计算机科学
COMPUTER SCIENCE

区块链 BFT 共识算法研究进展

冯了了，丁滟，刘坤林，马科林，常俊胜

引用本文

冯了了，丁滟，刘坤林，马科林，常俊胜. 区块链 BFT 共识算法研究进展[J]. 计算机科学, 2022, 49(4): 329-339.

FENG Liao-liao, DING Yan, LIU Kun-lin, MA Ke-lin, CHANG Jun-sheng. Research Advance on BFT Consensus Algorithms[J]. Computer Science, 2022, 49(4): 329-339.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210700011

区块链 BFT 共识算法研究进展

冯了了　丁　滟　刘坤林　马科林　常俊胜
国防科技大学计算机学院　长沙 410073
(llfeng@nudt.edu.cn)

摘　要　自 2008 年比特币问世后,区块链逐渐成为学术界的研究热点,共识算法作为区块链的关键技术,受到了越来越多研究者的重视。由于区块链运行环境复杂多变,容易在系统中引入拜占庭节点,因此区块链拜占庭容错共识算法是必须要攻克的难关。文中系统地总结了区块链拜占庭容错共识算法的研究进展,以期为未来共识算法的创新提供参考。首先,梳理了现有的区块链拜占庭容错共识算法的四大派别,引出了 BFT 共识算法;其次,回顾了经典 BFT 共识算法 PBFT 中的几个重要临界值及其正确性证明;再次,提出了 BFT 共识算法具有去中心化、效能、安全性和容错率 ntg 四大优化目标;然后,基于共识轮次、共识节点个数、底层硬件、通信模式或加密算法、出错概率等维度,归纳出 BFT 共识算法的 5 种优化思路;最后,对 10 种经典 BFT 共识算法进行了详细分析与性能对比。

关键词:BFT;PBFT;优化;拜占庭容错;共识算法;区块链;分布式系统
中图法分类号　TP311

Research Advance on BFT Consensus Algorithms

FENG Liao-liao,DING Yan,LIU Kun-lin,MA Ke-lin and CHANG Jun-sheng
School of Computer Science,National University of Defense Technology,Changsha 410073,China

Abstract　Since the advent of Bitcoin in 2008,blockchain has gradually become a research hotspot in academia. As the key technology of blockchain,consensus algorithm has also attracted more attention from researchers. It's easy to introduce Byzantine fault nodes in blockchain system because of its complex and variable runtime,so the blockchain Byzantine fault tolerant consensus algorithm is a difficulty that must be overcome. This paper systematically summarizes the research progress of the blockchain Byzantine fault tolerant consensus algorithm,in order to provide a reference for the innovation of consensus algorithms in the future. Firstly,sorting out the four major factions of the existing blockchain Byzantine fault tolerant consensus algorithms and introducing the BFT consensus algorithm. Secondly,reviewing several important values in the classic PBFT algorithm and its correctness proof. Thirdly,putting forward the four optimization goals of the BFT consensus algorithm:decentralization,efficiency,fault tolerance rate and security. Then,based on the dimensions of consensus rounds,number of consensus nodes,underlying hardware,communication mode or encryption algorithm,probability of fault nodes,five optimization ideas of BFT consensus algorithm are summarized. Finally,analysising 10 classic BFT consensus algorithms in detail and making performance comparison.

Keywords　BFT,PBFT,Optimization,Byzantine fault tolerant,Consensus algorithm,Blockchain,Distributed system

1　引言

近年来,比特币的大获成功吸引了众多研究者对其底层技术区块链的关注。区块链的本质是融合了密码学和 P2P 等技术的分布式账本,因此传统分布式系统所面临的共识问题也是区块链中的难题。

共识是分布式系统中各节点达成一致性的过程,一致性指各节点数据的值相同,保持各节点的一致性是系统能安全对外提供服务的关键。在分布式的经典问题"状态机复制"中,每个节点从一致的初始状态开始,按照一致顺序执行指令,最终到达一致的新状态。为了达成此目标,各节点需要对指令的执行顺序达成共识,具体到区块链场景下,需要达成共识的内容则是区块的顺序。

在分布式系统中,非拜占庭错误节点可以表现为宕机错误。在区块链出现之前,传统分布式系统可以通过准入机制和管控等手段来剔除系统中的拜占庭错误节点。因此 VR[1] 和 Paxos[2] 等许多非拜占庭容错共识算法早在 20 世纪 80 年代就已经被提出并成功应用于生产中。Ongaro 等于 2014 年基于 Paxos 提出了简化版的 Raft 算法,其在工业界得到了广泛应用。非拜占庭容错算法可被广泛用于分布式系统中的

到稿日期:2021-07-01　返修日期:2021-12-10
基金项目:国家自然科学基金(U19A2060,61502510)
This work was supported by the National Natural Science Foundation of China(U19A2060,61502510).
通信作者:常俊胜(junshengchang@nudt.edu.cn)

存储、消息发送和服务发现等场景。2006 年 Google 的分布式锁服务 Chubby 将 Paxos 算法引入工业界。在 Raft 算法提出后，工业界更是喷涌出大量优秀的应用，如 PingCAP 的分布式事务型的键值数据库 TiKV、阿里巴巴的分布式消息系统 DLedger、Google 的服务发现框架 Consul、HashiCorp 的集群和应用程序调度服务 Nomad 等。

在分布式系统中，拜占庭错误节点可以表现出任意行为，如改变自己的状态、发送内容错误的消息或者向不同进程发送不同的消息等。拜占庭容错(Byzatine Fault Tolerant，BFT)共识算法则是为解决拜占庭错误(Byzatine Fault)而设计的。拜占庭容错共识算法的目标是，在一个含有拜占庭错误节点的比例不超过某个阈值的系统中，使得所有正确节点就目标值达成一致，同时保证系统的可用性与活性。

拜占庭容错是比非拜占庭容错更严格的容错要求，因此非拜占庭容错共识算法能解决的问题，拜占庭容错共识算法也能解决。但是，在实际生产中，我们需要权衡容错与效率功耗等各个要素。区块链正是拜占庭容错共识算法的绝佳应用场景，因为区块链中没有可信的第三方媒介。区块链与传统分布式共识场景的最大区别在于，其运行环境复杂开放，更容易引入拜占庭节点，因此在区块链诞生之后，拜占庭容错共识算法的研究热度逐渐上升，催生出了大量优秀的拜占庭容错共识算法。区块链拜占庭容错共识算法发展至今，可以大致分为以下 4 个派别。

(1) BFT 共识算法。这类算法是经典分布式拜占庭容错算法在区块链场景中的优化，目标是提高其性能与可扩展性，节点之间通过多次投票来达成共识。系统中假设拜占庭节点不超过总结点数的三分之一，否则算法的正确性将无法保证。Pease 等于 1982 年定义了拜占庭将军问题[3]，同时给出了基于口头消息和签名消息的两种拜占庭容错共识算法 OM 和 SM，但这两种算法的消息复杂度均为指数级，高昂的复杂度使其一直停留在理论研究中。Castro 等于 1999 年提出了实用拜占庭容错算法(Practical Byzantine Fault Tolerance，PBFT)[4]，该算法的消息复杂度降低为多项式级，成为了首个适用于实际生产生活的 BFT 算法，同时打开了 BFT 共识算法的思路。随后，研究者们在 PBFT 的基础上衍生出了一系列优化算法，并针对区块链的应用场景对其进行了优化。Tendermint[5] 团队于 2014 年提出了一种共识算法，简化了 PBFT 中的消息类型。同年，Copeland 等结合 Raft 和 PBFT 提出了 Tangaroa[6] 算法，该算法继承了 Raft 简单易懂的优点，并且直接启发了研究者们提出具备更优性能的 ScalableBFT[7] 算法。

(2) 基于某种资源证明的算法。这类算法的思想承袭于比特币，也多应用于数字货币场景。参与者通过掌握虚拟或物理资源来争夺记账权，获得记账权的概率与节点掌握资源的量成正比。应用该类算法的系统假设攻击者无法同时集中超过 50%的资源，否则账本可能面临分叉的威胁。2008 年，Nakamoto 首次在比特币项目中提出基于算力证明的共识算法(Proof of Work，POW)[8]，该算法的安全性超群，但其对算力的浪费一直被诟病。2012 年，King 首次在点点币(Peercoin，PPC)项目中实现了基于股权证明的共识算法(Proof of Stake，POS)[9]，该算法有效改善了 POW 对算力资源的浪费，但也容易造成中心化，以及引发无利害关系(Nothing at Stake)问题。2013 年，比特股项目提出了代理权益证明算法(Delegated Proof of Stake，DPOS)，在 POS 的基础上，利用委员会制度大幅缩短了共识周期。Zhong 等人基于 POS 提出性能改进的 Silkworm[10] 算法，保证了区块链的安全性和健壮性。Ateniese 等于 2014 年定义了一种基于计算机硬盘空间证明的共识算法(Proof of Space，PoSp)[11]。Ball 等于 2017 年提出的有益工作证明(Proof of Useful Work，PoUW)[12] 则将 POW 中无意义的算力竞争转化为对有意义的数学问题的求解。这类共识算法还包括权益流通证明(Proof of Stake Velocity，PoSV)、燃烧证明(Proof of Burn，PoB)和行动证明(Proof of Activity，PoA)等[13]。

(3))带有信任机制的投票算法。这类算法与 BFT 类算法同为投票上链的共识算法，不同点在于 BFT 类算法是给区块投票，而此类算法是通过多次给交易投票来缩小交易候选集。此类算法一般依赖于某种定制的信任机制，即信任其中的某些代表节点，因此其中心化程度更高，同时性能也相对更好。Schwartz 等于 2014 年提出了瑞波协议共识算法(Ripple Protocol Consensus Algorithm，RPCA)[14]，该共识算法先针对交易投票，通过将赞成率高于某个阈值的交易组成交易候选集，不断增加阈值，以生成新的交易候选集，最终将赞成率高于 80%的交易打包成区块。该共识算法解决了异步网络的高延迟问题，具有低延迟和高鲁棒性的特点。2015 年，Stellar.org 首席科学官 Mazieres 教授结合联邦拜占庭协议和 RPCA 协议提出了恒星共识协议(Stellar Consensus Protocol，SCP)，该共识算法是第一个安全可证的联邦拜占庭协议(Federated Byzantine Agreement，FBA)，能有效节约计算和经济资源，具有分散控制、低延迟、灵活信任和渐近安全的属性[15]。随后，超级账本结合 RPCA 和 SCP 提出了法定人数投票共识算法(Quorum Voting)[10]，该算法可立即确定交易。

(4) 基于有向无环图(Directed Acyclic Graph，DAG)的共识算法。近年来兴起的一类新型的有向无环图共识算法不再采用链而采用 DAG 结构来存储数据，将共识的最小单位设为一个交易，省去了将多个交易打包成区块的步骤。这类算法无需矿工角色，具有全民记账的特点，这是为解决链式结构并行性能差的问题而做出的重大改进。其中，Byteball[16] 将单个交易作为独立单元，每个单元包含一个或多个之前单元的 Hash 值，算法的安全性依赖于恶意见证人数量不超过总见证人数量一半的假设，但该算法的吞吐率较低。Avalanche[17] 中的每笔交易都需要通过全网节点发送的投票来进行确认，因此网络资源消耗较高。在 Hashgraph[18] 中，节点可以根据交易的验证历史来计算交易所获得的全网其他节点的投票，从而判断部分交易是否达成共识，这种设计减少了额外通信。此外，IOTA[19] 针对物联网区块链系统设计了特殊的 Tangle 数据结构。Conflux[20] 中一个区块以之前的两个区块为父节点，其效率的瓶颈不再是共识算法，而是节点的处理能力。此类有向无环图共识算法允许多线程交易，但往往难以解决双花问题。

4 个派别各有优劣，分别侧重于不同场景，运用不同的思路实现了拜占庭容错。派别(1)和派别(3)基于节点投票，使得少数恶意节点无法影响全局决策，派别(2)和派别(4)则

需要多个追加的区块/交易来对历史区块/交易进行确认，这意味着越靠前的区块/交易越安全，参与系统的节点被某种奖惩机制约束，这些奖惩机制旨在提高作恶成本，使节点作恶变成非理性行为。

本文致力于研究派别(1)的 BFT 共识算法的进展。该类算法是区块链拜占庭共识算法的经典思路，节点依靠相互传递信息来做出最终决策，且通信开销较大，因此更适用于联盟链场景。相比其他几个派别的共识算法或基于概率来确认区块，BFT 共识算法几乎可以做到上链即确认，即交易一旦上链就不可回滚。此特性可以杜绝如自私挖矿攻击、双花攻击之类的安全隐患。

由于 BFT 共识算法大多受到 PBFT 的启发，因此本文先回顾了 PBFT 算法，再对各经典 BFT 共识算法进行了分析。本文首先分析了经典的 PBFT 算法；其次提出了去中心化、效能、安全性和容错率 4 个优化维度；然后从优化共识轮次、匿名选举委员会、通信基础设施、结合可信硬件、投机与乐观 5个方面分析了现有的共识算法的优化思路；随后总结了各优化思路的优缺点，并从吞吐量、交易确认时间等效能指标和容错率维度对 10 种优化算法进行对比；最后总结全文并展望未来。

2 经典 PBFT

经典 PBFT 算法具有运行效率高和上链即确认的优点，在实际生产生活中 PBFT 类共识算法的应用已经非常成熟，许多项目采用 PBFT 或其改良作为共识层算法，如 Hyperleger Fabric，Tendermint 等。现有的流行的 BFT 算法受 PBFT 的影响很深，因此有必要对其展开深入分析。

2.1 拜占庭将军问题

Lamport 等于 1982 年描述了拜占庭将军问题：在古老的拜占庭王朝，多位将军驻守在一座城市外，某个时刻各将军需要就“进攻还是撤退”这个问题协商达成一致意见，于是需要通过信使传递自己的投票，通过自己和别人的投票来做出决策。但将军可能叛变，信使可能在中途遭遇不测，消息可能丢失或被篡改。拜占庭将军问题的目标是忠诚的将军通过某种算法最终能达成一致的决策，具体到计算机分布式领域，拥有独立处理能力的节点对应该问题中的拜占庭将军，信道对应该问题中的信使。节点可能发送错误的投票，还可能针对不同的接收者发送不同的投票；信道是不可信的，消息可能丢失、延迟、乱序。

能解决拜占庭将军问题的算法统称为拜占庭容错算法。Lamport 在提出问题的同时也提出了基于口头消息的算法 OM 和基于签名的算法 SM，两者的消息复杂度都为指数级。20 世纪 80 年代，研究者们还提出了其他的拜占庭容错的算法，如 FTMP[21]，MMFCS[22] 与 SIFT[23] 等。

2.2 PBFT 算法

PBFT 算法由 Castro 等于 1999 年提出，它是对传统 SM 方案的改进，将算法复杂度从指数级降到了平方级。

PBFT 算法包括常规机制、垃圾回收机制和视图更换机制这 3 个子机制。系统在运行良好时处于常规机制，垃圾回收机制用于清理无用的日志消息，主节点的错误行为会触发视图更换机制。视图是一系列配置，一个视图中有一个主节点和若干副官节点，不同视图中各节点轮流担任主节点，一个视图可以类比为一个主节点的任期，视图编号连续增长。

常规机制包含 3 个阶段，即 pre-prepare 阶段、prepare 阶段和 commit 阶段，如图 1 所示。图 1(a)为客户端与主节点的交互流程，图 1(b)为各节点共识模块之间的一轮通信，虚线连接的两部分是等价的，各节点运行相同的共识模块。在收到来自客户端的请求后，主节点开始 pre-prepare 阶段，给每个请求分配唯一编号并广播 pre-prepare 消息，该消息中包含编号和请求的映射关系。副官节点在 pre-prepare 阶段和 prepare 阶段进行投票，通过两轮投票的节点执行请求，然后给客户端回复。pre-prepare 阶段和 prepare 阶段确保在同一视图中的一致性，prepare 阶段和 commit 阶段保证被提交的请求跨视图的一致性，而视图更换保证了当主节点为错误节点时的系统活性。

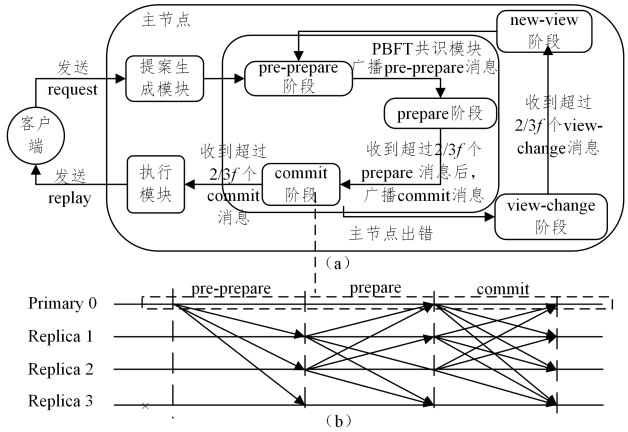

图 1 PBFT 系统与 PBFT 共识流程

Fig. 1 PBFT system and PBFT consensus process

PBFT 中有 3 个重要的临界值。假设系统中的恶意节点最多有 f 个，那么系统需要至少 $3f+1$ 个节点才能确保算法的正确性，投票环节各节点需要至少 $2f+1$ 个投票才能做出决策，客户端确认结果需要收到至少 $f+1$ 个来自不同副官节点的相同回复。若某节点在 prepare 阶段收到 $2f+1$ 个某值对应的 prepare 票，则称该节点锁定该值；若某节点在 commit 阶段收到 $2f+1$ 个某值对应的 commit 票，则称该节点决定该值。

算法的正确性包括算法的安全性和活性。在一个节点个数为 $3f+1$ 的系统中，节点数大于等于 $2f+1$ 的子集之间的交集必有一个正确节点。由此可推出，在 prepare 阶段不会有正确的节点锁定不同的值，从而确保了同一轮中的一致性。同时，若主节点作恶，导致正确节点长时间不能决定某个值，则会触发换主条件，算法的活性因此得到保证；若在换主时某个值已经被部分正确节点决定，则该值会被带入新的视图中，最终所有正确节点都会决定该值，即不会有正确的节点决定不同的值，确保了不同轮间的一致性。同一轮中的一致性通过视图更换机制，扩展为不同轮间的一致性，最终确保了算法的安全性。

PBFT 的提出早于区块链诞生近 10 年，因此并未针对该场景做任何优化。PBFT 解决的问题可以概括为多节点就连续请求的排序达成一致。这里提到的请求形式并不固定，因此 PBFT 能解决任意确定性的状态机复制问题。例如，当

PBFT 被应用于数据库集群，请求可以是数据库日志；而就区块链场景而言，请求则可以是一个区块。Tendermint 即 PBFT 在区块链场景下的优化，除了算法上的改良，Tendermint 还加入了打包和校验区块的内容。就应用场景而言，PBFT 与 Tendermint 的关系可类比为程序的接口与实现。

3 优化目标

在分布式领域中常使用 FLP 不可能原理、CAP 定理和 BASE 模型来指导算法设计。FLP 不可能原理于 1985 年被提出，该定理证明了在异步通信场景下，分布式系统中只要有一个进程发生故障，任何共识算法都无法保证其完全正确。由于 FLP 的证明采用的模型比实际情况严格，因此其具有很强的普适性。FLP 被提出后，学术界开始寻找能够进行取舍的分布式系统设计因素。Brewer 等于 2000 年提出了 CAP 定理。该定理首次将一致性、可用性和分区容忍性这 3 个要素提炼为分布式系统设计的重要特征，并猜想这三者无法在分布式系统中同时被满足，需要根据业务特点进行取舍。在此基础上，BASE 模型对 CAP 定理中的一致性和可用性进行了权衡，其核心思想是强调系统的基本可用性，支持分区失败、软状态及最终一致性。BASE 模型的软状态允许系统状态在短时间内不完全同步，但必须满足最终数据的一致性。在 BFT 共识算法中，当错误节点超过阈值时，通常选择牺牲可用性来维护一致性。如何对上述 3 个要素进行取舍应取决于具体应用场景。

通过对 BFT 共识算法的分析研究，可以从去中心化、效能、安全性和容错率 4 个维度来评价该类算法。

3.1 去中心化程度

共识算法的去中心化程度可以用各个节点功能的对等程度来衡量，在传统的 P2P 应用中，各个节点的功能完全对等，可以称为完全去中心化。在大多数区块链共识算法中，有一个单独的打包区块的节点，如 POW 中的记账人、PBFT 视图中的主节点。特别在选举委员会一类的共识算法中，委员会节点的职能高于一般节点，这些都是中心化的体现，但是去中心化不代表完全没有中心化节点，我们可以用中心化节点的权利被削弱的程度和中心化节点的选取规则来评判算法的去中心化程度。

3.2 效能维度

从效能角度来讲，共识算法的效能可以用其能耗、吞吐率、区块生成时间、交易确认延迟、节点可扩展性和动态性等参数来衡量。能耗指算法在运行过程中的算力，如网络、内存磁盘空间和电力等资源消耗；吞吐量指每秒钟系统可以处理的交易数量；区块生成时间指从打包交易到上链整个过程的时间；交易确认时间指一笔交易从生成到确认上链的时间；节点可扩展性指算法在运行中加入新节点的能力；动态性指算法中的参数根据网络或者其他运行环境动态调整到更优的能力。能耗越小，吞吐率越大，区块生成时间和交易确认时间越短，节点可扩展性越高，动态性就越高，共识算法的性能也越好。

3.3 安全维度

安全维度关注系统是否具有抵御各种攻击的能力，在区块链环境下，常见的攻击包括双花攻击、女巫攻击、月食攻击、自私挖矿攻击和无利害关系攻击等。双花攻击指将一份钱花两次并控制超过一半的算力，使两笔不合法的交易上链；女巫攻击指攻击者伪造多个虚假身份来控制网络，以多数票击败网络上的诚实节点；月食攻击指为了控制某个节点，攻击者先控制与目标节点相连的其他节点，进而达到操控目标节点的目的；自私挖矿攻击指先挖到矿的节点不发布区块，而是基于挖到的区块继续挖矿，最后直接发布最长链；无利害关系攻击指攻击者在过去不同的分叉链上进行挖矿，以获取更大的利益。

3.4 容错率

从容错率角度来讲，在传统的拜占庭容错问题中，系统中的错误节点不能超过系统节点总数量的 1/3，否则算法的正确性或活性无法得到保证。但现在也有许多共识算法提升容错率。

共识算法优化的 4 个维度如图 2 所示，在实际的设计中需要在 4 个维度中进行权衡。通常提升效能需要降低一定的去中心化程度，同时也会导致容错率和安全性的降低。

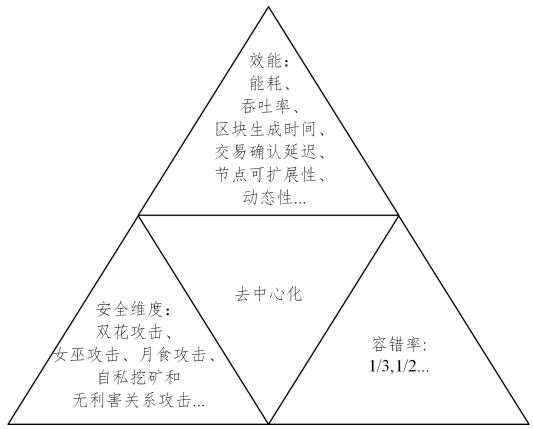

图 2 共识算法优化的 4 个维度

Fig. 2 Four dimensions of consensus algorithm optimization

4 优化思路

相比 Lamport 首次提出拜占庭错误时给出的方案，虽然 PBFT 已经有了指数级的性能提升，但平方级的时空复杂度和较低的容错率还是使得其可扩展性较低，同时，随着计算机技术的发展，算法在底层硬件的使用和通信机制的选择等维度也具有了可优化的选择。因此，研究者们提出了多种优化策略，这些策略主要的优化思路可以归类为以下 5 种。

4.1 主线 1：优化共识轮次

PBFT 为常规机制、垃圾回收机制和视图更换机制分别设计了不同的消息格式。Kwon 于 2014 年创造的 Tendermint[5] 使用变量代替 PBFT 中视图更换消息的功能，将 PBFT 的视图更换机制与常规机制进行了统一，当节点锁定某个值时，仅更新相关变量而无须将消息留存于日志中。基于此，Tendermint 将 PBFT 中的垃圾回收机制也进行了删减。经简化的 Tendermint 算法仅有 3 个阶段，相比 PBFT 更加简明易懂。

4.2 主线 2：选举委员会

参与共识的节点数对 PBFT 共识算法的消息复杂度有着直接影响，因此研究者们试图精简参与共识的节点数量。

采取这类措施的算法被 Fu 等归纳为委员会＋投票模式[24]，此模式从系统所有的节点中选出一部分作为委员会，委员会节点参与共识，然后将共识结果发布给其他节点，从而提升系统的可扩展性。

选举委员会有多种可选方案。一些算法受到 PoW 和 PoS 算法的启发，依据节点关于“某种能力的证明”来进行委员会的选举。dBFT[25] 和 Tendermint 等算法都基于 PoS 来选举委员，这类算法将节点与节点准入的保证金联系在一起，以解决原生 PoS 的无利害关系问题。Elastico[26] 和 ByzCoinX 等[27]算法基于 PoW 算法选举出领导者，选举领导者的过程和交易验证完全分开。Permacoinp[28] 和 Spacecoin 等[29]算法则基于参与者空闲磁盘大小来选举领导者。而选举领导者的策略稍经修改就可以用作选举委员会的策略。除此之外，Elastico 还采用分片的思想将挖矿网络分为一些较小的委员会，每个委员会并行地运行拜占庭共识算法来处理相互独立的交易。ByzCoinX 共识算法借鉴了 Elastico 算法，使用类似二级树的形式改变了共识组之间的信息传播机制，通过牺牲部分高扩展性来获得鲁棒性。PoPT[30] 提出了一种衡量节点在公有链系统中的参与度的算法，并基于参与度排名选举委员会。VPBFT[31] 则引入了 PoV 机制。

随机可验证函数（Verifiable Random Function，VRF）也常被用于委员会的选举。VRF 是一种将输入映射为可验证的随机数输出的加密方案，具有可验证性、唯一性和随机性。老牌公链项目 Difinity[32] 和本体项目 Ontology 的 VBFT[33] 将 VRF 用作一般伪随机函数，即给出相同输入以得到相同输出。这类算法将上一个区块的哈希作为输入，通过 VRF 来决定参与本轮共识的节点集合，以保证委员的不可预测性。

选举委员会减少了共识成本，带来了性能的提升，但是也削弱了区块链去中心化的优点，从而引入了不安全因素。有些算法使用匿名选举委员的方式来降低委员节点被恶意节点攻击控制的可能性。匿名选举可以通过随机可验证函数 VRF 来实现，节点可将约定信息和自己的私钥作为 VRF 函数的输入以得到一个哈希值和一个证明，拥有对应公钥和证明的人可以验证该哈希值是否由对应节点生成。利用 VRF 函数选出的委员会，在投票之前都可以隐藏自己的身份，做到匿名投票，该方法可以防止委员会提前暴露，从而遭到恶意攻击，进而减少视图更换，加快共识进程，且能抵抗自适应攻击。BA* 算法[34] 利用二项分布的 VRF 函数选出委员会。除 VRF 外，还有一些密码学算法可为委员会或主节点提供匿名性，如 Sun 等提出的 RBFT 算法[35] 利用环签名也实现了每一轮共识中主节点的匿名选举。

4.3 主线 3：选用合适的签名算法或底层通信模式

系统采用的底层数据结构、通信模式以及密码学算法也会影响共识算法的设计。

门限签名是一种分布式多方签名协议，由 Desmedt-Frankel 于 1991 年提出。在 (n,t) 门限签名中，n 个成员作为一个签名群体具有一对公私钥，群体内超过 t 个任意成员的签名可以组合成群体签名，以被群体公钥验证。近年来，用门限签名替代广播投票的方案在区块链共识算法中被广泛应用，如 RBFT[35]，SBFT[36] 和 HotStuff[37] 等，通过引入门限签名将算法顺利进行时的系统通信复杂度降至 $O(n)$。随后，Jalalzai 等改进 HotStuff 算法提出 Fast-Hotstuff[38] 算法，在共识的 NewView 阶段使用聚合签名代替门限签名，提高了共识算法在第二轮投票环节发生错误时的执行效率。Li 等则对 Fast-Hotstuff 算法进行补充，提出一种改进的 Fast-Hotstuff 算法[39]，引入一个新的区块扩展方式，提高了共识算法在第一轮投票环节发生错误时的执行效率。

Gossip 由施乐帕洛阿尔托研究中心于 1987 年提出，是 P2P 对等网络的一种通信模式。与 PBFT 中全连接的广播不同，Gossip 中的节点周期性随机选择相邻节点发送消息。Gossip 协议基于六度分隔理论，传播消息迅速，被广泛用于数据同步和消息泛洪等。许多共识算法都采用了该协议，如前面提到的 Tendermint 及其衍生算法等。Zhang 等提出的基于 Gossip 协议的共识算法 GBC[40] 提高了系统的容错性，使系统的容错率从 1/3 提升到了 1/2。Zhang 等提出的 CCG[41] 算法优化了 Gossip 协议，通过评估临近节点的通信度而非盲目泛洪消息，提高了消息传递的稳定性。

4.4 主线 4：结合可信硬件

另一种思路是将 BFT 共识算法与可信硬件结合，用可信硬件约束作恶节点，使拜占庭节点只能产生非拜占庭错误。引入可信硬件之后的 BFT 算法的容错率由 1/3 提升到了 1/2，但由于可信硬件的存储空间大多有限，很难容纳全部共识代码的体量，因此这类算法大多在可信硬件中实现一个单调递增的计数器功能，将某个计数器的值 i 与某些消息进行唯一绑定，使得节点不能执行不同的第 i 个操作，从而让节点从拜占庭节点退化为非拜占庭节点，即节点只能发生宕机错误。如 A2M-PBFT[42] 选择使用可信硬件 TPM 来实现这个单调计数器的功能；MinBFT[43]，CheapBFT[44] 和 FastBFT[45] 等则在可信执行环境（Trusted Execution Environment，TEE）中实现该计数器功能。

4.5 主线 5：投机与乐观

鉴于真实生产环境中节点很少出错这一事实，一种投机的优化思路是投机 BFT（Speculative BFT），即系统通常针对较长时间的良好情况快速达成共识，只在节点出错的情况下回退到拜占庭容错共识算法，如算法 Zyzzyva[46] 和其与可信硬件的结合版本 MinZyzzyva[43]。Zyzzyva 中参与共识的副本节点按照领导者提出的顺序执行请求，而不运行任何明确的一致协议。执行完成后，将所有副本回复给客户端。如果领导者模棱两可，客户端将收到不一致的回复，此时客户端会纠正副本节点从不一致状态恢复到公共状态，然后执行完全的 PBFT 算法。另一种类似的优化思路是乐观 BFT（Optimistic BFT）[47]，在这种思路下，只需要一部分副本来运行一致协议，其他副本被动地更新它们的状态，只有在一致协议失败时才会积极地参与其中，运用该类优化思路的有 OBFT[48]，CheapBFT[44] 和 FastBFT[45] 等。这类算法在环境较理想时先在子节点范围内运行简单的共识，若共识成功，则其他正确节点只需更新共识结果，若发现节点的错误行为，则激活其他节点，如 CheapBFT 在节点出错时退化为 MinBFT[43]。乐观 BFT 与投机 BFT 的不同之处在于，投机 BFT 在节点不出错的情况下不需要运行明确的协议。

4.6 优化思路对比

优化共识轮次的思路针对共识机制进行了改进，加快了

共识进程，实施此种思路需要精确掌握原生算法；选举委员会的思路牺牲了一定的去中心化性能，提高了算法的效能，但也降低了安全性，容易遭受女巫攻击、月食攻击和拒绝服务攻击。但该思路可以通过提高更换委员会的频率、使用 VRF 等技术引入委员会的匿名性等手段来降低攻击的成功率；优化签名算法或底层通信模式的思路通过减少消息的复杂度或规避主节点作恶行为来加速算法进程，提高了算法效能；结合可信硬件的思路从根源上保护了共识算法中容易被攻击者操纵的部分，杜绝了节点的作恶行为，将算法容错率提高到了1/2；投机与乐观的思路能在节点不出错时快速达成共识，在提升算法效能的同时在其他维度没有明显损失。综上，5 种优化思路的优缺点如表 1 所列。

表 1 5 种优化思路的优缺点

Table 1 Advantages and disadvantages of five optimizations

思路	优点	缺点
主线 1：优化共识轮次	从算法设计上优化，易于共识流程的理解。	此种优化思路较个性化，门槛高，普适性不强
主线 2：选举委员会	提高了共识效能	削弱了去中心化程度，容易受到恶意节点攻击和控制，安全性降低
主线 3：选用合适的签名算法或底层通信模式	给上层算法的设计带来便利；减小消息复杂度	对底层密码算法或通信模式有特殊要求
主线 4：结合可信硬件	最小化可信基，减小受攻击面，放宽了安全假设，从而给上层算法的设计带来了便利，提高了算法的容错率	对硬件要求更高，成本提高，维护困难
主线 5：投机与乐观	在运行良好的情况下极大地提升了算法效能和平均效能	在运行状况较差时效能比未优化时更低

5 经典 BFT 共识算法

本节详细分析了经典的 10 种共识算法机制，部分算法结合了不止一种上述优化思路。

5.1 经典 BFT 共识算法的详细分析

5.1.1 dBFT 共识机制

委托拜占庭容错(dBFT)是小蚁链(Neo)结合了 PoS 与 PBFT 机制提出的算法。节点根据自己的股权进行投票，选出委员会进行共识，非委员会节点不参与共识，但是可以看到整个流程。选举的机制让 dBFT 具有良好的性能和可扩展性，适应公链场景，dBFT 中引入了数字身份，这意味着委员可以是个体或者某些机构。dBFT 是基于 PBFT 的改进算法，具有上链即确认的特性，避免了 PoW 和 PoS 的确认时间和隐患，能在金融场景的应用中提供更好的安全性。NEO 生成一个区块需要 15～20 m，交易吞吐量约为 1 000 TPS，优化后的算法更能达到 10 000 TPS。虽然 NEO 采用的 dBFT 共识机制投票选举委员会提高了性能，但选举流程是静态的，选举方案和结果完全由项目方决定，因此 NEO 也被外界质疑过于中心化。

5.1.2 Tendermint 共识机制

Tendermint 是针对区块链的共识算法。术语“高度”是当前共识区块在链上的编号，一个高度可能需要多轮才能达成共识，一轮共识包含提议、投票和预提交 3 个阶段，与 PBFT 常规机制中的 3 个阶段一一对应，如图 3 所示。Tendermint 中轮次的更换对应着 PBFT 中视图的更换。与 PBFT 不同的是，如果不赞成某个值，Tendermint 中的节点会投出空票，这个设计加速了共识的推进；同时，即使主节点没有作恶，Tendermint 也有可能进入新的一轮。最后，由于链上区块需要包含前一个区块的哈希值，因此各区块的共识不能并行，Tendermint 的设计也不像 PBFT 一样能并行执行多值的共识。之后学术界基于 Tendermint 又衍生出了 Casper[49]，Ouroboros[50]，Tezos[51] 和 Honeybadger[52] 等新的共识算法。这类算法常因与 PoS 的结合而被关注，且其对 PBFT 算法的轮次也进行了大量优化。

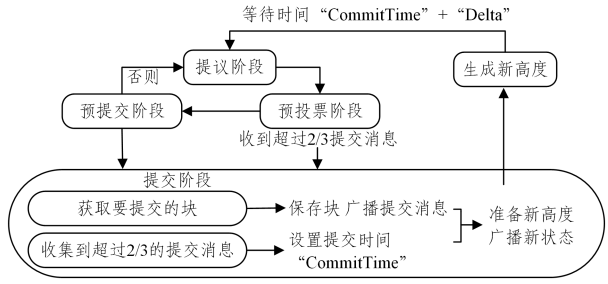

图 3 Tendermint 的优化 3 个阶段

Fig. 3 Three stages of Tendermintoptimization

5.1.3 RBFT 共识机制

Huang 等于 2020 年提出的 RBFT[53] 共识算法结合改进的 Raft 机制，选举出领导者组建委员会进行 PBFT 共识。首先采用一致性 Hash 算法对网络节点进行分组，同时将同一监督节点分配到不同的组内，各分组内运行 Raft 机制并选举领导者，领导者组建委员会运行 PBFT 共识算法，监督节点对所在分组领导者的日志消息的签名进行验证和比较，从而判断领导者是否为拜占庭恶意结点。共识过程如图 4 所示，包含委员会 PBFT 共识和组内 Raft 共识。在 RBFT 两级共识阶段，利用 PBFT 的共识特性和 Raft 的共识特性保证了算法的一致性和安全性，同时，PBFT 阶段的视图切换和 Raft 阶段的领导者选举等为 RBFT 共识提供了活性。仿真实验结果显示，相比 PBFT 共识机制，RBFT 在网络节点数量为 117 时降低了 98.8%的通信开销，相同节点规模下共识时延远短于 PBFT，吞吐量为 PBFT 的 300%～400%，且具有更高的容错性。

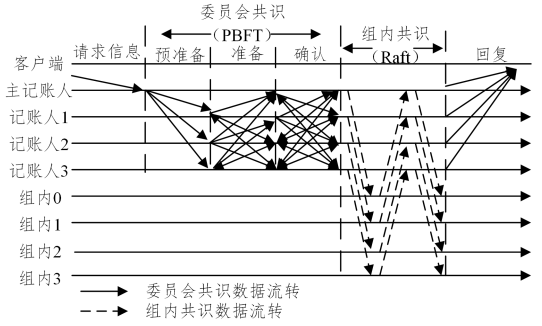

图 4 RBFT 共识流程图

Fig. 4 RBFT consensus process

5.1.4 VPBFT 共识机制

Qang 等于 2019 年提出了 VPBFT 共识算法，利用 POV 机制提升了 PBFT 算法的动态性。POV 机制下的网络节点分为普通节点和特殊节点，特殊节点又包括 3 类角色：投票

节点、候选节点、生产节点。普通节点可以灵活地接入或退出网络，确保了算法的可扩展性；投票节点由经过实名认证的普通节点构成，能够选举和监督生产节点；候选节点由普通节点申请而成，可由投票节点投票变为生产节点；生产节点被随机指定生成区块，投票节点负责监督这些区块的合法性，若生产节点生成有效区块，则可优先进入下一轮，生产节点任命到期后退化成候选节点，候选节点可选择退出候选行列退化为普通节点。网络中所有节点都能转发并验证交易，有效的交易会被发送给投票节点和生产节点，并最终由生产节点放入交易池，生产者从交易池打包区块，一个区块的确认需要得到过半的投票节点同意。不同类型的节点之间存在一定的比例关系，算法可根据网络情况动态地调整比例参数。VPBFT 算法的网络模型如图 5 所示。该算法分别在能耗、时延、容错性和动态性 4 个维度上对 PBFT 进行了提升。

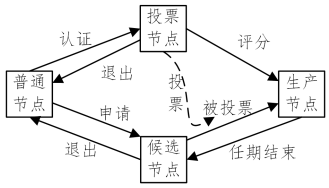

图 5 VBFT 算法的网络模型

Fig. 5 Network model of VBFT algorithm

5.1.5 Dfinity 共识机制

Dfinity 利用 VRF 选举委员，将算法架构分为 4 层：第一层为身份认证层，用于认证加入系统的节点身份；第二层为随机灯塔层，用于生成不可预测的随机数，该随机数被用来确定节点的优先级或决定委员会成员；第三层为区块链层，用于将交易打包为区块；第四层为验证层，用于区块的共识。Dfinity 共识机制在初始化阶段随机选出多组委员会，各节点在每一轮开始都使用上一个区块的哈希值作为参数来调用第二层的随机函数，生成一个统一的不可预测的随机数 ξ_r，由这个随机数来决定参与本次共识的委员会。一轮共识分为 PROPOSAL 和 NOTARIZATION 两个阶段。在 PROPOSAL 阶段，系统中任意节点都可以提议区块。Dfinity 共识机制根据本轮的随机数 ξ_r 来确定各节点的优先级，再根据节点优先级计算其提议区块的权重，优先级越高，权重越大。Dfinity 共识机制可以同时包括多条链，链的权重为链上的区块权重之和，在提议区块时，区块应该选择当前链权重最大的链作为前导链。在 NOTARIZATION 阶段，委员在等待一段时间后对收到的权重最大的区块进行签名并广播，直到某个区块获得超过一半的节点的联合签名，在此期间收到权重更大的区块也会签名并广播。Dfinity 采用门限签名技术，允许多条链同时存在，在多轮迭代之后会最终决定，因此不是上链即确认的算法。

5.1.6 Algorand 共识机制

Gilad 等于 2017 年提出的 BA* 算法是 AlgoRand[34] 中的共识解决方案。BA* 算法随机选举一定期望数量的节点子集作为委员会，由委员会成员运行共识算法。为了防止针对委员会的恶意攻击，网络节点各自独立运行可验证随机函数，VRF 判断自己是否是区块生成节点或共识节点，同时 VRF 生成能够验证共识结点身份的证明，其他用户能够依据该证明验证其身份。与 PBFT 算法不同的是，BA* 在共识的每个阶段都会选举新的委员会。BA* 共识算法包括 Reduction 和 BinaryBA* 两部分，其中 Reduction 算法中选中的委员会成员将观察到的区块广播，并收集其他节点投票的区块，提交超过阈值的区块，若超时则提交空块；BinaryBA* 算法在一个最大步数限制的情况下进行多次投票，首先选中的委员会成员对 Reduction 得到的正常区块或者空块进行投票，随后收集投票结果，同时将此时已经达成共识的区块状态标记为 final，取得足够数量 final 标记的区块将被持久地保存到区块链上。BA* 共识算法通过加密抽签的方式成功缩小了共识网络规模，提升了算法性能和扩展性。其次，VRF 的完全随机性和匿名性以及共识算法在每个阶段选举新的委员会又保证了委员会节点的安全性。

5.1.7 K-PBFT 共识机制

2019 年 Chen 等提出的 K-PBFT 共识机制[54] 采用改进的 K-medoids 聚类算法对 PBFT 的共识节点进行聚类和层次划分。改进后的 K-medoids 引入惰性系数 P 作为聚类中心节点更换的概率，使共识节点聚类过程更加可控。K-PBFT 根据自定义的聚类特征对共识节点进行聚类划分，将聚类特征相似度高的节点划分为一个簇。将簇内的聚类中心节点作为该簇内所有节点的主节点，将非聚类中心节点作为该簇内的从节点。每个簇为一个子共识集群，非聚类中心节点和 K 个主节点构成了 K 个子共识集群，骨干共识集群由 K 个主节点的集合构成，具体流程如图 6 所示，包含子共识集群 PBFT 共识阶段、骨干集群 PBFT 共识阶段、提交阶段和执行阶段。骨干共识集群在提交阶段对共识后的区块进行数字签名，并将此数字签名集合和共识区块广播给所属子共识集群中的所有从节点，从节点在验证阶段对区块附带的数字签名集合进行验证，从而决定是将区块记录上链还是对所属主节点的恶意行为进行举报。相比 PBFT，K-PBFT 单次共识耗时缩短了 20%，通信次数降低 3 个数量级，有效缩短了共识过程的耗时，明显减少了通信次数，提高了共识效率。

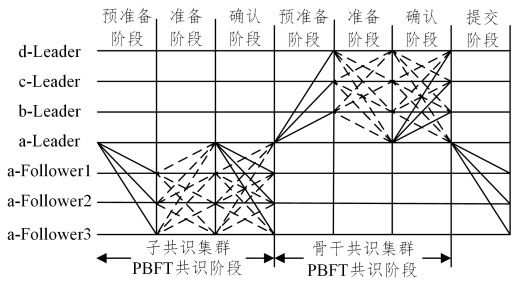

图 6 K-PBFT 共识流程

Fig. 6 K-PBFT consensus process

5.1.8 FastBFT 共识机制

FastBFT[42] 是基于可信执行环境 TEE 的乐观拜占庭容错算法。该算法分为 normal-case 和 fallback 两种模式，normal-case 模式针对通信环境良好的情况。在 normal-case 模式下只需一个节点子集来主动运行协议，流程大致与 PBFT 的流程对应，但 commit 和 reply 阶段被分为两部分，并且在共识开始之前新增了一个 pre-processing 阶段，在此阶段中，主节点的 TEE 中会生成多个组密钥，并将这些组密钥的哈希值公开，同时在 TEE 中有一个单调的计数器，TEE 将计数器

的每个值 c 与一个组密钥 S_c 进行唯一关联，然后把每个组密钥都分割成多个密钥碎片，TEE 之间可以构建起安全的信道，用于机密地分发密钥碎片。待客户端发起请求后，活跃节点开始运行 3 个阶段，如图 7 所示。

(1)prepare 阶段：主节点将客户端的请求 r 与一个计数值 c 绑定。由于在 pre-processing 阶段计数值 c 和某个组密钥 S_c 确定了绑定关系，从而请求 r 与组密钥 S_c 一一对应。将请求 r 及其与计数器 c 的绑定关系打包成 prepare 消息发送给各个活跃节点。

(2)commit 阶段：若活跃节点 i 认同收到的 prepare 消息，则将自己的密钥碎片 S_c^i 定向发给主节点表示自己投票。密钥的使用是一次性的，若主节点收到全部 $f+1$ 个活跃节点的密钥碎片(f 仍然表示最大恶意节点数量)，可以重组出组密钥 S_c，则节点执行请求 r 得到执行结果 res，并将其与下一个计数器值 $c+1$ 绑定，从而实现将执行结果 res 与 S_{c+1} 的绑定。然后，将 commit 消息发送给各个活跃节点，commit 消息中包含组密钥 S_c、执行结果 res 及其与计数值 $c+1$ 的绑定关系。活跃节点可以用 pre-processing 阶段收到的哈希值来验证组密钥 S_c。

(3)reply 阶段：各个活跃节点验证 commit 消息并通过后，执行请求 r 得到执行结果 res，然后获取 S_{c+1} 的密钥碎片发给主节点，若主节点能重组出 S_{c+1} 密钥，则将两个密钥(S_c 和 S_{c+1})及其对应的哈希值一起发送给客户端和非活跃节点。

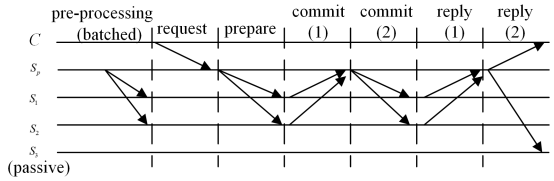

图 7 FastBFT 共识流程

Fig. 7 FastBFT consensus process

构造出正确的 S_c 密钥说明各活跃节点同意执行请求，构造出正确的 S_{c+1} 密钥说明各活跃节点已经执行了请求。主节点利用 *TEE* 保证了密钥的分发过程是安全的，即主节点不能伪造任一副本节点，同时密钥共享的机制降低了主节点聚合密钥的开销。

当故障检测机制检测到活跃节点中的错误节点超过一定阈值时，算法退化到 fallback 模式，fallback 模式类似于 MinBFT 算法。该模式在共识开始前同样进行预处理，投票机制的不同会导致密钥分配算法的不同，fallback 模式采用门限签名的密钥分配算法，此种模式下全部 $2f+1$ 个节点都需要参与共识，收到其中任意 $f+1$ 个密钥可生成组密钥。除了乐观的优化思路，FastBFT 还采用了多用户签名的方式代替传统共识算法的投票，在 TEE 中实现了关于密钥的拆分与分发，同时采用了树的通信结构来降低通信复杂度，设计了故障检测算法，从而提升了算法性能和可扩展性。

5.1.9 Zyzzyva 共识机制

Kotla 等提出的投机 BFT 算法 Zyzzyva[43] 可以在大多数正常情况下将状态机复制的开销降低到接近最优，达成仅 $O(n)$ 的通信复杂度。与 PBFT 类似，Zyzzyva 算法包含 agreement，view-change 和 checkpoint 3 个子机制，其中后两个机制与 PBFT 中的同名机制类似。agreement 机制如图 8 所示，在一个节点数为 $3f+1$ 的系统中，客户端给主节点发送请求，主节点给请求排序后广播给各个副官节点，各副官节点投机性执行后给客户端回复 responses。若客户端收到全部$3f+1$ 个一致的 responses，则认为系统达成一致，如图8(a)所示；若客户端收到的一致回复 responses 的数量在 $2f+1$ 到 $3f$ 之间，那么客户端搜集 responses 并广播 commit certificate 给副官节点，副官节点收到该消息后，如果认可该 commit certificate，则给客户端发送 commit certificate 的确认消息。一旦客户端收到 $2f+1$ 个 commit certificate 的确认消息，则认为系统达成一致，如图 8(b)所示；若足够的副本节点怀疑主节点是错误的，则会触发 view-change 协议，更换视图。

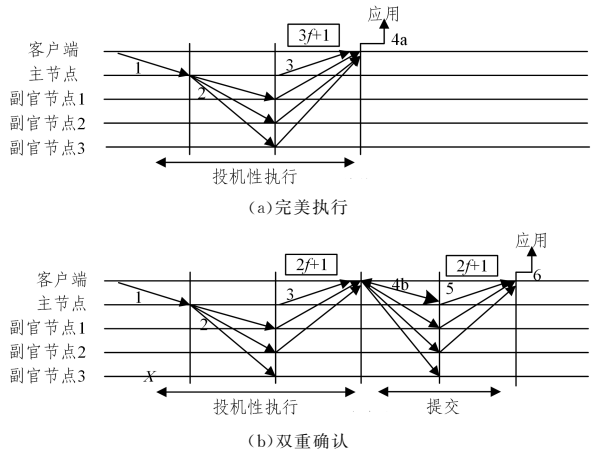

图 8 Zyzzyva 共识流程

Fig. 8 Zyzzyva consensus process

5.1.10 OBFT 共识机制

Wang 等于 2020 年提出了 OBFT 共识算法[48]，该算法在一定程度上解决了 PBFT 的主节点选取和通信耗时问题。针对主节点选取开销大的问题，OBFT 算法采用积分制选举主节点的方式对其加以改进，在保证选取的主节点一定符合最长链原则的基础上加速了选取进程，针对网络通信耗时的问题，OBFT 分别为同步和异步网络环境设计了不同的算法。首先执行针对同步模式下的乐观共识算法，若超时则立即切换为部分同步模式下的共识算法。此方法既保留了在良好网络状况下算法的性能，也确保了在极端网络状况下算法的安全性。OBFT 算法的流程如图 9 所示。相比 PBFT 和 OBFT 在吞吐量、交易时延和可扩展性等方面均有较大的提升。

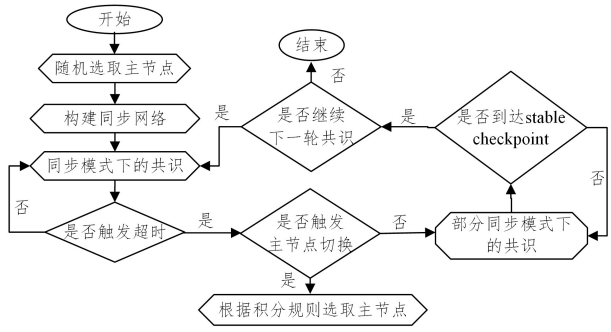

图 9 OBFT 共识流程

Fig. 9 OBFT consensus process

还有许多优秀的 BFT 共识算法，在此就不一一列举，本文根据 5 种优化思路对 BFT 算法的演进历史图进行了归纳总结，如图 10 所示，其中加粗的算法为上文详细梳理流程的算法。

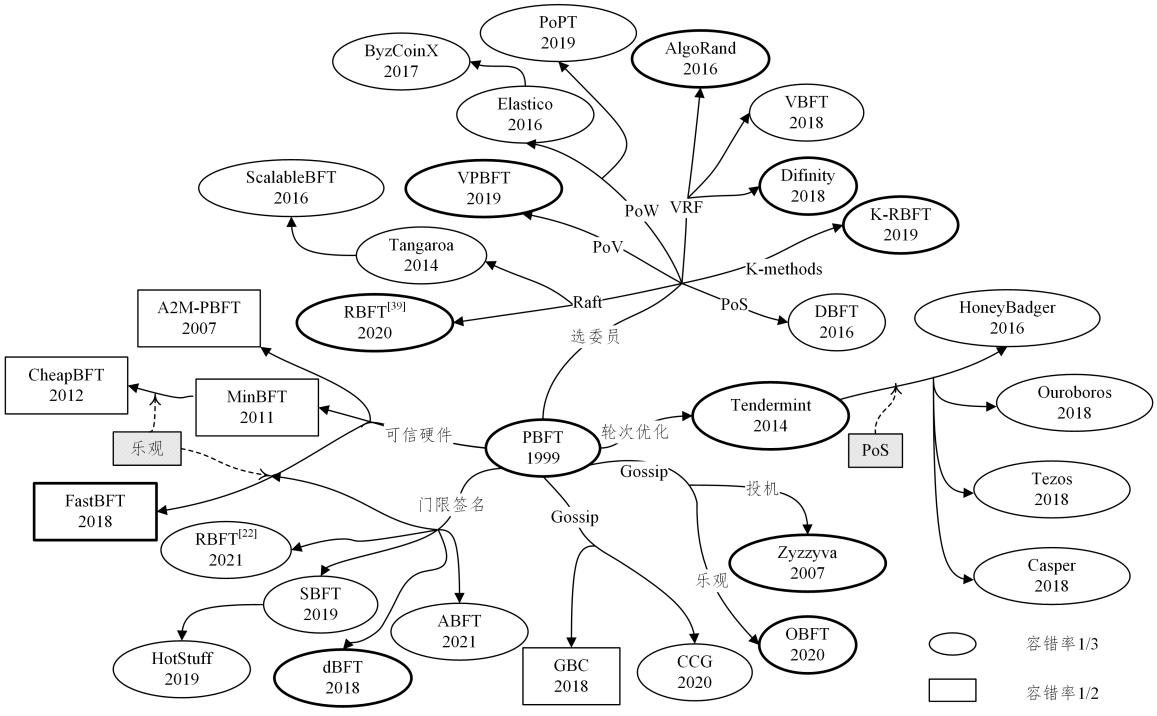

图 10 BFT 算法的演进历史图

Fig. 10 Development history of BFT algorithm

5.2 经典 BFT 共识算法对比

区块链系统的时间花费包括交易打包、加解密、消息传输、共识算法和交易上链等。通常共识算法的时间远长于其他时间消耗项，容易成为系统的性能瓶颈。共识算法的性能关乎整个区块链系统的性能。

表 2 列出了提到的各典型 PBFT 优化算法的效能和容错率，其中优化思路字段中的数字对应表 1 中的 5 种优化思路的编号。

表 2 经典 BFT 共识算法的性能比较

Table 2 Performance comparision of classical BFT consensus algorithm

名称	提出年份	吞吐量/TPS	容错率	生成区块时间	交易确认时间	代表性应用	优化思路
PBFT	1999	1 000	小于 1/3	秒级	1～2s	Hyperledger v0.6.0	—
dBFT	2016	1 000	小于 1/3	15～20s	立即确认	NEO	2
Tendermint	2014	1 000	小于 1/3	1s 左右	—	Monax	1,2,3
RBFT[53]	2021	3 000～4 000	—	—	<1	—	2
VPBFT	2019	—	小于 1/2	—	—	—	2
Dfinity	2018	1 000	小于 1/2	—	5～10s	Dfinity	2,3
Algorand	2016	875	小于 1/3	—	60s	ArcBlock	2
K-PBFT	2019	—	小于 1/3	—	小于 1s	—	2
FastBFT	2018	1 000	小于 1/2	—	3～4s	—	3,4,5
Zyzzava	2007	2 700	小于 1/3	—	1	—	5
OBFT	2020	1 400	小于 1/3	—	1	—	2,5

结束语 BFT 共识算法是区块链共识算法的核心，本文系统性地总结了区块链 BFT 共识算法的研究进展。在梳理 BFT 共识算法时，本文先提出了四大优化目标，再结合现有的区块链 BFT 共识算法总结了 5 种优化的思路，这 5 种优化思路可以叠加。以目前的研究现状来讲，对于 BFT 共识算法的优化趋势为：针对系统正常的时间比节点出错时间长的情况设计了投机或乐观算法，选举委员会提升了算法的可扩展性，同时使用密码学算法或可信硬件来保证算法的安全性。

参 考 文 献

[1] OKI B M, LISKOV B H. Viewstamped replication: A new primary copy method to support highly-available distributed systems[C]// Proceedings of the Seventh Annual ACM Symposium on Principles of distributed Computing. 1988: 8-17.

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

[3] PEASE M, SHOSTAK R, LAMPORT L. Reaching agreement

in the presence of faults[J]. Journal of the ACM (JACM), 1980, 27(2): 228-234.
[4] CASTRO M, LISKOV B. Practical byzantine fault tolerance [C]// OSDI. 1999, 99(1999): 173-186.
[5] KWON J. Tendermint: consensus without mining [EB/OL]. (2021-11-04) [2022-2-11]. https://tendermint.com/static/docs/tendermint.pdf.
[6] COPELAND C, ZHONG H. Tangaroa: a byzantine fault tolerant raft[EB/OL]. (2014-12-15) [2021-05-30]. https://www.scs.stanford.edu/14au-cs244b/labs/projects/copeland_zhong.pdf.
[7] MARTINO W. The first scalable, high performance private blockchain[EB/OL](2017-04-20)[2022-02-11]https://blockchainlab.com/pdf/Kadena-ConsensusWhitePaper-Aug2016.pdf.
[8] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system [EB/OL]. https://www.debr.io/article/21260.pdf.
[9] KIAYIAS A, RUSSELL A, DAVID B, et al. Ouroboros: A provably secure proof-of-stake blockchain protocol[C]// Annual International Cryptology Conference. Cham: Springer, 2017: 357-388.
[10] ZHONG Z S. An Improvement on Blockchain-Based PoS Consensus Algorithm[J]. Journal of Chongqing Technology and Business University(Natural Science Edition), 2021, 38(4): 36-41.
[11] ATENIESE G, BONACINA I, FAONIO A, et al. Proofs of space: When space is of the essence[C]// International Conference on Security and Cryptography for Networks. Cham: Springer, 2014: 538-557.
[12] BALL M, ROSEN A, SABIN M, et al. Proofs of Useful Work [EB/OL]. (2021-02-27) [2021-05-30]. https://eprint.iacr.org/2017/203.pdf.
[13] YUAN Y, NI X, ZENG S, et al. Blockchain consensus algorithms: the state of the art and future trends[J]. Acta Automatica Sinica, 2018, 44(11): 2011-2022.
[14] SCHWARTZ D, YOUNGS N, BRITTO A. The Ripple Protocol Consensus Algorithm[EB/OL]. https://ripple.com/files/ripple_consensus_whitepaper.pdf.
[15] MAZIERES D. The stellar consensus protocol: A federated model for internet-level consensus [J]. Stellar Development Foundation, 2015, 32: 1-45.
[16] CHURYUMOV A. Byteball: A decentralized system for storage and transfer of value [EB/OL]. https://byteball.org/Byteball.pdf.
[17] ROCKET T. Snowflake to avalanche: A novel metastable consensus protocol family for cryptocurrencies [EB/OL]. http://knowen-production.s3.amazonaws.com/uploads/attachment/file/1922/Snowflake%2Bto%2BAvalanche%2B-%2BA%2BNovel%2BMetastable%2BConsensus%2BProtocol%2BFamily.pdf.
[18] CHEN J W, XIAN X B, YANG Z G, et al. Improving the practical Byzantine fault-tolerant consensus algorithm with BLS aggregate signature[J/OL]. https://doi.org/10.19734/j.issn.1001-3695.2020.12.0403.
[19] POPOV S. The tangle [EB/OL]. http://www.descryptions.com/Iota.pdf.
[20] LI C, LI P, ZHOU D, et al. Scaling nakamoto consensus to thousands of transactions per second[EB/OL]. https://arxiv.org/pdf/1805.03870.pdf.
[21] HOPKINS A L, LALA J H, SMITH T, et al. The evolution of fault tolerant computing at the Charles Stark Draper Laboratory, 1955-85[M]// The Evolution of Fault-Tolerant Computing. Vienna: Springer, 1987: 121-140.
[22] DRISCOLL K, PAPADOPOULIS G, NELSON S, et al. Multimicroprocessor flight control system[C]// Proceedings of the IEEE/AIAA 5th Digital Avionics Systems Conference, Institute of Electrical and Electronic Engineers. New York, NY. 1983: 11.
[23] WENSLEY J H, LAMPORT L, GOLDBERG J, et al. SIFT: Design and analysis of a fault-tolerant computer for aircraft control [J]. Proceedings of the IEEE, 1978, 66(10): 1240-1255.
[24] FU X, WANG H, SHI P. A survey of Blockchain consensus algorithms: mechanism, design and applications[J]. Science China Information Sciences, 2021, 64(2): 1-15.
[25] WANG Q, YU J, PENG Z, et al. Security Analysis on dBFT protocol of NEO[C]// International Conference on Financial Cryptography and Data Security. Cham: Springer, 2020: 20-31.
[26] LUU L, NARAYANAN V, ZHENG C, et al. A secure sharding protocol for open blockchains[C]// Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 17-30.
[27] KOKORIS-KOGIAS E, JOVANOVIC P, GASSER L, et al. Omniledger: A secure, scale-out, decentralized ledger via sharding [C]// 2018 IEEE Symposium on Security and Privacy (SP). IEEE, 2018: 583-598.
[28] MILLER A, JUELS A, SHI E, et al. Permacoin: Repurposing bitcoin work for data preservation[C]// 2014 IEEE Symposium on Security and Privacy. IEEE, 2014: 475-490.
[29] PARK S, PIETRZAK K, ALWEN J, et al. Spacemint: A cryptocurrency based on proofs of space[C]// International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2018: 480-499.
[30] XIANG F, HUAIMIN W, PEICHANG S. Proof of previous transactions (PoPT): An efficient approach to consensus for JCLedger[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2019, 51(4): 2415-2424.
[31] WANG H Y, GUO K X, PAN Q Q. Byzantine fault-tolerant consensus algorithm based on voting mechanism[J]. Journal of Computer Application, 2019, 39(6): 1766-1771.
[32] HANKE T, MOVAHEDI M, WILLIAMS D. Dfinity technology overview series, consensus system[J]. arXiv: 1805.04548, 2018.
[33] THE ONTOLOGY TEAM. Ontology Launches VBFT, a Next-Generation Consensus Mechanism, Becoming one of the First VRF-Based Public Chains[EB/OL]. https://medium.com/ontologynetwork/ontology-launches-vbft-a-next-generation-consensus-mechanism-becoming-one-of-the-first-vrf-based-91f782308db4.
[34] GILAD Y, HEMO R, MICALI S, et al. Algorand: Scaling byzantine agreements for cryptocurrencies[C]// Proceedings of the 26th Symposium on Operating Systems Principles. 2017: 51-68.
[35] SUN H F, ZHANG W F, WANG X M. Byzantine fault-tolerant consensus algorithm for anti-adaptive attack based on threshold

and ring signature[J/OL]. Acta Automatica Sinica[2021-05-30]. https://kns.cnki.net/kcms/detail/11.2109.TP.20210308.1350.001.html.

[36] GUETA G G,ABRAHAM I,GROSSMAN S,et al. Sbft:a scalable and decentralized trust infrastructure[C]// 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE,2019:568-580.

[37] YIN M,MALKHI D,REITER M K,et al. Hotstuff:Bft consensus with linearity and responsiveness[C]// Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing. 2019:347-356.

[38] JALALZAI M M,NIU J,FENG C,et al. Fast-Hotstuff:a fast and resilient hotstuff protocol[EB/OL]. (2021-10-04) [2022-02-11]. https://arxiv.org/pdf/2010.11454.pdf.

[39] LI Q,XUE Z,ZHANG X. Improved Fast-HotStuff Blockchian Consensus Algorithm[J]. Computer Engineering,2021,47(8):14-21.

[40] ZHANG S J,CAI J,CHEN Z H,et al. Byzantine consensus algorithm based on Gossip protocol[J]. Computer Science,2018,45(2):20-24.

[41] ZHANG Q W,WANG Z Q,ZHANG Y Q. Research on trust collection consensus algorithm based on Gossip protocol[J]. Computer Science,2020,47(S1):391-394.

[42] CHUN B G,MANIATIS P,SHENKER S,et al. Attested append-only memory:Making adversaries stick to their word[J]. ACM SIGOPS Operating Systems Review,2007,41(6):189-204.

[43] VERONESE G S,CORREIA M,BESSANI A N,et al. Efficient byzantine fault-tolerance[J]. IEEE Transactions on Computers,2011,62(1):16-30.

[44] KAPITZA R,BEHL J,CACHIN C,et al. CheapBFT:Resource-efficient Byzantine fault tolerance[C]// Proceedings of the 7th ACM European Conference on Computer Systems. 2012:295-308.

[45] LIU J,LI W,KARAME G O,et al. Scalable byzantine consensus via hardware-assisted secret sharing[J]. IEEE Transactions on Computers,2018,68(1):139-151.

[46] KOTLA R,ALVISI L,DAHLIN M,et al. Zyzzyva:speculative byzantine fault tolerance[C]// Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles. 2007:45-58.

[47] DISTLER T,CACHIN C,KAPITZA R. Resource-efficient Byzantine fault tolerance[J]. IEEE Transactions on Computers,2015,65(9):2807-2819.

[48] WANG R H,ZHANG L F,XU Q Q,et al. Byzantine fault-tolerant consensus algorithm that can be applied to the alliance chain[J]. Application Research of Computers,2020,37(11):3382-3386.

[49] TIM F. Ethereum's Casper protocol explained in simple terms [EB/OL]. https://www.finder.com/ethereum-casper.

[50] DAVID B,GAŽI P,KIAYIAS A,et al. Ouroboros praos:An adaptively-secure,semi-synchronous proof-of-stake blockchain [C]// Annual International Conference on the Theory and Applications of Cryptographic Techniques. Cham:Springer,2018:66-98.

[51] GOODMAN L M. Tezos:A self-amending crypto-ledger position paper[EB/OL]. (2014-08-03) [2021-05-30]. https://icohoo.com/pdf/position_paper.pdf.

[52] MILLER A,XIA Y,CROMAN K,et al. The honey badger of BFT protocols[C]// Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. 2016:31-42.

[53] HUANG D Y,LI L,CHEN B,et al. RBFT:Byzantine fault-tolerant consensus mechanism based on Raft cluster[J]. Journal of Communications,2021,42(3):209-219.

[54] CHEN Z H,LI Q. Improved PBFT consensus mechanism based on K-medoids[J]. Computer Science,2019,46(12):101-107.

FENG Liao-liao, born in 1999, postgraduate, is a member of China Computer Federation. Her main research interests include blockchain consensus algorithm.

CHANG Jun-sheng, born in 1979, Ph.D, associate professor, is a member of China Computer Federation. His main research interests include high performance computing, high-speed interconnection network and blockchain.

(责任编辑:李亚辉)