优化可扩展的拜占庭容错共识算法

韩嗣诚，朱晓荣，张秀贤
（南京邮电大学通信与信息工程学院，江苏 南京 210003）

摘　要：区块链是一个去中心化的账本，可为交易中互不信任的双方提供信任，其最初作为支撑比特币的底层框架，近年来逐渐成为具有颠覆价值的新兴技术。共识算法是区块链的核心技术之一，没有共识算法就无法实现分布式节点间的状态一致。简单介绍了一种目前联盟链中常用的共识算法——实用拜占庭容错（PBFT, practical Byzantine fault tolerance）算法，并在其基础上优化算法机制，增加可扩展性，提出了一种改进的算法。经改进后，降低了算法的复杂度，并且允许共识节点加入和退出系统。仿真结果表明，改进后的算法可显著减少交易共识完成的时间和节点间的通信次数，从而在支持更多节点、减少系统通信开销和 CPU 计算资源消耗的同时，增大了整个系统的吞吐量。

关键词：区块链；共识算法；拜占庭容错；可扩展
中图分类号：TP391
文献标识码：A
doi: 10.11959/j.issn.2096−3750.2020.00166

Optimized scalable Byzantine fault tolerance algorithm

HAN Sicheng, ZHU Xiaorong, ZHANG Xiuxian
College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

Abstract: Blockchain is a decentralized ledger which provides trust to both parties which distrust each other in a transaction. Blockchain initially served as the underlying framework that underpins bitcoin and has increasingly become a disruptive new technology in recent years. Consensus algorithm is one of the core technologies of blockchain. Without a consensus algorithm, the state consistency among distributed nodes cannot be achieved. The practical Byzantine fault tolerance (PBFT) algorithm was briefly introduced, which was a commonly used consensus algorithm in consortium blockchain. An optimized and scalable algorithm based on it was proposed. The improved algorithm mainly reduced the algorithm complexity and allowed consensus nodes to join and exit the system. Simulation results show that the improved algorithm can significantly reduce the transaction consensus completion time and the number of communication times between nodes, so as to increase the throughput of the whole system while supporting more nodes and reducing the system communication overhead and CPU computing resource consumption.

Key words: blockchain, consensus algorithm, Byzantine fault tolerance, scalable

1　引言

区块链是一个由分布式对等网络、数据加密、共识算法等技术组合而成的去中心化账本或数据库。区块链运用技术手段为用户提供了对彼此的信任，使得他们可以在相互之间无信任基础的前提下完成交易，且其去中心化的存储方式配合共识机制使得交易的记录无法被篡改。区块链最初诞生于比

收稿日期：2020−03−16；**修回日期**：2020−04−17
通信作者：朱晓荣，xrzhu@njupt.edu.cn
基金项目：国家自然科学基金资助项目（No.61871237）；江苏省重点研发计划（No.BE2019017）；南京邮电大学教学改革研究项目（No.JG00218JX75）
Foundation Items: The Natural Science Foundation of China (No.61871237), The Key R&D Plan of Jiangsu Province (No.BE2019017), The Teaching Reform Research Project of Nanjing University of Posts and Telecommunications (No.JG00218JX75)

特币[1]中，作为服务交易的底层架构，逐渐扩展到各式各样的“数字货币”甚至金融业务中。随后，对架构和共识机制做了一定改变的联盟链被认为可以超越金融领域，用于数字资产认证、供应链溯源等商业化领域。

共识算法是区块链非常重要的一部分，也是其去中心化和信任机制建立的基础。具体来说，在分布式状态机构成的异步网络中，需要有一种办法保证每个状态机最终达成一个一致的状态，这种方法被称为共识算法。共识算法在区块链中的具体作用是保证每个节点中记录的区块信息以及区块中的交易或请求信息都是相同的，这样才能达到“共同维护”的效果。

比特币中的工作量证明（PoW, proof of work）是最早使用的共识算法，其核心是通过算力来竞争记账权，也就是创建新区块的权利。算力体现在给定一个字符串，在其后连接一个整数值串，然后对连接后的整个字符串进行 SHA256 哈希运算，如果运算后得到的字符串是以若干个 0 开头则验证通过，最先算出这个字符串的节点可以创建新区块，并得到一定数量比特币的奖励。这种算法非常耗费资源，除了最终胜出的节点，其他节点都在不停地计算却一无所获。以太坊出于对优化资源的考虑，设计了一种相对简单的共识算法，即权益证明（PoS, proof of stake）[2]。该机制引入了币龄的概念，通过一种与币龄相关的方式来决定创建新区块的权利，拥有币龄越大的节点获得区块创建权的可能性越高。

PoW 和 PoS 适用于参与节点众多的公有链，但却不适合节点数较少、节点间存在一定信任度的联盟链和私有链。联盟链和私有链常用的算法是基于消息传递的共识算法，如实用拜占庭容错（PBFT, practical Byzantine fault tolerance）算法[3]、Paxos 算法[4]和 Raft 算法[5]等。其中，PBFT 算法是一种主要用于联盟链的、可以解决拜占庭错误的算法，本文将在第 2 节进行介绍；而 Paxos 算法和基于其改进的 Raft 算法则主要用于不存在拜占庭错误的私有链中。Raft 算法不考虑拜占庭错误，节点仅可能因为故障而宕机，Raft 算法将节点分为集群，每个集群通常包含 5 个节点（服务器），允许最多有两个节点（服务器）发生故障。

除了上述已经成熟且投入应用的算法，近年也出现了一些新算法，如 Ripple 算法[6]和小蚁算法[7]等。区块链的高热度促使很多研究者对这些算法做了不同程度的改进。文献[8]将 PoW 和 PoS 结合，提出了一种双跳的共识算法。文献[9]提出了一种基于投票证明（proof of vote）的算法，为参与者建立不同的安全身份，再根据安全身份进行投票，来决定交易的提交和区块的创建。文献[10]提出了轻量化、动态化的 Raft 算法为 Pirogue 算法，其核心在于有节点发生故障后，剩余无故障节点可根据新的投票规则继续对交易进行共识，直到故障节点数量达到上限。文献[11]为 PBFT 算法增加了数据同步机制。

本文的主要贡献和成果如下。

1) 对 PBFT 算法进行改进，在确保共识机制可靠的前提下降低了算法复杂度，提升了共识效率，降低了计算资源消耗，提升了系统吞吐量。

2) 允许共识节点自由地加入和退出共识系统，并设计了加入和退出机制，使得算法能够更好地适用于实际的区块链系统。

2　PBFT 算法简介

PBFT 算法的关键在于解决拜占庭错误，拜占庭错误源于拜占庭将军问题，这是一个假想问题。其实质在于要寻找一种方法，使得将军们（其中存在叛徒，但不知道是谁）可以对某一个决策达成共识，但难点在于他们只能通过口头通信的方式来传递消息，详情可参考文献[12]。

PBFT 算法把将军们记为节点集群，假设系统总节点数为 n，其中有 f 个问题节点，问题节点可能会发送错误消息，同时另有 f 个节点可能因为网络问题而响应很长时间或不响应。其流程为：客户端发送一条请求，节点集群执行一个三阶段协议，然后客户端等待来自不同节点的 f+1 条相同的回复，该回复即为请求的执行结果。为了保证共识结果正确且客户端能收到 f+1 条相同回复，节点总数 n 应满足条件 $n>3f$，即 n 的最小值为 $3f$+1。假设在最坏的情况下，f 个善意节点在共识过程中失去响应，而 f 个错误节点都发送了相同的错误信息给客户端，但客户端还会收到剩余 f+1 个善意节点回复的 f+1 条相同的正确信息，所以仍可记录正确的共识结果。

2.1　正常情况流程

PBFT 算法的三阶段共识流程如图 1 所示，共识节点分为主节点（primary）和备份节点（backup）两种。图 1 中以 4 个节点为例，节点 C 是客户端，节点 0 是主节点，节点 1、节点 2 和节点 3 是备份

节点，其中，节点 3 是一个恶意节点。

当客户端向主节点发送一个请求后，算法正式开始执行。首先，主节点收到请求后会给该请求分配一个序号，并向所有节点广播 pre-prepare 消息。收到 pre-prepare 消息的节点对该消息的内容进行验证，若验证通过则向所有其他节点广播 prepare 消息。收到 prepare 消息的节点再对消息的内容进行验证，通过后则向节点广播 commit 消息。收到 commit 消息的节点将执行客户端发送的请求并给客户端回复。客户端等待 f+1 条相同的回复，并将该回复作为请求的执行结果。

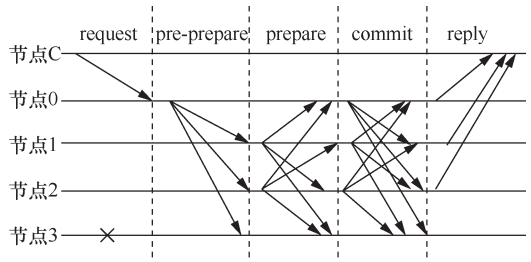

图 1　PBFT 算法的三阶段共识流程

2.2　视图更改与检查点

视图（view）是算法中的一个概念，每个节点在处理一次共识请求时，会有一个视图号，除非主节点变更，否则视图号不变。当主节点为恶意节点或发生故障时，系统会运行视图变更（view change）协议，接收协议的节点将变更视图号。此外，每执行完固定次数的请求后，系统将触发检查点（check-point），此时节点将会把已经执行完的消息从本地日志中删除。

3　算法改进

3.1　PBFT 算法的问题

从第 2 节的介绍可以看出，PBFT 算法可以很好地在存在拜占庭错误的分布式系统中达成共识，但其还存在一些问题，具体如下。

1) PBFT 算法本身并不是针对区块链设计的，所以将其用于区块链需要先做一定的修改。

2) PBFT 采用三阶段协议方式达成共识，在安全方面表现很好，但其通信算法复杂度非常高，时延很长，且 commit 阶段的存在只是为了确保在发生概率并不高的视图变更的情形下算法仍能继续运行，所以可以对此进行优化。

3) PBFT 算法是在封闭系统中运行的算法，并未考虑实际系统中可能出现节点数量变化的情况。

3.2　改进的共识机制

基于 3.1 节中 PBFT 算法存在的问题，本文对其做出了改进，提出了一种优化可扩展的拜占庭容错（OSBFT, optimized scalable Byzantine fault tolerance）算法。

3.2.1　数据共识

在本文场景下，假设客户端有一个专门的数据处理节点（SC）负责生成交易单并存储一份节点清单，数据处理节点不会有恶意行为。此外，假设恶意节点只能在共识阶段发送错误的验证结果（将有效的交易判为无效或将无效的交易判为有效）给其他节点以及存储无效的交易信息或不存储有效的交易信息。

交易数据共识过程如图 2 所示，其中，节点 SC 是数据处理节点，节点 P 是主节点，节点 3 是一个恶意节点。从 inform 阶段开始，数据处理节点 SC 向主节点 P 发送一条 inform 消息，格式为<record, d, t>。其中，d 是完整的需要记录在数据库中的、具有标准格式的交易数据。d 包含交易双方的地址、交易的频谱数量、租用价格、使用时间等信息，如果交易双方使用了智能合约，那么智能合约也会被附在其中。t 代表时间戳，用一个字符序列的方式唯一地标识某个时间，该时间为交易双方确认交易的时间，也就是交易生成时间。record 是记录命令，代表数据处理节点要求主节点记录该交易数据。

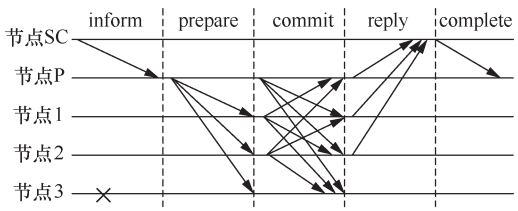

图 2　交易数据共识过程

主节点 P 收到 inform 消息后，先给该笔交易加上序号。为了简化共识流程、降低时延，给交易序号分配设定了一个规则。首先，设定区块的生成依据交易的数量而非时间，即每当通过共识成功记录 k 笔交易后，这 k 笔交易将被装进一个新生成的区块中，这样每个区块都有且只有 k 笔交易信息。在此条件下，每笔交易的序号将按照区块号-交易号的形式分配。这个规则是预先写在每个节点中的，即便主节点不分配序号，每个节点也都知道下一笔待验证交易的序号。该方式使得节点不需要通过共识来对序号达成一致。主节点 P 分配好序号后将发送

prepare 消息给所有备份节点，消息格式为<prepare, *b-m*, *d*, *t*>。*d* 和 *t* 与 inform 消息中一样，代表交易数据和时间戳。需要注意的是，如果主节点同时收到多笔交易（因网络时延问题）或者未来得及为某一笔交易广播 inform 消息就已收到了新的交易，则将根据时间戳对交易排序，先广播更早的那一笔交易，其余交易暂时放入缓存中，等待后续广播。*b-m* 是序号的格式，如 23-3。*b* 是待生成的新区块的区块号，其值为数据库中已存储的最新区块的区块号+1。*m* 则是区块 *b* 中的交易号，满足 $1 \leqslant m \leqslant k$，由于是分布式存储，每个节点都有完整的当前区块链信息，所以每个节点都可以依据序号规则判断序号是否正确。

除了主节点自身外，每个备份节点都将收到 prepare 消息。节点收到 prepare 消息后将进行判断，若满足以下条件，则备份节点将接收该 prepare 消息。

1) *d* 附带的签名是正确的；

2) 区块号确实是本地数据库中最新区块的区块号+1，交易号是上一笔验证成功交易的交易号+1，且不超过 *k*；

3) 节点未收到具有相同的交易数据 *d* 和时间戳 *t*，但序号不同的交易。

接收 prepare 消息后，节点将验证该笔交易是否有效（主要用于支付的虚拟货币来源是否合规、是否存在“双花”行为以及交易中的频谱是否可用、是否已被交易等）。验证后节点将发送 commit 消息通知所有其他节点自身的验证结果，消息格式为<commit, *n*, *d*, *t*, valid/invalid, *i*>。其中，*d* 和 *t* 仍是数据和时间戳，*n* 是序号。valid 和 invalid 是发送该消息的节点对交易的验证结果，消息中仅会出现 valid 或 invalid，不会同时出现两种结果。最后 *i* 是该节点的节点编号，每一个参与共识的节点都会有一个唯一且不变的编号。节点必定会接收 commit 消息，如果某个节点发送的 commit 消息中出现序号错误，则该节点将直接被认为是恶意节点，接收消息的节点不会将恶意节点的验证结果纳入考虑，主节点将在所有 prepare 消息发送完成后发送 commit 消息。

善意节点接收 commit 消息后，将综合其他节点的判断与自身的判断来做出最终决定。因为善意节点对交易是否有效的判断一定是正确的，所以当善意节点收到至少 f 条与自身判断结果相同的 commit 消息后，便会认为该交易已得到了多数节点的认可，且验证结果一定是正确的。这样，如果恶意节点从攻击中恢复，那么其将收到至少 f+1 条带正确结果的消息和最多 f−1 条带错误结果的消息，在最终回复时仍然可以回复正确的消息并将正确的信息记录至数据库。节点做出最终判断后，将发送一条 reply 消息给数据处理节点 SC，格式为<reply, *n*, valid/invalid, *i*>。该消息构成简单，只有交易序号、结果和自身的节点编号。如果交易有效，则节点将把该笔交易放入缓存池中，等待区块生成时装入区块并最终写入数据库；如果交易无效，那么将只发送信息通知数据处理节点 SC，不会将交易放入缓存池中。

数据处理节点负责最终接收 reply 消息，它将等待至少 f+1 条带有相同结果的消息，并将此结果作为最终结果。如果交易有效，则它将交易序号、验证有效和前 f+1 个发送结果为有效的 reply 消息的节点编号一并存入本地数据库（数据处理节点并不作为区块链分布式数据库的一部分）；如果交易无效，它将发送消息通知交易双方该笔交易无效，需重新交易。最后它将发送<consensus complete, *n*>消息给主节点，*n* 是交易编号。当且仅当主节点收到该消息后，才会就下一笔交易发送 prepare 消息给备份节点。因为算法取消了视图机制，所以用此方法保证了只有上一笔交易共识完成后才会开始进行下一笔交易共识。

3.2.2　新区块创建与数据同步机制

每当完成 *k* 笔交易的共识之后，主节点将负责创建新区块，把之前完成的交易信息存入新区块中，并向其他共识节点广播新的区块。新区块从主节点创建到最终记录在所有节点的数据库的过程还包含了数据同步机制，以确保最终每个节点记录的区块信息和交易信息无误。

当数据处理节点发送完第 *k* 笔交易的共识完成消息后，它将向所有节点发送数据确认消息，内容为<data-confirm, D>。其中，D 是交易数据构成的表格，记录了新区块需存入的所有交易的编号和加密的交易内容摘要。节点收到消息后将对比 D 和本地数据库记录的信息，若两者相同，则发送一条<confirmed, right, *i*>消息表明数据一致，*i* 是节点编号；若两者不同，则发送<confirmed, wrong, *i*, *N*>消息，*N* 是所有不相同交易的编号构成的集合。对于回复 wrong 消息的节点，数据处理节点将回复相应编号的交易的完整数据信息，节点将信息更新完成后再回复数据处理节点。

数据处理节点收到所有之前发送 wrong 消息的节点的更新完成回复后，将通知主节点开始创建新区块，新区块创建流程如图 3 所示。数据处理节点首先发送一条 inform 消息，消息内容为<start, new block>。主节点收到该消息后开始创建新区块，并将所有交易存入新的区块中。之后主节点将发送消息通知所有其他共识节点记录新的区块，消息格式为<prepare, *n*, *t*, *b*>，*n* 是区块号，*t* 是一个时间戳，表示区块创建的时间，*b* 是区块内容。其他共识节点接收这条 prepare 消息后先进行判断，主要是验证区块编号是否正确以及区块头中的内容是否包含正确的上一区块的哈希值。当这两项都验证通过后，节点将区块信息放入缓存，并发送一条 commit 消息给其他共识节点，消息格式为<commit, *n*, *t*, *i*, r/w>，r 表示区块信息正确，w 表示错误。节点收到 commit 消息后，将参考其他节点的判断，若某一结果超过半数，则判定其为正确结果。在区块共识阶段，节点将断开与外网的连接以防止攻击并保证正确记录区块信息，所以采用收到超过半数的相同结果则做出决定的方式。

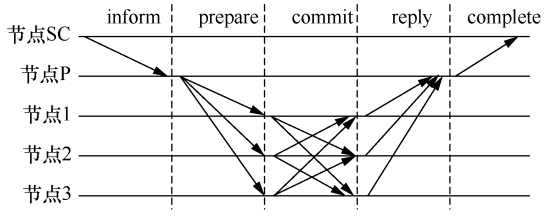

图 3 新区块创建流程

节点做出判断后将回复主节点消息，如果区块信息无误，则将本地缓存中的交易存入区块并把区块加入本地区块链，然后回复<added, *i*>；如果区块信息不正确，则回复<err-block, *i*>。主节点等待所有共识节点的回复，并将多数结果作为最终结果（理论上来说结果应是一致的）。如果区块信息错误，那么主节点将删除缓存中的区块数据，重新创建区块并重复流程；如果区块信息正确，主节点将把区块加入本地区块链，并回复数据处理节点区块创建成功，消息为<success built, *b*>，*b* 为区块号。若经过 t_0 时间仍有节点未回复或部分节点回复的结果不正确，则主节点将发送命令要求这些节点将区块加入本地区块链中，在得到这些节点添加成功的回复后，再告知数据处理节点创建成功。数据处理节点收到 success built 消息后完成区块创建，可以开始向新的主节点发送新的交易数据。

3.2.3 主节点更换

在正常情况下，一个主节点将负责一个新区块的创建、共识以及该区块内包含的所有交易的共识过程，直到新区块成功创建才会更换主节点以负责下一个区块事宜。但若主节点在共识过程中长时间不响应或主节点需要退出，为了保证系统运行就需要更换主节点。在本算法中，由于序号是固定不变的，主节点为恶意节点时仅能发送错误结果。但该行为并不会影响最终共识结果，所以即便主节点为恶意节点，也不需要立即更换。由此可知，只有在主节点发生故障导致长时间不响应时，才需要发起主节点更换。

主节点更换过程如图 4 所示，图 4 中节点 P 是原主节点，P_{new} 是新的主节点。假定数据处理节点在向主节点 P 发送一笔交易后经过 t_1 时间都未等到一条关于该交易的 reply 消息，则判定主节点发生故障，需要更换。此时数据处理节点将发送一条<be primary, *b*, *l*>消息给新的主节点 P_{new}。其中，*b* 是新的主节点负责的区块号，也就是当前交易所在的区块，*l* 是该区块剩余的交易数。

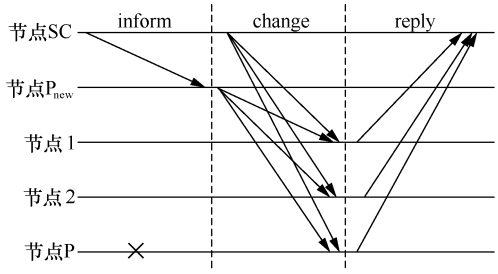

图 4 主节点更换过程

如果在区块 *b* 的第 *m* 笔交易共识时主节点发生故障，则 *l* 的值为 *k*−*m*。由于这是联盟链，由多个参与方共同维护，所以新的主节点将选择与当前故障主节点不同的参与方节点。之后数据处理节点将直接发送格式为<change primary, *p*, p_{new}, *b*, *l*>的消息给所有节点，同时新的主节点也会发送一条格式为<new primary, p_{new}, *b*, *l*>的消息给其他共识节点。其中，*p* 是原主节点的节点编号，p_{new} 是新主节点的节点编号，收到这样两条消息并核对 p_{new}、*b*、*l* 一致的节点将回复一条格式为<get, p_{new}>的消息给数据处理节点。数据处理节点收到所有节点回复后或经过 t_2 时间后，将给新的主节点发送 inform 消息。

3.3 节点加入和退出机制

节点加入和退出机制是对共识算法流程的补

充机制，也是实际应用中应该考虑的一个问题。为了保证节点可以退出共识且系统不受影响，应保证有节点退出后仍然可以容忍 f 个恶意节点。因为恶意节点是由网络攻击导致的，所以 f 不是一个固定值。根据系统安全性能评估可知受到外界强烈攻击时最多会出现的恶意节点数量，并用 $f_{\max}$ 表示，f 的值从 0 到 $f_{\max}$ 的概率是逐渐降低的。即便如此，系统仍需保持至少 $3f_{\max}+1$ 个共识节点，这就要求多数时候都要有大于 $3f_{\max}+1$ 个节点参与共识。

3.3.1　节点退出机制

节点退出机制如图 5 所示，节点 i 为待退出的节点。退出机制共分为 request（请求）、reply（回复）和 confirm（确认）3 个阶段。首先，在 request 阶段，需要退出的节点（编号记为 i）向数据处理节点发送退出请求，消息格式为<quit-req, i, b, m>，b 和 m 分别是区块号和交易号，意在通知数据处理节点自身退出的时间（在交易 b-m 共识完成后退出）。因为不允许在共识过程中退出，所以节点需要声明其在哪笔交易后退出，也就是退出时间。在 reply 阶段，数据处理节点收到 request 阶段节点 i 发送的退出请求后将回复节点 i 一条消息，同意则为<agree-quit>，不同意则为<refuse-quit>。在正常情况下，除非该节点退出将导致系统总节点数少于 $3f_{\max}+1$ 或退出后该节点所属联盟参与方无共识节点在用，否则不会拒绝节点退出。若数据处理节点同意该节点退出，则在发送<agree-quit>消息后向所有节点广播一条<quit, i, b, m>消息。同时，退出节点 i 在收到 agree 消息后，也将向所有节点广播相同的<quit, i, b, m>消息，此为 confirm 阶段。当其他节点收到来自数据处理节点和节点 i 的两条相同（节点编号、区块号和交易号一致）的退出消息后，判定该节点将在交易 b-m 后退出，从交易 b-(m+1) 开始将不再给该节点发送 prepare 和 commit 消息。

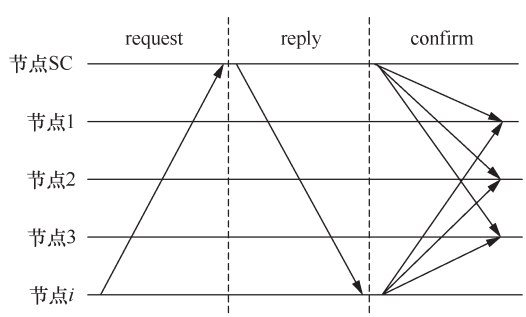

图 5　节点退出机制

3.3.2　节点加入机制

节点加入机制如图 6 所示，节点 j 为待加入的节点。加入机制需要节点间相互确认，比退出机制复杂，主要是考虑联盟链对加入者有一定要求，不能采用公有链的随时、自由加入/退出机制。加入机制分为 4 个阶段，分别是请求（request）、确认（confirm）、回复（reply）和完成（complete）阶段。

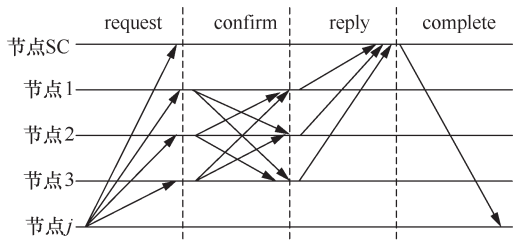

图 6　节点加入机制

从 request 阶段开始，待加入的节点向所有共识节点以及数据处理节点都发送一条加入请求消息。消息格式为<join-req, ip, party>，ip 表示待加入节点的 IP 地址，用于其他节点向其发送消息，party 是节点所属联盟参与方，在发送给数据处理节点的消息中需要额外附上防攻击能力等安全信息。之后是 confirm 阶段，所有收到加入请求的共识节点将向其他共识节点发送消息，以确认是否其他节点也收到了新节点的加入请求，消息格式为<new-join, ip, i>。其中，ip 是待加入节点的 IP 地址，i 是发送消息的节点自身的节点编号。发送完消息后，每个节点将同时接收 new-join 消息，当接收到至少 $2f$+1 条消息后，进入 reply 阶段。收到 $2f$+1 条 new-join 消息表明大多数节点都已收到该消息，可判定确实有新节点要加入，且收到消息的善意节点数量比恶意节点多。

进入 reply 阶段后，每个节点将给数据处理节点回复并表明是否同意新节点加入，消息格式为<reply, ip, agree/refuse>。同意或拒绝的决策者实际上是节点所属的联盟参与方，参与方将根据自身的利益和联盟规则决定是否同意新节点加入。数据处理节点收集所有节点的回复并最终判定其安全性能，只有超过半数节点同意新节点加入且其安全性能合格，数据处理节点才会最终同意其加入并在节点列表中添加相关信息。如果从节点 j 向数据处理节点发起申请起经过 t_1 时间，数据处理节点仍未收集到所有节点的意见，则它将根据已回复的结果做出决定，将未回复节点视为放弃。最后数据处理

节点将发送一条消息给待加入的节点，如果同意加入，则消息为<agree-join, *j*>，其中，*j* 是为待加入节点分配的节点编号（编号规则为递增，且节点退出后其编号将保留，不会被新节点占用）；如果不同意加入，则消息为<refuse-join>。同样地，数据处理节点也会给其他所有共识节点发送消息，通知其最终结果。如果同意，则下一笔交易开始时新节点便可作为共识节点参与共识。在新节点加入前，还需同步所有过往区块和交易信息。

4　性能分析

本节对提出的 OSBFT 算法进行性能分析，并与 PBFT 算法对比。假定到达主节点的交易数量和节点处理消息的时间均呈指数分布，其均值约为 1.1 ms，方差约为 1.23 ms。消息在链路中的传输时间与消息所含数据量大小有关，还可能存在额外时延。本文取节点数量分别为 4 个、7 个、10 个、13 个、19 个、25 个、31 个和 40 个，并研究在这些节点数量下的算法性能。

在没有错误节点的情况下，OSBFT 算法与 PBFT 算法的共识时间对比如图 7 所示。由图 7 可以看出，两种算法的共识时间都会随着节点数量的增加而增加，且增加速度也越来越快。这是因为节点数量越多，则确认一笔交易所需发送的消息数量就越多，自然更耗时间。在节点数量较少的情况下，两种算法的共识完成时间相差不大，但当节点的数量逐渐增加，尤其是超过 20 个节点以后，OSBFT 算法的共识时间相较 PBFT 算法显著减少，证明减少共识步骤的改进算法达到了预期的效果。

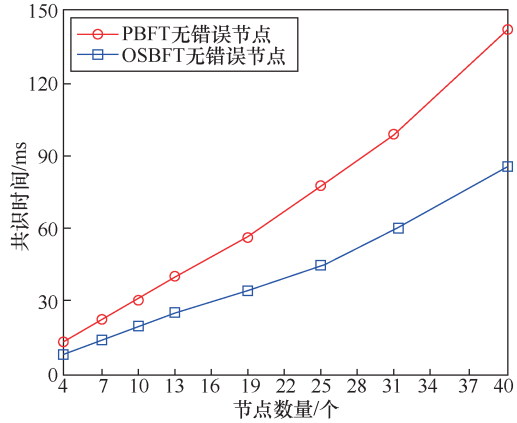

图 7　OSBFT 算法与 PBFT 算法的共识时间对比（无错误节点）

在 *n* 个节点系统中存在(*n*−1)/3 个错误节点的情况下，OSBFT 算法与 PBFT 算法的共识时间对比如图 8 所示。OSBFT 算法的时延与无错误节点时相同，都优于 PBFT 算法，但两种算法的共识完成时间都多于不存在错误节点的情况。这是因为错误节点可能会延迟发送消息使节点通信时间更长或者发送错误消息增加节点处理时间，从而导致交易确认得更慢。无错误节点越多，则消息的有效性更高，消息利用率也更高。在图 8 中，系统拥有 40 个共识节点，OSBFT 算法比 PBFT 算法在单笔交易确认时间上减少约 80 ms，这表明 OSBFT 算法可支持更快的交易到达速度，也就是单位时间内可容纳更多的交易数量，从而能够提升整个系统的吞吐量。

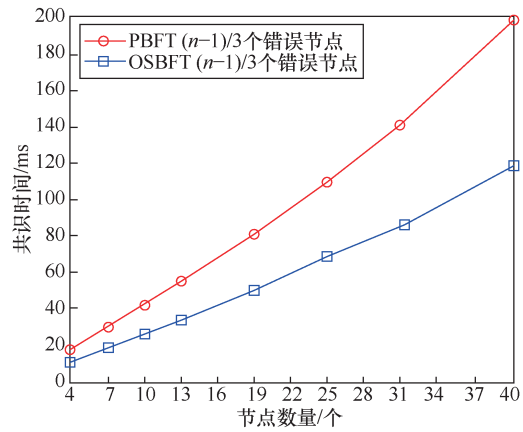

图 8　OSBFT 算法与 PBFT 算法的共识时间对比（系统有(*n*−1)/3 个错误节点）

在不同节点数量下，节点通信次数如图 9 所示。其值为假设 *n*=3*f*+1 个共识节点中存在 *f* 个拜占庭错误节点，在错误节点不会向其他节点发送消息且最终数据处理节点只收到 *f*+1 条回复消息的情况下，完成共识所需的节点间互发消息数。

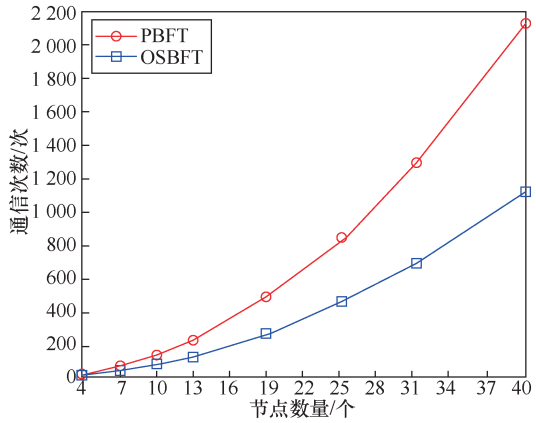

图 9　节点通信次数

从图 9 可以看出，PBFT 算法的通信次数随节点数量的增加而增加，OSBFT 算法的增加速度相对

慢很多，通信次数直接关系着部署算法所需的计算资源。减少一个共识步骤的优化方法可以有效节省计算资源，降低服务器负载，同时增加算法可容纳的共识节点数量。

5　结束语

本文介绍了常用的共识算法，从联盟链目前相对成熟的 PBFT 算法着手研究，对其进行改进，提出了 OSBFT 算法。OSBFT 算法减少了共识步骤，增加了节点加入和退出机制，使得共识节点数量可变化。最后，本文对 OSBFT 算法进行性能分析。分析结果表明，改进后的算法可显著减少交易共识完成的时间和节点间的通信次数，从而在支持更多节点、减少系统通信开销和 CPU 计算资源消耗的同时，增大了整个系统的吞吐量。

参考文献：

[1] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system[R]. 2019.

[2] KING S, NADAL S. PPCoin: peer-to-peer crypto-currency with proof-of-stake[R]. 2012.

[3] CASTRO M, LISKOV B. Practical Byzantine fault tolerance[C]// OSDI. 1999, 99(1999): 173-186.

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

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

[6] SCHWARTZ D, YOUNGS N, BRITTO A. The ripple protocol consensus algorithm[R]. 2014.

[7] Onchain. 小蚁白皮书[S]. 2015.

[8] DUONG T, FAN L, ZHOU H S. 2-hop blockchain: combining proof-of-work and proof-of-stake securely[EB/OL]. 2017.

[9] LI K, LI H, HOU H, et al. Proof of vote: a high-performance consensus protocol based on vote mechanism & consortium blockchain[C]// 2017 IEEE 19th International Conference on High Performance Computing and Communications. IEEE, 2017: 466-473.

[10] PÂRIS J F, LONG D D E. Pirogue, a lighter dynamic version of the raft distributed consensus algorithm[C]//2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC). IEEE, 2015: 1-8.

[11] JIANG Y, DING S. A high performance consensus algorithm for consortium blockchain[C]//2018 IEEE 4th International Conference on Computer and Communications (ICCC). IEEE, 2018: 2379-2386.

[12] LAMPORT L, SHOSTAK R, PEASE M. The Byzantine generals problem[M]//MALKHI D. Concurrency: the Works of Leslie Lamport. New York: ACM, 2019.

[作者简介]

韩嗣诚（1995− ），男，江苏南京人，南京邮电大学通信与信息工程学院硕士生，主要研究方向为区块链以及基于区块链的频谱共享等。

朱晓荣（1977− ），女，山东临沂人，南京邮电大学教授、博士生导师，主要研究方向为 5G/6G 移动通信、物联网、区块链、人工智能等。

张秀贤（1982− ），女，江苏南京人，南京邮电大学通信与信息工程学院博士生，主要研究方为区块链、边缘计算、人工智能等。