain Consensus Algorithms [J]. Computer Science, 2023, 50(6A): 220400200-12.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220400200

区块链共识算法综述

谭朋柳　王润庶　曾文豪　王诗堃　邹雯诗
南昌航空大学软件学院　南昌 330063

摘　要　共识算法维持着分布式系统的稳定和安全,同时又是发展区块链方向的关键技术。随着区块链技术快速发展,共识算法的研究也越来越受到研究人员的重视和青睐。现如今,在不同应用场景下选择合适的共识算法是研究人员所要面对的一个选择性难题。主要从服务对象节点种类出发,把共识算法归类为公有链、联盟链和私有链这3个大分类。在这3个大分类的基础之上,分别阐述了现在主流的和一些新的区块链共识算法的基本原理,共列举了9种共识算法,并从去中心化、安全性和可扩容性这3个方面对这9种共识算法进行性能评估。并且对相关算法进行了优缺点的分析总结,给出了优化区块链共识算法的相关方向,以供研究人员研究和参考,从而促进区块链共识算法的稳步发展。

关键词:区块链;共识算法;公有链;联盟链;私有链;优缺点
中图法分类号　TP301

Overview of Blockchain Consensus Algorithms

TAN Pengliu,WANG Runshu,ZENG Wenhao,WANG Shikun and ZOU Wenshi
School of Software,Nanchang Hangkong University,Nanchang 330063,China

Abstract　Consensus algorithm not only maintains the stability and security of distributed system,but also is the key technology in the development direction of blockchain. With the rapid development of blockchain technology,the research of consensus algorithm has attracted more and more attention and favor of researchers. Nowadays,choosing an appropriate consensus algorithm in different application scenarios is a selective problem that researchers have to face. Starting from the types of service object nodes,this paper classifies the consensus algorithm into three categories:public chain,alliance chain and private chain. Based on these three categories,this paper expounds the basic principles of some mainstream and some new blockchain consensus algorithms,a total of 9 consensus algorithms,and evaluates the performance of these 9 consensus algorithms from three aspects:decentralization,security and scalability. This paper also analyzes and summarizes the advantages and disadvantages of the relevant algorithms,and gives the relevant directions to optimize the blockchain consensus algorithm for researchers' research and reference,so as to promote the steady development of blockchain consensus algorithm.

Keywords　Blockchain,Consensus algorithm,Public chain,Alliance chain,Private chain,Advantages and disadvantages

1　引言

在金融危机的背景下,2008年中本聪发表了“比特币”白皮书[1]。2009年1月3日,中本聪在比特币系统的创世区块中留下这样一句话:“the times 03/jan/2009 chancellor on brink of second bailout for banks”[2]。借此可以看出,中本聪想凭借区块链技术来解决传统金融系统所解决不了的问题。随后区块链逐渐进入人们的视野。

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,由一块一块的区块连接组成一条最长链,而链条就是由哈希指针组成,按照区块产生的时间形成的一种链式结构[3-4],区块链可以看作是一本共享账本在分布式系统中[5]。现阶段,区块链的快速发展已使得区块链走进了人们的生活之中。例如金融服务[6-7]、供应链技术[7-8]、政府治理[9]、游戏服务[10]等基于区块链技术的产品已经出现并得到成功运用。区块链颠覆了传统的创新商业模式,引发了技术创新[11-13]。但要满足更多场景,还需解决区块链中存储大小以及共识算法中高吞吐量、高容错性、高安全性和低资源消耗等问题。

本文首先从节点参与网络的方式将共识算法分为3类,进而阐述3类共识算法的原理。最后从3个大的方面(去中心化程度(Decentralization)、安全性(Security)和可扩展性(Scalability))去比较和评价本文所阐述的9种共识算法。最终总结出它们的优缺点。在此基础之上也总结出了几个优化区块链共识算法方面的建议,给相关研究人员提供参考。

2　区块链的分类

最早公有链的提出是在中本聪发布“比特币”[1]后,之后根据不同的应用场景和各条链的特点,又在公有链的基础之上为区块链添加了两个新的分类,分别为:联盟链和私有链。

公有链中的节点非常自由,可以自由加入和离开节点间

基金项目:国家自然科学基金(61961029);江西省科技厅重点研发计划(20171ACE50025)
This work was supported by the National Natural Science Foundation of China(61961029)and Key Research Plan of Jiangxi Province Department of Science and Technology(20171ACE50025).
通信作者:谭朋柳(pltan@nchu.edu.cn)

的共识过程。正因如此,公有链中的共识算法需要更完善的安全机制来保证公有链正常运行,而不是会因此而崩溃,在提高安全性的同时,公有链牺牲的是共识速度和共识吞吐量。

联盟链中的节点需要得到各个组织或联盟的授权才能加入或退出节点间的共识,而不像公有链一般,任何节点都可进行验证区块和生成区块。联盟链将部分同级的参与节点一起视为验证者,是多方实体一起对区块进行验证。一般情况下,联盟链的节点代表相应的机构或联盟。故联盟链相对于公有链来说去中心化程度较低,联盟链中的共识算法需要提高共识的吞吐量,减少系统资源的消耗。

私有链中的节点参与共识也需要某个机构来授权,与联盟链所属情况不同的是,该机构仅仅是一个单一的实体,而不是多个实体,私有链通过一个相对封闭的系统只允许一个实体来进行任命主节点,进而由主节点产生区块。本质上来说,联盟链就相当于由私有链组成而成,只是私有的程度不同。私有链对交易效率、安全性能等多个方面有更高要求,故私有链相对于联盟链来说又更加中心化。其次,相对于联盟链共识算法来说,私有链共识算法所需吞吐量应得到进一步的提升,交易效率需更高。

表 1 从 7 个方面总结了公有链、联盟链和私有链之间的主要区别,如表 1 所列。

表 1 公有链、联盟链和私有链的比较

Table 1 Comparison of public chain, consortium chain and private chain

类别	公有链	联盟链	私有链
管理节点组织	公共管理	部分机构(多方实体)	个人或单个机构(单一实体)
参与方式	自由参与	需审核	需审核
节点身份	匿名	公开	公开
透明度	高	较低	低
共识节点范围	大	较小	小
网络连接性	低	较高	高
网络同步	异步/部分同步	部分同步/同步	部分同步/同步
交易吞吐量	低	较高	高
应用案例	Bitcoin, Blackcoin, Ethereum 等	Hyperledger Fabric, NEO, R3 Coda 等	R3 CodaQuorum, Zookeeper, GoogleChubby 等

3 共识算法的分类

在分布式系统中共识算法是核心部分,它让处于复杂网络环境下的节点在不造成分叉的情况下达到数据一致性。节点间的共识过程应该分为 4 个阶段:加入共识阶段、出块阶段、进行验证投票阶段和推出共识阶段。其中出块阶段和进行验证投票阶段是这 4 个阶段中的核心阶段。

从节点参与网络方式的不同,可以把区块链分为公有链、联盟链和私有链[14]。每种链都有各自的特点,链中的共识算法也需满足不同链所在环境的需求,从而给出一个具体的分类体系。

根据公有链的特点,公有链所需的共识算法需要安全性高和允许出现拜占庭节点这两种特性来保证公有链的正常运作。

联盟链所需的共识算法需要在性能消耗尽可能少和允许一定数量拜占庭节点的情况下保证有多个实体来共同管理链。

私有链所需的共识算法需要在节点不出现拜占庭节点的情况下,有运行性能高、能耗小等特点,优先考虑系统的速度,在中心化高的情况下管理链。

从本质上来看,公有链、联盟链和私有链并不互相矛盾,每种链所应用的共识算法也不相互矛盾,最重要和最基本的一个原则是选择合适的算法运用到合适的环境下,达到物尽所用的效果即可。

故本节将在分类体系的准则上来对相关共识算法的原理进行阐述。本文挑选了 9 种共识算法分别进行了讨论,9 种共识算法中不仅包含了一些经典的共识算法,还包含了近些年被广泛运用和非常具有创新性的共识算法。虽然这 9 种共识算法不能覆盖全部共识算法的总体类型,但能覆盖如今共识算法发展和创新所需的主体方向。

3.1 公有链

公有链也可以称为“公链”,公有链中所有的节点都是自由进出的,所有的节点都可以打包交易参与共识,无需有关节点允许。公有链中没有任何集中的组织。其中 PoW, PoS, DPoS 等算法是公有链算法中主流的共识算法。

3.1.1 PoW

PoW(Proof Of Work) 共识算法,也可以称为工作量证明。PoW 算法一开始是用来预防垃圾邮件[15-16]的一种算法,即发送邮件需要进行一个小的运算阶段,人为延迟发送邮件速度。发件人每天发几封邮件感觉不出延迟,若每天发上千封垃圾邮件则发件人可以明显感觉到发送效率降低。后来,中本聪巧妙地把 PoW 算法应用到比特币上面,在 2008 年发表的《Bitcoin: A Peer-to-Peer Electronic Cash System》论文[1]中,解决了分布式系统达到共识的问题。

PoW 算法所用到的哈希函数为 SHA-256,此函数满足密码学中的 3 个性质[17]:Collision(抗碰撞性)、Hiding(隐蔽性)和 Puzzle friendly(迷惑友好性)。Collision 就是指两个不同值的输入,经过哈希运算所得到的结果却是相等的,则说明产生了哈希碰撞。Hiding 指哈希的过程是不可逆的,得到的哈希值不可以反推出这个值。Puzzle friendly 指哈希值在计算之前是不可知、不可预测的,想要某个值的哈希在一定的范围内就必须一个一个去尝试,没有捷径。因此 PoW 算法使用的哈希函数为 SHA-256。

PoW 算法在比特币系统中的运作过程是对区块头不断地做双重 SHA256 哈希运算,直到计算结果小于目标挖矿难度即可。区块头中包括父区块哈希值、版本、时间戳、Nonce 值、nBits、Target 和 Merkel 根等信息[18]。PoW 算法要达到的目标公式具体如下:

$$SHA256(SHA256(\text{父区块哈希值}(32byte)+\text{版本号}(4byte)+\text{时间戳}(4byte)+Nonce\ \text{值}(4byte)+nBits(4byte)+Merkel\ \text{根}(32byte)))<Target \quad (1)$$

其中,父区块哈希值大小为 32 字节,即上一个区块的哈希值,指向前面一个区块的哈希指针;版本大小为 4 字节,即当前版本的版本号;时间戳大小为 4 字节,即生成这个区块的时间信息,精确到秒的 UNIX 时间戳;Nonce 值大小为 4 字节,初始值为 0,Nonce 值可以不断增加,是工作量证明的关键;nBits 大小为 4 字节,存储的是压缩格式的当前目标 Hash 值,即本区块的难度值;Merkel 根大小为 32 字节。在区块所收集的交易中,对每个交易进行哈希,再对两两交易的哈希值进行哈希,一直重复第二个步骤,直至哈希值为一个,这个哈希值称

为 Merkel 根。这个计算过程称为 Merkel 树，树的叶子节点必须为偶数，若叶子节点为奇数则需将最后一个交易复制一份，再进行上面的步骤。具体过程如图 1 所示。

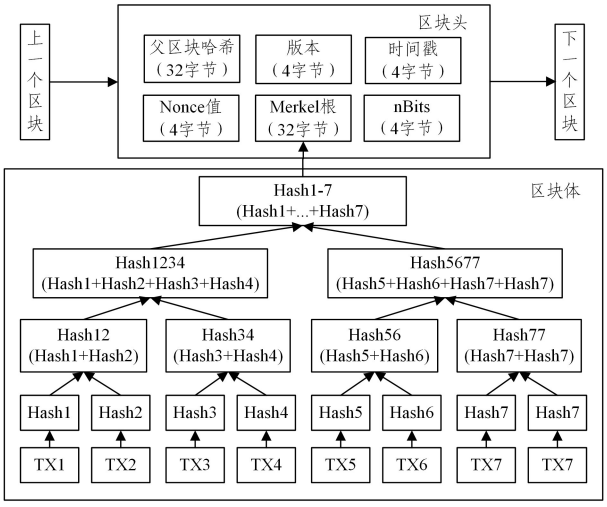

图 1　Merkel 树形成过程

Fig. 1　Merkel tree formation process

Target：大小为 4 字节。Target 即目标阈值。当前区块的哈希值必须小于这个阈值，寻找 Nonce 值的难度与 Target 成反比。具体公式如下：

$$D=\frac{\mathrm{Tm}}{T} \tag{2}$$

其中，D 是难度值，以压缩的形式存储在目标阈值编码（nBits）中。Tm 为最大目标值，存储于创世区块中，是个常数，大小为 0x1D00FFFF。T 为目标阈值，T 不会等于 0，T 的最小值为 1。可以看出目标阈值 T 越小难度值 D 就越大，T 为最小值时，难度值最大。

PoW 共识算法的核心是不断地尝试 Nonce 值，进而改变双哈希区块头后所得到的哈希值，再和目标阈值作比较。如果满足目标公式则将区块广播给其他节点验证，通过验证的节点会停止本地工作量证明运算，立即进入下个区块 Nonce 值寻找。如果区块经全网 51%以上节点验证通过，则区块会上链到主链上。区块上链到主链后，此区块的创造者拥有记账权且可以获得相应的出块奖励和交易费，以此来调动参与者参与和维护区块链安全的积极性。比特币系统中一个区块的产出一般要控制在 10 min，每出 2016 个区块就会调整一次难度。调整难度公式为：

$$T=\frac{T_a}{T_e} \tag{3}$$

其中，T 为 Target。T_a 为 actual-time 实际产生 2016 个区块所花费的时间。T_e 为 expected-time 产生 2016 个区块本应花费的时间，2 016 个区块中每产生一个区块需 10 min，故本应花费的时间是两周。如果 T_a 大于两周，即平均出块实际大于原先的 10 min，这时应把 D（难度值）调低，T 变大，使得区块更容易产生，从而保证区块产生的稳定性。

3.1.2　PoS

PoS（Proof of Stake）共识算法，也可以称为权益证明[19]。本算法的主要目的是解决 PoW 算法消耗大量计算能力造成浪费的问题。2011 年，一位名为 Quantum Mechanic 的网友在 Bitcointalk 论坛上首次提出 PoS 算法。

第一个 PoS 项目是 2012 年发布的点点币[19]（Peercoin）。点点币中引入了“币龄”这个概念，即持币数量乘以天数所得出来的结果。具体公式如下：

$$CoinAge=CoinNum\times CoinDay \tag{4}$$

其中，$CoinAge$ 为币龄；$CoinNum$ 为该节点所拥有的币数；$CoinDay$ 为持有币的天数。早在 2010 年，中本聪就在 BTC 设计中提出并使用了币龄[19]这一概念，用于确定交易的优先级，但这个概念在其安全模式中没有起到很重要的作用。币龄只是被简单地定义为货币的持有时间段。点点币中，节点每发现一个区块，币龄就会清零，每清 365 个币龄，就可以从区块中获得 0.05 个币，也就是年利率为 5%。例如：一个节点持有 10 个点点币，持有天数是 30 天，那么这个节点的币龄为 10×30=300，后来这个节点发现一个 PoS 区块，那么币龄将清零，这个节点从区块中获得的利息为 300÷365×0.05=0.04。节点想在点点币上产生区块就必须投入一定量的代币，就如公司入股一样，每过一段时间会产生一定量的利息。点点币系统中节点币龄越大，就越容易产生区块。具体目标公式如下：

$$Hash(n\ StakeModifier+tx\ Prev.block.n\ Time+tx\ Prev.offset+tx\ Prev.n\ Time+tx\ Prev.vout.n+n\ Time)<bn\ Target\times bn\ CoinDayWeight \tag{5}$$

其中，$Hash$ 为哈希函数中的哈希算法。$n\ StakeModifier$ 为权重修正因子。$tx\ Prev.block.n\ Time$，$tx\ Prev.offset$，$tx\ Prev.nTime$，$tx\ Prev.vout.n$ 为点点币中区块头中的部分信息属性。$n\ Time$ 为当前区块的时间戳。$bn\ Target$ 为目标阈值。$bn\ CoinDayWeight$ 为节点所拥有的币龄。从上面的公式可以看出，$bn\ CoinDayWeight$ 越大，节点就越容易达到目标不等式的要求，越容易获得记账权。

点点币 PoS 算法中目标阈值和难度值与 PoW 算法一样是成反比的。目标阈值越小，难度值越大。具体公式如下：

$$bn\ Target=pre\ bn\ Target\times(1\,007\times10\times60+2\times pre\ bn\ Target)\div(1\,009\times10\times60) \tag{6}$$

其中，$bn\ Target$ 为当前区块目标值，$pre\ bn\ Target$ 为前一个区块目标值，$2\times pre\ bn\ Target$ 为前两个区块时间间隔。点点币中一个区块的生成时间大概是 10 min 一周产生 1 008 个区块，故 $1\,007\times10\times60$ 表示 1 007 个区块产生的理论时间，$1\,009\times10\times60$ 表示 1 009 个区块产生的理论时间。如果前两个区块时间的时间间隔大于 10 min，则 $1\,007\times10\times60+2\times pre\ bn\ Target$ 的值将会大于 $1\,009\times10\times60$，故当前区块的目标值也会随之提高，从而降低了拥有当前区块的难度值。反之，如果前两个区块时间间隔小于 10 min，则 $1\,007\times10\times60+2\times pre\ bn\ Target$ 的值将会小于 $1\,009\times10\times60$，那么当前区块的目标值会随之降低，从而提高当前区块的难度值。

目前的 PoS 协议存在一些潜在的安全问题：币龄会被恶意的节点滥用以获得更高的网络权重并成功实施双花攻击。值得注意的是，由于币龄的问题，创世的节点可以通过定期开启钱包进行权益累积而滥用这一系统。故为了加强 PoS 算法的安全性，基于 PoS1.0 协议，2014 年 rat4（Pavel Vasin）提出了 PoS2.0 协议，并发布了黑币[20]。黑币中前 5000 个块是使用 PoW 算法产出的，目的是为了解决黑币分配不公平的问题。第 5001 个块到第 10000 个块是 PoW 算法和 PoS 算法共同产生的。从第 10 001 个块开始就是采用纯 PoS 算法。新的 PoS2.0 协议年利率为 1%，且目标公式中把币龄从等式

中去除。黑币中新的 PoS 算法证明公式为：

$$Hash(n\ StakeModifier + tx\ Prev.block.n\ Time + tx\ Prev.n\ Time + tx\ Prev.vout.hash + tx\ Prev.vout.n + nTime) < bn\ Target \times n\ Weight \quad (7)$$

其中，$n\ StakeModifier$ 为权重修正因子，在 PoS2.0 中用于预防计算攻击每次都会变化。$tx\ Prev.block.n\ Time$，$tx\ Prev.nTime$，$tx\ Prev.vout.hash$，$tx\ Prev.vout.n$ 为黑币中区块头的信息属性。$n\ Time$ 为当前区块的时间戳信息。$bn\ Target$ 为目标阈值。$n\ Weight$ 为节点拥有的币数。从式(7)中可以看出，黑币 PoS2.0 中的合格区块已与币龄无关，但和币数有关联。

3.1.3 DPoS

DPoS(Delegated proof of Stake) 共识算法，也可以称为委托股权证明[21]。DPoS 共识算法由 Bitshares 开发者 Dan Larimer 于 2014 年提出并应用，是 PoS 共识算法的一种衍生算法。DPoS 算法中的节点分为两大角色，分别为普通节点和委托人。委托人必须由持币节点进行投票，被选中的委托人负责管理区块链项目。某种程度上类似于“人民代表”制度。

其中，任何的持币节点都可以参与到投票和竞选委托人的环节中，在这些环节中节点可以随时进行投票、撤票等操作，且节点的持币量与投票的权重成正比。普通持币节点可以投票选出 n 个委托人。

n 个委托人之间没有什么差别，各自的权益都是相等的。之后系统会把委托人之间的顺序打乱排列，从而在重新排列的顺序上依次进行出块任务。委托人的职责主要是在自己的出块时间内收集各个节点间的交易并且验证交易的有效性，从而把交易打包进区块，同时把区块广播出去。委托人若在不是自己的出块时间出块，则所产生的区块会被其他节点视为无效块。

在完成一轮的委托人出块后，系统会把没有履行好职责的委托人剔除，之后重新对各个节点统计票数，再选出 n 个委托人，再一次地打乱委托人之间的顺序。从已有的结果表明，打乱顺序而不按照原有的顺序进行出块任务是为了防止相邻委托人因长时间顺序不变而结识，从而相互串通进行分叉攻击；其次，各个委托人会按照排好的顺序依次出块，在成功出块后委托人会获得一定的奖励。

3.2 联盟链

联盟链(Consortium Blockchain)不同于公有链，它由一个中心控制。联盟链也不同于私有链，它由一个机构或者个人控制。联盟链介于这两者之间，往往是由若干组织共同控制和维护，通常适用范围是一个联盟或者一种行业。这些组织结合共识算法对整个联盟进行管理，联盟链可以视为“部分去中心化”，一般被运用于需要相互沟通和交流的单位组织中。联盟链中常见的共识算法有 PBFT 共识算法、SBFT 共识算法、DBFT 共识算法等。

3.2.1 PBFT

PBFT(Practical Byzantine Fault Tolerance) 共识算法，也可以称为实用拜占庭容错[22]。PBFT 共识算法是基于 BFT(Byzantine Fault Tolerance，拜占庭容错)技术[23]所提出的一个具体的算法。BFT 技术就是为了解决在分布式系统中有部分节点由于一些不可预知的错误而宕机或作恶，其他节点如何通讯达成一致性的问题。PBFT 算法由 Miguel Castro 和 Barbara Liskov 于 1999 年提出，解决了原先拜占庭容错效率不高、复杂度为指数级别等问题。

PBFT 算法中允许容忍的异常节点最大个数为 f，那么一个节点可以收到的消息量为 $N-f$ 条消息，消息中又会有作恶的节点发送错误的信息，最多为 f 条，诚实的节点个数又要大于所容忍的节点个数，故 PBFT 算法中总节点的个数要满足 $N-f-f>f$ 这个不等式，即 $N\geqslant 3f+1$。公式表明若出现 1 个异常的节点，要想确保其余节点消息的传达能达到一致性需求需要总节点的个数至少为 4 个，且其余 3 个节点必须为诚实可靠的节点。

PBFT 算法的共识流程主要是客户端发送请求给主节点，之后主节点广播给其他节点开始三阶段的共识流程，各个节点收到的信息达到一定数量后回传给客户端，客户端收到回复确认共识完成。具体如图 2 所示。

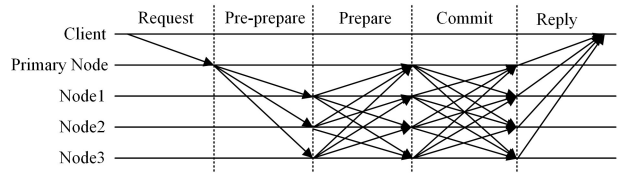

图 2 PBFT 算法共识过程

Fig. 2 Consensus process of PBFT algorithm

其中 Cline 为客户端；Primary Node 为主节点；Node1，Node2，Node3 为节点 1，2，3。

首先在 Request 阶段，客户端发消息给主节点。主节点验证请求消息并负责签名加密，之后封装 Pre-prepare 消息$\langle\langle Pre_prepare, v, n, d\rangle s, m\rangle$，该消息中 v 为视图编号，一轮共识一般都是在同一个视图下进行；n 代表序号；d 为消息摘要；s 为主节点对消息的签名；m 为原始消息数据。此时系统进入 Pre-prepare 阶段，主节点把 Pre-prepare 消息广播给其他节点，其他节点收到主节点发送过来的 Pre-prepare 消息后进行以下操作：

(1)首先对 m 验证签名的合法性；

(2)验证节点中的视图编号 v 是否与消息中的视图编号 v 相同；

(3)验证节点在当前视图上没有收到额外的 Pre-prepare 消息，即不存在另一个 $d'=hash(m')$；

(4)Pre-prepare 消息中的 n 满足不等式 $h\leqslant n\leqslant H$，h 和 H 代表序号 n 的高低阈值。

若验证通过，节点会封装对应的 Prepare 消息$\langle Prepare, v, n, d, i, s\rangle$，其中 i 表示节点的身份。之后系统进入到 Prepare 阶段，节点在 Prepare 阶段把 Prepare 消息广播给其余的节点。其余的节点在收到 Prepare 消息后会对其中的 v，n，d 进行验证。等节点收到 Prepare 消息数量为 $2f+1$(包括本身消息)且验证通过时，该节点开始封装 Commit 消息$\langle Commit, v, n, d, i\rangle$。系统进入到 Commit 阶段，节点开始广播 Commit 消息给其余的节点。其余的节点在收到 Commit 消息后，验证其中的 v，n，d 是否与本身的 v，n，d 无差别。若验证通过且通过数量为 $2f+1$(包括本身消息)时，系统会进入到 Reply 阶段，各个节点会将 Commit 消息回传给客户端。当客户端收到 $f+1$ 个相同的 Commit 消息时，这表明各个节点达成了共识，之后客户端会把消息存入本地状态数据库中。以上就是一次共识的完成过程。

此外，PBFT 算法中有视图切换机制，当节点发现主节点在一定的时间里没有完成共识，发生了宕机或主节点作恶，则启动视图切换机制。视图切换的公式为：$P=v \bmod |N|$，P 为主节点编号。每完成一轮共识，也会触发视图切换机制。其中需要注意的是，已经达成的共识不会在新的视图中进行回滚。

3.2.2 SBFT

SBFT 共识算法[24]是在 PBFT 算法的基础上进行扩展改进的，是一种可扩展和分散式信任模型基础设施算法，由 Gueta 等于 2018 年提出。

SBFT 算法在 PBFT 算法的基础上添加了 C-Collector 和 E-Collector 两个角色，它们都由副本节点担任（一个角色可由多个副本节点担任），这两个角色的作用都是收集并组合阈值签名并且转发相关结果给其他的节点，从而降低消息复杂度。

值得注意的是，SBFT 算法中有两条路径在不同的情况下促使节点达成共识，分别为快速路径（Fast path protocol）和线性 PBFT 路径（Linear-PBFT）。默认情况下，SBFT 算法执行快速路径。当快速路径无法达成共识，SBFT 算法就会运行线性 PBFT 路径。

快速路径中，SBFT 算法的共识过程分为 5 个阶段：Pre-prepare（预准备阶段）、Sign-share（签名分享阶段）、Commit-Proof（提交证明阶段）、Sign-state（签名状态阶段）、Execute-Proof（执行证明阶段），具体如图 3 所示。

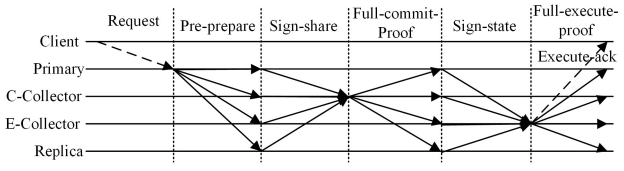

图 3 SBFT 算法快速路径共识过程

Fig. 3 Fast path consensus process of SBFT algorithm

客户端发送〈“*request*”，o，t，k〉消息给主节点，主节点再转发消息给其他副本节点。那么主节点进入到 Pre-prepare 阶段，主节点收到客户端发来的 Request 请求，随后创建一个〈“*pre-prepare*”，s，v，r〉消息决策块并将该块作为预准备消息广播给其他副本节点。其中 s 是当前序号，v 是视图编号，r 是 request 请求集合。Sing-share 阶段，副本节点收到预准备消息后，对决策块进行校验，校验通过的条件为：

（1）决策块中的视图编号与本节点的视图编号相同；

（2）视图 v 之前没有接受序列 s 的“pre-prepare”；

（3）序号 s 应满足公式 $ls<s<$win，其中 ls 为最后的稳定序列号，win 为一个常量用来限制未完成的块数；

（4）r 是通过认证和访问控制要求的一系列有效操作。

校验通过后副本节点会对相关信息进行哈希加密，公式为：

$$h=H(s \parallel v \parallel r) \tag{8}$$

其中，H 是加密哈希函数（SHA256）。接着对 h 进行阈值签名操作从而得到$\sigma_i(h)$，并向 C-Collector 节点的 *C-collectors*(s,v)集合发送〈“*sign-share*”，s，v，$\sigma_i(h)$〉签名消息。在 commit-Proof 阶段，每个 C-Collector 收到消息后，进行校验，校验条件为：

（1）C-Collector 节点的视图编号 v 与消息中的视图编号 v 相同；

（2）在同一个视图中，C-Collector 节点先前没有接收相同序列 s 的“*sign-share*”；

（3）阈值签名$\sigma_i(h)$通过 C-Collector 节点的验证。

当 C-Collector 节点接受到 $3f+c+1$ 条不同的签名消息且通过时，C-Collector 节点会组合签名$\sigma_i(h)$，并且广播〈“*full-commit-proof*”，s，v，$\sigma_i(h)$〉消息给所有节点。Sign-state 阶段，副本节点收到“*full-commit-proof*”消息后依旧对相关的参数进行校验，校验通过后提交 r（r 为序号为 s 的请求）。当序号 s 之前的所有请求都被执行，并且 r 是序列 s 的提交请求块时，副本节点通过执行请求 r 使状态D_{S-1}更新为 D_S。随后副本节点更新状态摘要为 $d=digest(D_S)$，签名为 $\pi_i(d)$，并向 E-Collector 节点的 *E-collectors*(s)集合发送〈“*sign-state*”，s，$\pi_i(d)$〉签名消息。Execute-proof 阶段，E-Collector 节点收到 *sign-state* 消息并验证签名，当收到 *sign-state* 消息数达到 $f+1$ 条时，E-Collector 节点会将消息组成一个单一的签名 $\pi(d)$，并且发送〈“*full-execute-proof*”，s，$\pi(d)$〉消息给所有节点表明决策块状态是持久的。E-Collector 节点会给客户端发送〈“*execute-ack*”，s，l，val，o，$\pi(d)$，$proof(o,l,s,D,val)$〉消息，其中 o 为请求（$o\in r$），l 为请求 o 的位置，val 为请求 o 的状态响应值，$proof(o,l,s,D,val)$为请求 o 是否被执行的证明。客户端在接收到 *execute-ack* 消息后会校验 $\pi(d)$是否有效和 $proof(o,l,s,D,val)$是否为真，校验通过后客户端标记请求 o 为已执行和设置 val 作为其返回值。SBFT 共识过程中客户端只需要接收确认一条消息就可以完成共识。

线性 PBFT 路径的触发条件是在规定的时间内客户端没有接收到 Execute-act 消息（即快速路径无法达成共识）。则客户端会把 Request 请求发送给所有的节点，并请求一个新的线性 PBFT 路径。线性 PBFT 路径中的 Sign-state 阶段包含两个签名，分别为速路径所需要的$\sigma_i(h)$和线性 PBFT 路径所需要的$\tau_i(h)$。当一定的时间内，C-Collector 节点只收集到足够创建签名$\tau_i(h)$的消息，而没有收集到足够创建签名$\sigma_i(h)$的消息，则 C-Collector 节点发送〈“*prepare*”，s，v，$\tau(h)$〉消息给其他节点，从而进入到 Prepare 阶段。在该阶段中副本节点收到 *prepare* 消息后进行验证，通过后发送〈“*commit*”，s，v，$\tau_i(\tau(h))$〉给全部的 Collector 节点。随后在 PBFT commit-proof 阶段，C-Collector 节点（包含主节点）收集到一定数量的签名后发送〈“*full-commit-proof-slow*”s，v，$\tau(\tau(h))$〉消息给所有节点。如果节点已收到过 *pre-prepare* 消息和 *full-commit-proof-slow* 消息，并且通过 $h=H(s \parallel v \parallel r)$验证，随后节点就提交 r（序列为 s 的决策块），其他节点接收到 r 进而触发快速路径中 Sign-state 阶段和 Execute-proof 阶段，从而使节点达成共识。

3.2.3 T-PBFT

T-PBFT 共识算法[25]（EigenTrust-Based Practical Byzantine Fault Tolerance）是一种基于特征信任模型的优化实用拜占庭容错共识算法，也是一种多阶段共识算法，通过节点间的交易来评估节点信任，从而选择网络中质量较高的节点来构建共识群。T-PBFT 算法由 Gao 等于 2019 年提出。

T-PBFT 算法引入 EigenTrust 信任模型[26]，根据计算的全局信任值，把信任值高的节点挑选出来组成一个共识组，以此作为节点达成共识的基础。全局信任值的公式为：

$$T_i = C_{1i}T_1 + \cdots + C_{ni}T_n \tag{9}$$

其中，T_i 是 $node_i$ 的全局信任值，C_{ij} 是 $node_i$ 对 $node_j$ 的局部信任值（通过 $node_i$ 对 $node_j$ 两个节点的交易满意数量来计算）。局部信任值分为两种，分别为直接信任值（两个节点有直接交易）和推荐信任值（两个节点没有直接交易）。由于共识组的存在，参与共识的节点数量减少，从而使得共识在大规模的网络环境下的共识过程更加高效。随着区块上链，节点间会产生新的交易，全局信任值随之改变，共识组的节点也因全局信任值而动态改变。最后，T-PBFT 可以开始新的一轮共识。T-PBFT 共识算法达成共识需要满足两个前提条件，分别为：1）行为一致性，即在本算法中，那些全局信任值高的节点值得信任，并以高概率诚实行事；2）有限的交易时间，以在一定时间内的交易量为基础，计算节点的全局信任值。从而保证 EigenTrus 的有效性和交易集的稳定性。

T-PBFT 算法的共识过程分为 4 个阶段：Group Process 阶段（共识组阶段）、Pre-Prepare 阶段（预准备阶段）、Prepare 阶段（准备阶段）、Reply 阶段（回复阶段）。具体如图 4 所示。

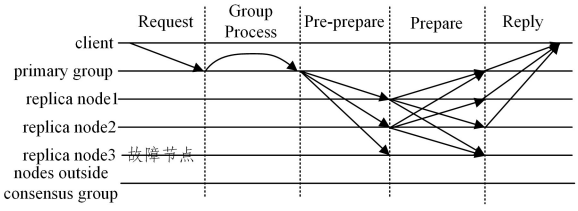

图 4 T-PBFT 算法共识过程

Fig. 4 Consensus process of T-PBFT algorithm

Group Process 阶段，主组中的相关节点收到客户端发来的消息后，将消息打包且放到预生成的块中，随后广播给主组中的其他成员。其他主组成员收到预生成块后进行验证，验证通过后每个主组成员节点将用相同的视图临时记录预生成块。其中，若主组中的一个节点发生故障，也可以立即替换他，并不会触发视图更改过程。同时进入到 Pre-Prepare 阶段，主组将预生成块和组签名的预准备消息广播给共识组的副本节点进行审核和验证。其中节点可以验证主组签名的有效性，但无法发现是哪个主组成员签了该名。在 Prepare 阶段，副本节点验证预生成块的有效性，每个副本节点在预生成块中使用预先安排的交易顺序模拟打包事务执行，同时计算块哈希，如果块哈希值与当前块哈希值一致，则通过有效性检查。随后副本节点向其他节点广播带有它们签名的准备信息。一旦有节点收到的消息数量超过 $2f$（其中 f 是共识组中的拜占庭数量），则该节点会发送一条回复给客户端。之后共识到达 Reply 阶段，客户端在收到相同应答消息数为 $f+1$ 时，确认预生成块，并将块添加到区块链的尾部。随后每个节点更新它们的日志记录，从而达成共识。

3.2.4 MBFT

MBFT 共识算法（mixed Byzantine fault tolerance）[27] 是一种结合分层技术和分片技术的共识算法。分层可以有效减少单个节点的负载，提高一致性效率。分片可以将事务分配给不同的节点组，从而提高算法的处理能力，降低延迟。同时，本算法还给出了一种基于 VR 和阈值共享的方法来确保系统的错误率。算法设置了一个信用评分系统来提高节点打包交易的积极性。MBFT 算法由 Du 等于 2020 年提出。

在 MBFT 算法中，算法将节点分为 3 组，分别为：备份节点、低水平共识组（LCG）和高水平共识组（HCG）。组里的成员每个周期（5000 个块为一个周期）都要进行重新任选（在上一个周期末进行选举）。选举分为 3 个时期，分别为承诺期（前 x 个区块）、披露期（后 100 个区块）、额外披露期（最后 100 个区块）。所有想要在下一个周期成为验证节点的节点都需要生成一个随机数并计算一个哈希值。在披露期内，所有的共识节点都需要披露承诺期内公布的随机数。在额外披露期间，所有验证节点检索交易，以识别披露期间未公布的随机数。所有节点都可以获得参与选举节点的公钥和随机数。随机选择算法满足 3 个要求：

（1）在承诺期内，一个节点的随机数被划分为多个子随机数，并由其他节点保存。

（2）在承诺期内，任何少于 $f+1$ 个子随机数的节点都不能提前恢复随机数。

（3）在额外的披露周期内，一个节点可以使用一定数量的子随机数来恢复原来的随机数。

节点得到公钥和随机数后可以根据相关公式计算出候选节点和随机数之间的距离。节点选择距离最小的相同验证节点，再根据每个共识组中的节点数量和分配规则，依次将验证节点分配给高层共识组和低层共识组，其他候选节点则是备份节点。

MBFT 算法中还存在信用机制，以此来激励节点维护共识，信用评分公式为：

$$W = \frac{K w_0 \, e^{r_1 r_2^{\varepsilon}(t_C - 200 - b^{\varepsilon})}}{K + w_0 (e^{r_1 r_2^{\varepsilon}(t_C - 200 - b^{\varepsilon})} - 1)} \tag{10}$$

其中，K 为函数的最大值，w_0 为节点加入系统时的初始值，r_1 为生长系数，r_2 为惩罚系数，ε 为节点故障个数，t_C 为验证节点参与的周期数，b 为回归系数。该信用评分公式是一个 S 型函数，当节点参与共识时，节点的信用分数会缓慢增长。信用分数达到一定的数值后，会趋于不变，从而防止“超级节点”的产生。当发现某些节点作恶时，作恶节点的信用分数以指数级别降低。从而系统可以快速地剔除作恶节点，增加作恶成本，减少损失。

MBFT 将验证节点分为 LCG 和 HCG 两层，LCG 达成共识后会发送一个 mini-block 给 HCG。HCG 验证通过且达成共识后，会生成一个 large-block 添加到区块链中。具体共识流程如下。

LCG 共识中，客户端向备份节点发起一个事务请求 $\langle REQUEST, tx, t_c \rangle_{\sigma_c}$ 给备选节点，其中 tx 是请求执行的事务，t_c 是发起事务的客户端的时间戳，σ_c 表示客户端 c 对请求进行签名。备份节点接收到客户端的请求后，验证客户端的身份和区块链上的时间戳。如果认证成功，并且事务请求的时间戳和最近的块之间的时间差 Δt 小于预定义的时间，则备份节点会生成新的请求消息 $\langle EQUEST, tx, t_c \rangle_{\sigma_{bi}}$，并将其发送到由事务分配规则（事务分配规则是对每个事务的每笔交易的第一个输入进行模运算，运算结果作为 LCG 号）确定的低层次共识组中的节点。其中 σ_{bi} 备份节点对客户端请求的签名。LCG 中的节点将请求转发给组中的主节点。主节点需要验证事务，它确认备份节点的签名是否正确以及交易是否与交易等待池 $Epool_p$ 或交易打包池 $Ppool_{bi}$ 中或已记录在区块链中的其他交易冲突。如果验证成功，交易将编号和签名为 $\langle TRANSACTION, \langle REQUEST, tx, t_c \rangle_{\sigma_{bi}}, cycle, m \rangle_{\sigma_p}$ 的

消息广播给提议共识组中的所有节点，其中 $cycle$ 是 LCG 的当前周期，m 是来自当前主节点的事务数，σ_p 是主节点 p 的签名。当 LCG 中的节点接收到交易消息时，它首先验证交易的数字和签名。节点确认信息后，将交易添加到本地交易等待池 $Epool_{tni}$，并发送 $\langle AGREE, cycle, m\rangle_{\sigma_{tni}}$ 给共识组中的主节点。与此同时，主节点存储 AGREE 消息。如果主节点接收到某个事务的 $2f+1$ 条 AGREE 消息，事务被移出本地交易等待池 $Epool_p$，并且这条交易连同收到的 $2f+1$ 条 AGREE 消息会被放入交易打包池 $Ppool_p$ 中。预打包时间到后，共识组的主节点 p 将交易打包池 $Ppool_{tni}$ 中的交易打包，并将区块信息 $\langle MINIBLOCK, height, mini\_block_{lcg}\rangle_{\sigma_p}$ 发送给同一个共识组的其他节点，其中 $height$ 是当前区块的高度，$mini\_block_{lcg}$ 是节点已经验证并打包的区块。同一共识组中的节点收到主节点打包的区块信息后，对区块信息和区块中包含的每笔交易进行验证。节点需要确认 $MINIBLOCK$ 中的事务是正确的或事务在 $Epool_{tni}$ 中。验证后，节点广播 $\langle AGREE\_BLOCK, HASH(mini\_block_{lcg}), height\rangle_{\sigma_i}$，将信息发送到同一 LCG 中的其他节点。当主节点接收到足够的 AGREE_BLOCK 信息时，可以将 $\langle AGREE\_BLOCK_{LCG}, mini\_block_{lcg}, [sig_i(mini\_block_{lcg})], height\rangle_{\sigma_p}$ 发送给高层共识组，其中 $block_{lcg}$ 是 LCG 对当前区块高度进行共识后得到的区块，$[sig_i(mini\_block_{lcg})]$ 是 LCG 中签名的集合，这些签名来自 AGREE_BLOCK。交易可以在发送小区块后继续验证并推送到 $Ppool_{tni}$，因为 LCG 只需要在大区块上链后保持最终的一致性。因此，共识组中的每个验证节点都可能在打包一个小块后，在 $Epool$ 或 $Ppool$ 中进行交易。如果满足某些条件，这些交易可以在新一轮的共识中优先确认。因此，在所有节点收到大块后，对大块进行验证，并根据大块检索本地 $Epool_i$ 和 $Ppool_i$ 中的交易，并删除已经打包到大块中的交易。在新一轮的共识中，节点将对本地池中的现有交易进行寻址，并将满足交易等待时间要求但在本轮未打包入区块的 $Epool_i$ 中的交易发送到主节点共识组。

HCG 共识中，节点接收到 LCGs 发送的 mini-block 后，高层共识组中的节点需要进行检查，判断 mini-block：1）每个小块中的签名是否足够；2）验证节点的所有签名是否正确；3）前一个块的哈希点是否指向当前的大块；4）所有交易的输入是否冲突。在此基础之上，LCG 中的节点确认消息 $\langle RECEIVE_{BLOCK}, HASH(mini\_blcok_{lcg}), height\rangle_{\sigma_i}$，且 LCG 中的节点负责确认 mini-block 被 HCG 中的大多数节点（$2f+1$）接收。如果 HCG 中的主节点在最大打包时间内收集了所有的 mini-block 或超过 3/4 个 mini-block，广播 mini-block 的序列号确认信息 $\langle BLOCK, HASH(mini\_block_{lcg}), height\rangle_{\sigma_i}$，并从 HCG 中的节点收集共识信息，其中，$height$ 是当前大块的数量，σ 是主节点的签名。HCG 的节点验证主节点发出的这些序列是否合法。如果节点与主节点发出的序列一致，则节点返回 AGREE 消息。如果本地缺少某些小区块 $mini\_block_{lcg}$，则节点向对应节点请求区块信息，验证成功后返回 $\langle AGREE, [HASH(mini_{block\ lcg})], height\rangle_{\sigma_i}$。如果验证最终失败，则广播 $\langle DISAGREE, [HASH(mini\_block_{lcg})]\rangle_{\sigma_i}$。如果任一节点收集到 $2f+1$ 个 DISAGREE 消息，则可以广播 $[\langle DISAGREE\rangle]$ 集并进入下一个出块阶段。

3.3 私有链

私有链一般来说是由某个组织来进行管理的区块链，这个组织可以对任何信息进行查看和调用，以此来改善数据处理能力和处理遇到的不同问题。在私有链中节点一般都是可以信任的，节点出块也没有相关的政策激励。正因如此私有链共识算法一般来说交易吞吐量会较高，且资源消耗成本也相对较低，私有链一般运用在个人或单位组织等可控且信息保密的情况下。其中 Paxos 和 Raft 算法是现今主流的私有链共识算法。

3.3.1 Paxos

Paxos 共识算法[28]是由 Lamport 于 1989 年提出的一致性算法，该名字取自作者故事中希腊岛屿的名称 Paxos。Paxos 算法中节点的角色有 3 种，分别为 Proposers（提议者）、Acceptors（接收者）和 Learners（学习者）。共识过程中一个节点可以有多个角色身份，多个角色之间负责发送或者接收消息就某个值达成一致性。

Proposers：提议者，往往对 Acceptors 发送 Proposal（提案），类似于客户端这类角色。

Acceptors：接收者，根据自己已拥有的相关信息对 Proposal 进行裁决。

Learners：同步者，负责对最终的提案进行同步。

Proposal：提案，一个提案是由 Proposal number 和 value 组成，Proposal number 被 Acceptors 用来裁决是否需要接收该 Proposal。value 就是 Proposal 本身的内容。

Paxos 算法中是不考虑拜占庭问题的，即消息不会因节点而篡改。除此之外 Paxos 算法还需满足 safaty 要求，safaty 要求为：

（1）只有被 Proposers 提出的 Proposal 中的 value 才能够被选定。

（2）Acceptors 只能选择一个 Proposal 中的 value 值。

（3）进程永远不会知道哪个 value 被选定，除非 value 值是真正被一半以上的 Acceptors 认可。

Paxos 算法中没有明确的 liveness 要求，然而算法目标是确保最终有 Proposal 中的 value 值被选定，且 Learners 进行了同步操作。

Paxos 算法具体的流程图如图 5 所示。

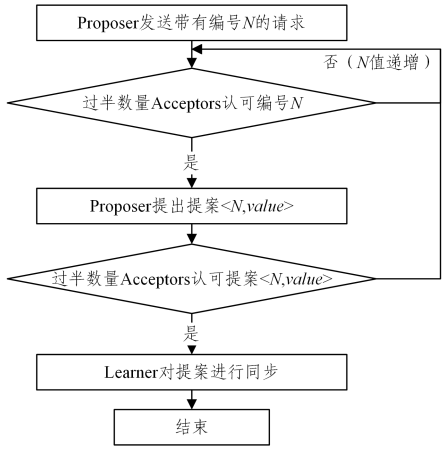

图 5 Paxos 算法流程图

Fig. 5 Flow chart of Paxos algorithm

将 Proposers 和 Acceptors 进行的流程放在一起，可以分为 prepare 阶段和 accept 阶段。

prepare 阶段，Proposers 会发送 Proposal 给一半数量以上的 Acceptors，其中 Proposal number 编号为 N。Acceptors

在接收到 Proposal 后会先把自己已接收的 Proposal number 与 N 比较，分为两种情况：第一种情况是，如果 N 小于已接收的 Proposal number，则 Acceptors 回应 Proposers 内容为拒绝接收和已接收的最大 Proposal number；第二种情况是，若 N 大于或等于已接收的 Proposal number，则 Acceptors 会把 N 保存下来，保证不再接收 Proposal number 小于 N 的编号，且 Acceptors 回应 Proposers 的内容为之前接收过的最大 Proposal number 的 Proposal，如果 Acceptors 本身从来没有接收过 Proposal 则回应为空。

accept 阶段，若 Proposers 没有收到半数以上 Acceptor 的接收信号，则 Proposers 改变编号 N（N 值应为递增，且不能重复）。反之，Proposers 在收到半数以上的 Acceptor 回复的接收信号后，Proposers 会发送一个〈N, $value$〉提案给 Acceptors。其中 value 值为 Proposers 接收回复信号的 Proposal 中最大 Proposal number 对应的 value 值（如若接收到的 Proposal 都为空，则 value 由 Proposer 决定）。Acceptors 在接收到 Proposers 发送的〈N, $value$〉提案后会再一次将自己已接收的 Proposal number 与 N 比较，如果 N 小于已接收的 Proposal number，则 Acceptors 响应 Proposers 的内容为拒绝接收和已接收的最大 Proposal number。

若 N 大于或等于已接收的 Proposal number，则 Acceptors 接受提案〈N, $value$〉，并回应 Proposers 已接收提案。Proposers 收到半数以上的接收信息后将 Proposal 发送给所有的 Learners 进行同步，从而达成共识。反之 Proposers 改变编号 N 重新开始 prepare 阶段。

3.3.2 Raft

Raft 算法[29]是一种用于管理复制日志的共识算法，产生的结果相当于(multi-)Paxos。Raft 算法与 Paxos 一样高效，但结构不同，这使得 Raft 算法相对来说更容易被理解[30]。Raft 算法是由 Ongaro 和 Ousterhout 于 2014 年提出的一种算法。

Raft 算法把节点分为 3 种角色，分别为 leader(领导者)、follower(追随者)和 candidate(候选人)。系统在正常情况下，只有一个节点是 leader，其余节点是 follower。candidate 角色只有在节点竞争 leader 角色时存在。

leader(领导者)：负责接收客户端发来的信息，与 follower 保持联系，且与 follower 保持日志同步。

follower(追随者)：接收 leader 的信息并同步，响应 candidate 发来的投票请求。

candidate(候选者)：发起投票请求给节点，在选举超时或者心跳超时后有 follower 节点会成为 candidate 节点。

Term(任期)：一开始 Term 号为 0，Term 号是连续递增的，每当有新的选举时 Term 号会自增。

Raft 算法具体共识过程分为两个过程，即领导选举和日志复制。

领导选举：Raft 算法在领导选举这个过程中有两个超时来控制选举。首先是选举超时，每个节点都有一个选举时间(时间在 150 至 300 ms 之间，每个节点的选举时间都是随机设置的)。当 follower 节点启动后，follower 节点在选举时间的时间内没有接收到 leader 的心跳(heartbeats)信息或者其他 candidate 的投票请求，则当前这个 follower 节点改变角色为 candidate，Term 加 1，开始新的选举任期并为自己投票，同时为 follower 节点和 candidate 节点向其他角色发起投票请求。其他角色节点收到投票请求后会根据接收条件来判断是否接受投票请求。接受投票请求的条件为：

(1)节点尚未接受过其他节点的投票请求。

(2)节点的 Term 小于请求节点的 Term。

如果都符合，则节点会投票给候选人，其中其他 follower 节点在接受投票请求后会重置自身的选举时间(时间随机)，其他的 candidate 节点会变为 follower 角色且重置自己的选举时间。candidate 节点在收到至少 $N/2+1$ 票数后成为新的 leader。若有两个 follower 节点在选举超时后，同时成为 candidate 节点，则根据票数来决定谁成为 leader 节点。若两个 candidate 节点收到的票数一样，则所有节点一起等待某个节点(包括 follower 节点和 candidate 节点)选举超时成为 candidate 节点($Term+1$)发起投票。第二个控制选举的超时是心跳超时，每个 follower 节点在心跳时间会收到 leader 的心跳信息，否则为心跳超时，与此同时触发 follower 节点的选举超时开始选举新 Term 的 leader。

日志复制：首先客户端发送请求给 leader，leader 在接收到请求后，会将请求附加到自身的日志中(请求为未提交状态)，在下一次的心跳时间把请求发送给 follower。follower 在验证请求通过后会复制这条请求信息到自身的日志条目中(状态为未提交)，随之发送响应成功给 leader。leader 在收到 $N/2+1$ 条成功的响应后会将请求状态改为提交状态并且发送响应给客户端，且在下一间隔的心跳时间通知 follower 节点请求已提交。follower 节点收到通知后也会将请求信息状态改为提交状态。至此，日志复制完成，系统大部分节点达到一致性状态。由于在分布式系统中网络是分割的，有些节点会发生宕机行为，从而使得 leader 和 follower 的日志不同，因此 follower 在接收 leader 发送过来的请求后，会验证请求中的 Term 号是否与自己的 Term 号相同，不同则代表着日志不同步。若 leader 的 Term 号大于 follower 的 Term 号，为了使日志同步，leader 需寻找到最后与 follower 日志一样的 Term 号，与此同时 leader 从这个 Term 号开始发送自己的日志给 Follower，以此达到日志的同步。反之，follower 节点会拒绝 Term 号小的 leader 发来的请求。

4 共识算法的比较

不同的区块链共识算法适用不同的应用场景，因此不同的系统采用不同的共识算法也会产生不同的效果。故本文将从去中心化程度、可扩展性程度和安全性程度这 3 个方面对上面 9 种共识算法进行比较与总结。

4.1 评价体系

去中心化程度：去中心化并不代表没有中心节点，通常情况下主节点就是中心节点。去中心化表示的是区块链交易模式相对于传统交易模式下的去中心程度，是个相对的概念。其中去中心化程度是根据共识节点数量、主节点选择方式和共识节点权重等指标来评判去中心化程度的高低。

(1)共识节点数量：指在区块链网络中，参与共识算法的节点数量。共识节点数量也可以反映出网络的活跃程度。同时，共识节点数量也可以影响区块链网络的性能。如果共识节点数量过多，可能会导致网络拥堵，影响区块链的性能。

(2)主节点选择方式：指共识过程中主节点的获取方式是

通过竞争、选举、还是取模计算或信用评估等方式产生。

(3)共识节点权重：指参与共识的节点成为主节点的概率是否相同。

安全性程度：共识算法的安全性受到许多因素的影响，其中节点的容错性和攻击成本是安全性的重要影响因素。容错能力高，攻击成本大，往往会使得共识算法更加安全可靠。故在本次评价中，安全性程度是根据容错性的高低和攻击成本的大小来评估判断的。

(1)容错性：指共识算法在保证能达到共识的情况下，所能容纳最多的拜占庭节点数量。

(2)攻击成本：指恶意的节点在发动相关的攻击时，所需付出代价的大小。如分叉攻击[31]、女巫攻击[32]、双花攻击[33]、DDoS攻击[34]和日蚀攻击[35]等。

可扩展性程度：可以理解为算法处理高业务量的能力。算法处理业务的能力受资源消耗和通讯次数等因素的影响。算法资源消耗量的减少和通讯次数的减少，可以提高交易的吞吐量和减少共识的时延，即大幅度地提高了算法处理业务的能力，提高节点间的共识效率。故在本次评价中，可扩展性程度是根据资源消耗和通讯复杂度等指标来评判可扩展性程度的高低。

(1)资源消耗：指各个节点在达成共识的过程中对算力、储存、电力和网络带宽等资源的消耗。

(2)通信复杂度：指在分布式计算中，完成一个任务所需的最大通信量。它衡量了在分布式系统中，节点之间交换信息的成本。

4.2 对比情况

表2列出了9种算法在共识节点数量、主节点选择方式、共识算法权重、容错率、攻击成本、资源消耗和通信复杂度这7个方面对去中心化、安全性和可扩展性[36]总体的一个比较总结结果。

表2 9种共识算法比较

Table 2 Comparison of 9 consensus algorithms

算法分类	去中心化				安全性				可扩展性		
	共识节点数量	主节点选择方式	共识节点权重	去中心化程度	共识节点数量	主节点选择方式	共识节点权重	去中心化程度	资源消耗	通信复杂度	可扩展性程度
PoW	多	竞争	中	高	$n/2$	$n/2$	高	较高	高	$O(N)$	较低
PoS	多	竞争	较低	高	$n/2$	$n/2$	较高	较高	较高	$O(N)$	中
DPoS	较多	选举	高	较高	$n/2$	$n/2$	较低	中	低	$O(N)$	较高
PBFT	中	计算	高	较高	$n/3$	$n/3$	较低	中	中	$O(N^2)$	较低
SBFT	中	计算	高	较高	$n/3$	$n/3$	中	中	中	$O(N)$	中
T-PBFT	较少	评估	低	较低	$n/3$	$n/3$	较高	中	较低	$O(N^2)$	中
MBFT	中	计算	高	较高	$n/3$	$n/3$	中	中	较高	$O(N^2)$	较低
Paxos	少	选举	高	中	0	$n/2$	低	中	低	$O(N^2)$	较低
Raft	少	选举	高	中	0	$n/2$	低	中	低	$O(N)$	较高

去中心化程度是根据共识节点数量、主节点选择方式和共识节点权重的一个综合评估来确定的，是在参与共识节点的环境下进行评估的。评估分数总共10分：共识节点数量总分为4分，数量为多加3分，数量为较多加3分，数量为中加2分，数量为较少加1.5分，数量为少加1分；主节点选择方式总分为3分，方式为竞争和计算加3分，选举加2分，评估加1分；共识节点权重总分为3分，权重为高加3分，权重为中加2分，权重为较低加1.5分，权重为低加1分。具体评分细则如表3所列。

表3 9种共识算法去中心化评分细则

Table 3 Decentralized scoring rules of 9 kinds of consensus algorithm

算法	共识节点数量	主节点选择方式	共识节点权重	总分
PoW	4	3	2	9
PoS	4	3	1.5	8.5
DPoS	3	2	3	8
PBFT	2	3	3	8
SBFT	2	3	3	8
T-PBFT	1.5	1	1	3.5
MBFT	2	3	3	8
Paxos	1	2	3	6
Raft	1	2	3	6

由表3可得这9种共识算法的得分为：PoW(9分)＞PoS(8.5分)＞DPoS(8分)＝PBFT(8分)＝SBFT(8分)＝MBFT(8分)＞Paxos(6分)＝Raft(6分)＞T－PBFT(3.5分)。具体的去中心化对比情况如图6所示。

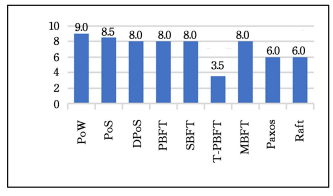

图6 9种算法去中心化程度对比

Fig. 6 Comparison of decentralization degree of 9 algorithms

安全性程度是根据容错率和攻击成本等其他因素来综合评估的。评分分数为10分：拜占庭容错总分为2分，$n/2$得2分，$n/3$得1分，0为0分；容错率总分为2分，$n/2$得2分，$n/3$得1分；攻击成本总分为4分，高得4分，较高得3分，中得2分，较低得1.5分，低得1分；其他因素总分为2分，因为安全性程度还受到其他因素影响，如联盟链中的节点相对于公有链更加可靠可信，故还需再加1分。Paxos和Raft算法通常是在私有链中使用，共识节点相对于联盟链中的共识节点又更加可控，故还需再加2分。具体评分细则如表4所列。

表4 9种共识算法安全性评分细则

Table 4 Security scoring rules of 9 kinds of consensus algorithm

算法	拜占庭容错	容错率	攻击成本	其他因素	总分
PoW	2	2	4.0	0	8.0
PoS	2	2	3.0	0	7.0
DPoS	2	2	1.5	0	5.5
PBFT	1	1	1.5	1.0	4.5
SBFT	1	1	2.0	1.0	5.0
T-PBFT	1	1	3.0	1.0	6.0
MBFT	1	1	2.0	1.0	5.0
Paxos	0	2	1.0	2.0	5.0
Raft	0	2	1.0	2.0	5.0

从表 4 可得安全性程度的比较顺序为：PoW(8 分)＞PoS(7 分)＞T－PBFT(6 分)＞DPoS(5.5 分)＞SBFT(5 分)＝MBFT(5 分)＝Paxos(5 分)＝Raft(5 分)＞PBFT(4.5 分)。具体安全性对比情况如图 7 所示。

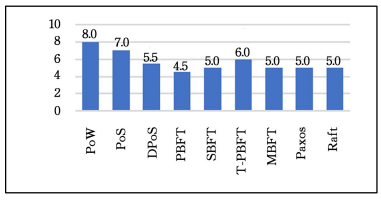

图 7 9 种共识算法安全性程度对比

Fig. 7 Comparison of security levels of 9 consensus algorithms

可扩展性是根据资源消耗和通信复杂度等其他因素综合评估确定。评分总分数为 10 分：资源消耗总分为 4 分，高得 1 分，较高得 2 分，中得 3 分，较低得 3.5 分，低得 4 分；通信复杂度总分为 3 分，$O(N)$得 3 分，为 $O(N^2)$得 1 分；其他因素总分为 3 分，因为 Paxos 算法在原理上很晦涩难懂，开发者们比较难在该算法中进行扩展改进，故该算法得分应当扣掉 3 分。具体评分细则如表 5 所列。

表 5 9 种共识算法可扩展性评分细则

Table 5 Scalability scoring rules of 9 kinds of consensus algorithm

算法	资源消耗	通信复杂度	其它因素	总分
PoW	1.0	3	0	4.0
PoS	2.0	3	0	5.0
DPoS	4.0	3	0	7.0
PBFT	3.0	1	0	4.0
SBFT	3.0	3	0	6.0
T-PBFT	3.5	1	0	4.5
MBFT	2.0	1	0	3.0
Paxos	4.0	1	－3	2.0
Raft	4.0	3	0	7.0

从表 5 可知，这 9 种算法的得分为：DPoS(7 分)＝Raft(7 分)＞SBFT(6 分)＞PoS(5 分)＞T－PBFT(4.5 分)＞PoW(4 分)＝PBFT(4 分)＞MBFT(3 分)＞Paxos(2 分)，具体可扩展性对比情况如图 8 所示。

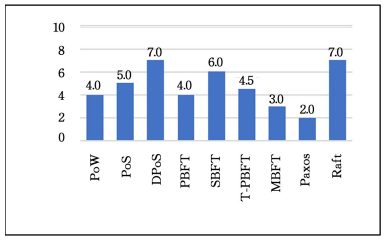

图 8 9 种共识算法可扩展性程度对比

Fig. 8 Comparison chart of scalability of 9 consensus algorithms

4.3 比较结论

PoW 算法让分布式系统不需要任何第三方的介入就可以确保各个节点达到一致性，且只要某个机构没有达到全网 51%算力就可以确保分布式系统的安全和稳定。正因为 PoW 算法过度依靠算力，导致大量的矿机和电力的投入，只为寻找到一个合适的 Nonce 值，若没有寻找到 Nonce 值，所有的花费就付诸东流，造成极大的浪费。随着矿机的出现，个人挖矿也越来越无力，许多机构和组织开始把算力聚集到一起，导致中心化程度加深，甚至有可能产生 51%的算力攻击。

PoS 算法是通过节点的币龄来争取记账权。PoS2.0 算法中去除了币龄这个概念，它通过节点的币数来争取记账权。上述分析表明 PoS 算法相比 PoW 算法更节省资源也更安全，受到 51%攻击的可能性极低。但正因 PoS 算法不需要算力，导致有些节点并不希望参与记账，但它们因为币龄而获得记账权，所以记账权没有专业化。随之而来的是拥有币数多的节点更容易得到记账权，容易形成记账权总被一些节点来回拥有的现象；也导致许多节点想着存币拿利息，使币的流动性降低。

DPoS 算法跟 PoW 算法以及 PoS 算法一样，遵守最长链是合法链这一原则，从而保证区块链的安全，防止不诚实节点发动的分叉攻击。除此之外，DPoS 算法进行的这些改进使得 DPoS 算法比 PoS 算法拥有更快的效率和更高的性能。然而 DPoS 算法为了追求性能对去中心化妥协，算法中委托人的性质就像 PoW 算法中的矿池，若某些 DPoS 项目中委托人数量少，就会导致区块链网络失去真正意义上的去中心化目的，不利于区块链的进一步发展。

PBFT 算法中无需消耗算力，也没有币的设定；其次，算法允许恶意节点的存在，也大大地缩短了共识时间，提高了共识效率。但算法的通信复杂度为 $O(n^2)$，当系统中节点的个数增多到一定数量时，分布式系统的性能下降也是指数级别的。该算法比较适合联盟链或者私有链。

SBFT 算法是由 PBFT 算法改进而来，在原本的基础上加入了 BLS 聚合签名算法，使通信复杂度从原来 PBFT 中的 $O(n^2)$降为本算法的 $O(n)$，对宽带要求减小，有更好的扩容性。SBFT 算法的性能优势随着客户端数量的增加而增加。但正因为 SBFT 存在 C-Collector 和 E-Collector 角色，使得在参与共识节点数量少的情况下，共识过程有中心化趋势。

T-PBFT 算法首先使用 EigenTrust 信任模型挑选节点来进行共识过程，使得参与共识的节点的稳定性和安全性是可控的。但正因为其根据信誉值来挑选节点，使得该算法越趋于中心化。如果节点不犯错，共识过程基本上就是由一些相同的节点长期控制着。同时，该算法如果参与共识的节点过多，也会像 PBFT 一样，共识效率随着共识节点增多而下降。

MBFT 算法通过分层和分片技术把节点分在不同层次和不同的共识组中。分层的优势是可以减少节点达成共识所需的负荷，分组的优势则是可以使得该算法能够处理更多事务。两者结合使得 MBFT 算法在共识节点多的情况下拥有更高的吞吐量。但在 LCG 中共识组的通信复杂度和 PBFT 算法一样，并没有得到改善。这使得 MBFT 算法中共识组的节点一旦过多，通信复杂度会呈指数级别增加，使得共识效率变低。且节点的分组和分层是通过计算而来，故算法的资源消耗会比 PBFT 高。

Paxos 算法中 3 类角色的分类和职责的不同使得该算法具有高度容错特性，让其成为目前解决分布式共识问题中最有效的算法之一。但 Paxos 算法会产生活锁的可能，在两个 Proposers 依次提出逐渐递增的 N，使得 Acceptors 会不断地认可编号大的提案，从而使 Paxos 算法无法就某个值达成一致。

Raft 算法的最大特点就是便于理解，它不像 Paxos 那样难懂。Raft 算法通过选举 leader 来实现消息达到一致性，使得 Raft 算法有强领导性。若 leader 节点发生故障，则需重新选举一个新的 leader，过程复杂，降低了共识效率，且 Raft

算法在节点作恶的情况下是毫无办法察觉和防范的。

9种算法优缺点的对比如表6所列。

表6　9种共识算法优缺点

Table 6　Advantages and disadvantages of 9 consensus algorithms

共识算法	优点	缺点
PoW	算法容错率高，安全性高	消耗大量资源，出块速度慢，吞吐量低
PoS	容错率高，相对于PoW算法来说更节省资源，出块速度略微提升	担任为主节点的节点可能不专业(没有当主节点的意愿)，还进行一些哈希计算，消耗资源，容易出现屯币现象造成币的流动性降低
DPoS	容错率高，出块速度快，解决了资源浪费的情况	委托人的出现使得该算法更趋于中心化，节点参与共识过程的积极性不高
PBFT	无需消耗算力，安全性高	节点间的通信开销太大，共识阶段太多，无法承受太多节点的参与
SBFT	节点间的通信开销低，安全性高，无需消耗算力	因C-Collector和E-Collector角色的存在，有中心化发展的趋势，若存在拜占庭节点，会极大地增加达成共识的时延
T-PBFT	安全性高，无需消耗算力	节点间通信开销大，节点变换少从而导致容易中心化
MBFT	无需消耗算力，安全性高，吞吐量高	资源消耗相对于PBFT高，比PBFT的开销大，节点间的通信开销太大，共识阶段太多，无法承受太多节点的参与
Paxos	出块速度快，性能高	算法基础原理晦涩难懂，不可容忍拜占庭节点的存在
Raft	便于理解，出块速度快，性能高	不可容忍拜占庭节点的存在

结束语　共识算法是区块链的核心之一，本文分别对PoW，PoS，DPoS，PBFT，SBFT，T-PBFT，MBFT，Paxos，Raft这9种共识算法进行了共识过程原理阐述，在此基础之上，也对这9种算法在去中心化、安全性和可扩展性这3个方面进行评估，最后进行了优缺点总结、分析和比较。希望可以为相关研究人员提供一些参考。

现如今，区块链成为了一个热门的研究方向，但要真正融入到人们的生活之中还需要走一段不同寻常的路。在现有的区块链共识算法的体系架构下，研究人员可以在资源优化、性能优化、修改共识选举机制、优化验证方式和能耗优化等方面优化共识算法[37]，也可以从多链共享和跨链共享这两个方面来提高区块链的吞吐量，优化共识算法的架构，从而促进区块链共识算法的应用和发展。

参 考 文 献

[1] NAKAMOTO S. Bitcoin: A Peer-to-Peer Electronic Cash System[EB/OL]. http://bitcoin.org/bitcoin.pdf.

[2] The global economy is in its weakest period in a decade, but bitcoin is quietly brewing the next bull market?[EB/OL]. https://baijiahao.baidu.com/s?id=1648083421346579310&wfr=spider&for=pc.

[3] SHAO Q F, JIN C Q, ZHANG Z, et al. Blockchain Technology: Architecture and progress[J]. Chinese Journal of Computers, 2018, 41(5): 969-988.

[4] ZHU L, YU H, ZHAN S X, et al. Research on high performance alliance blockchain Technology[J]. Journal of Software. 2019, 30(6): 17.

[5] CHEN W L, ZHENG Z B. Blockchain data analysis: status, trends and challenges[J]. Journal of Computer Research and Development, 2018, 55(9): 1853-1870.

[6] Blockchain for financial services[EB/OL]. https://www.ibm.com/blockchain/industries/financial-services.

[7] Ant chain[EB/OL]. https://antchain.antgroup.com/.

[8] Blockchain for Supply Chain-IBM Blockchain [EB/OL]. https://www.ibm.com/blockchain/supply-chain.

[9] Home-GBA Global[EB/OL]. https://www.gbaglobal.org/.

[10] State of the DApps－Ranking the Best Ethereum DApps[EB/OL]. https://www.stateofthedapps.com/zh/rankings/platform/ethereum.

[11] YUAN Y, WANG F. Blockchain: the state of the art and future trends[J]. Acta Automatica Sinica. 2016, 42(4): 481-494.

[12] OLSEN P, BORIT M. The components of a food traceability system[J]. Trends in Food Science & Technology. 2018, 77: 143-149.

[13] YUAN Y, WANG F. Blockchain and Cryptocurrencies: Model, Techniques, and Applications[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2018, 48(9): 1421-1428.

[14] LIU A D, DU X H, WANG N, et al Blockchain technology and its research progress in the field of information security[J]. Journal of Software. 2018, 29(7): 2092-2115.

[15] DWORK C, NAOR M. Pricing via Processing or Combatting Junk Mail[M]// Berlin: Springer, 2001: 139-147.

[16] BACK A. Hashcash-a denial of service counter-measure[EB/OL]. https://sites.cs.ucsb.edu/~rich/class/old.cs290/papers/hascash2.pdf.

[17] WANG M, DUAN M, ZHU J. Research on the Security Criteria of Hash Functions in the Blockchain [C]// Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts. ACM, 2018: 47-55.

[18] ZHENG W, ZHENG Z, CHEN X, et al. NutBaaS: A Blockchain-as-a-Service Platform [J]. IEEE Access. 2019, 7: 134422-134433.

[19] KING S, NADAL S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake[EB/OL]. https://bitcoin.peryaudo.org/vendor/peercoin-paper.pdf.

[20] VASIN P. Blackcoin's proof-of-stake protocol v2[J/OL]. https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf.

[21] YANG F, ZHOU W, WU Q, et al. Delegated Proof of Stake With Downgrade: A Secure and Efficient Blockchain Consensus Algorithm With Downgrade Mechanism[J]. IEEE Access. 2019, 7: 118541-118555.

[22] CASTRO M, LISKOV B. Practical byzantine fault tolerance and proactive recovery[J]. ACM transactions on computer systems. 2002, 20(4): 398-461.

[23] GRAMOLI V. From blockchain consensus back to Byzantine consensus[J]. Future Generation Computer Systems. 2020, 107: 760-769.

[24] GUETA G G, ABRAHAM I, GROSSMAN S, et al. Sbft: a scalable and decentralized trust infrastructure[C]// IEEE. 2019: 568-580.

[25] GAO S, YU T, ZHU J, et al. T-PBFT: An EigenTrust-based practical Byzantine fault tolerance consensus algorithm[J]. China Communications, 2019, 16(12): 111-123.

[26] KAMVAR S D, SCHLOSSER M T, GARCIA-MOLINA H. The Eigentrust algorithm for reputation management in P2P networks [C]// Proceedings of the 12th International Conference on World Wide Web. ACM. 2003: 640-651.

[27] DU M, CHEN Q, MA X. MBFT: A New Consensus Algorithm for Consortium Blockchain [J]. IEEE Access. 2020, 8: 87665-87675.

[28] LAMPORT L. Paxos made simple[J]. ACM Sigact News. 2001, 32(4): 18-25.

[29] ONGARO D, OUSTERHOUT J. In search of an understandable consensus algorithm (extended version) [C] // Proceeding of USENIX Annual Technical Conference. 2014: 19-20.

[30] WU Y, ZHONG S. Research on block chain consensus algorithm raft[J]. Netinfo Security. 2021, 21(6): 36-44.

[31] MCCORRY P, HEILMAN E, MILLER A. Atomically Trading with Roger: Gambling on the Success of a Hardfork[M]// Data Privacy Management, Cryptocurrencies and Blockchain Technology. Cham: Springer International Publishing, 2017: 334-353.

[32] DOUCEUR J R. The sybil attack[C]// Peer-to-Peer Systems. Berlin: Springer, 2002: 251-260.

[33] NAIR P R, DORAI D R. Evaluation of Performance and Security of Proof of Work and Proof of Stake using Blockchain [C]// 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV). IEEE. 2021: 279-283.

[34] DOULIGERIS C, MITROKOTSA A. DDoS attacks and defense mechanisms: classification and state-of-the-art [J]. Computer Networks, 2004, 44(5): 643-666.

[35] HEILMAN E, KENDLER A, ZOHAR A, et al. Eclipse attacks on bitcoin's peer-to-peer network[C]// 24th USENIX Security Symposium (USENIX Security'15). 2015: 129-144.

[36] ZHENZHEN J. Various possibilities around the "Impossible Triangle" of blockchain[EB/OL]. http://www.ccw.com.cn/channel/blockchain/2019-04-03/6967.html.

[37] ZHANG P Y, SONG J. Research Progress on efficiency optimization of blockchain consensus algorithm[J]. Computer Science, 2020, 47(12): 296-303.

TAN Pengliu, born in 1975, Ph.D, associate professor, is a member of China Computer Federation. His main research interests include blockchain, cyber-physical system, intelligent medical care, etc.