计算机科学
COMPUTER SCIENCE

**基于联盟链的实用拜占庭容错算法的改进**

谢卓，张志鸿，李磊，冯英杰，陈静

引用本文

谢卓，张志鸿，李磊，冯英杰，陈静. 基于联盟链的实用拜占庭容错算法的改进[J]. 计算机科学, 2022, 49(11): 360-367.

XIE Zhuo, ZHANG Zhi-hong, LI Lei, FENG Ying-jie, CHEN Jing. Improvement of PBFT Algorithm Based on Consortium Blockchain[J]. Computer Science, 2022, 49(11): 360-367.

---

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210900178

# 基于联盟链的实用拜占庭容错算法的改进

谢 卓 张志鸿 李 磊 冯英杰 陈 静
郑州大学信息工程学院 郑州 450000
(zhuoxiexz@163.com)

**摘 要** 作为一种新兴技术,区块链从诞生之初就引起了广泛的关注。共识算法是区块链技术的核心技术之一,共识算法的研究也是区块链发展的重中之重。针对广泛应用于联盟链的实用拜占庭容错算法(PBFT)存在的主节点选取随意以及节点无法动态加入、退出的问题,提出了一种动态的 PBFT 算法——DPBFT。首先,对 PBFT 的主节点选取方法进行改进,为每个节点设置信任度积分,根据节点在每轮共识中的行为动态更新信任度积分,依据积分值来选取主节点,提高了诚实节点当选主节点的概率。其次,为 PBFT 算法设置 4 个子协议(JOIN,EXIT,PCLEAR,RCLEAR),分别解决节点加入、退出的问题以及对作恶节点做出惩罚,使得系统拥有动态的网络结构。结果证明新加入的 4 个子协议本身具有良好的安全性和活性,且不影响原始 PBFT 算法的安全性和活性。最后,实验结果表明,DPBFT 算法相比传统 PBFT 算法具有更好的共识效率。

**关键词**:联盟链;共识算法;拜占庭容错;信任度;主节点选取
**中图法分类号** TP311

## Improvement of PBFT Algorithm Based on Consortium Blockchain

XIE Zhuo,ZHANG Zhi-hong,LI Lei,FENG Ying-jie and CHEN Jing
School of Information Engineering,Zhengzhou University,Zhengzhou 450000,China

**Abstract** As a new technology,blockchain has attracted the attention of all walks of life since its birth. Consensus algorithm is one of the core technologies of blockchain technology,and the research of consensus algorithms is also the top priority of blockchain development. Aiming at the problems of PBFT,which is widely used in consortium blockchain,such as the random selection of primary node and the inability of nodes to join and exit dynamically,an dynamic PBFT algorithm(DPBFT) is proposed. Firstly,the selection method of primary node of PBFT is improved,and the trust score is set for every node. The trust score is dynamically updated according to the behavior of the nodes in every round of consensus,and the primary node is selected according to the integral value,which improves the probability that an honest node is elected as the primary node. Secondly,four sub-protocols (JOIN,EXIT,PCLEAR,RCLEAR) are set for PBFT algorithm to solve the problem of joining and exiting nodes respectively and punish the offending nodes,so that the system has a dynamic network structure. It can be proved that the newly added four sub-protocols have good safety and liveness without affecting the safety and liveness of the original PBFT algorithm. At last,experimental results show that DPBFT algorithm has better consensus efficiency than traditional PBFT algorithm.

**Keywords** Consortium blockchain,Consensus algorithm,Byzantine fault tolerant,Trust,Selection of primary node

## 1 引言

2008 年,日本学者中本聪首次提出比特币的概念[1],作为比特币的底层技术,区块链也得到了越来越多的重视与发展。区块链技术是以数据库为数据存储载体,以 P2P 网络为通信载体,依赖密码学确定所有权和保障隐私,依赖分布式系统共识机制保障一致性,旨在构建价值交换系统的技术[2],具有去中心化、可追溯、不可篡改和公开透明等特性。

从去中心化的角度,区块链可分为公有链、联盟链和私有链。联盟链的去中心化程度介于公有链和私有链之间,一般由有多个不同的利益方的机构组成[3]。节点必须经过联盟链节点成员管理服务进行身份确认和鉴权,获得准入资格之后才可以加入联盟链网络。

在基于分布式系统的 P2P 网络中,去中心化思想需要一种新的方法来使所有的节点达成共识,共识机制便应运而生[4]。根据系统能否解决拜占庭错误可将共识协议分为崩溃容错协议(Crash Fault Tolerance,CFT)和拜占庭容错协议(Byzantine Fault Tolerance,BFT)两大类[5]。CFT 类共识

到稿日期:2021-09-22 返修日期:2022-02-22
基金项目:河南省重大公益专项(201300210300);郑州大学教育教学改革研究与实践项目(2021ZZUJGLX132)
This work was supported by the Major Public Welfare Project of Henan Province(201300210300) and Research and Practice Project of Education and Teaching Reform in Zhengzhou University(2021ZZUJGLX132).
通信作者:李磊(ielilei@zzu.edu.cn)

算法无法应对拜占庭错误，主要应用于私有链中；BFT 类共识算法可以应对拜占庭错误，因此被广泛应用于公有链和联盟链中。BFT 类算法主要有工作量证明算法（Proof of Work，POW）[1]、权益证明算法（Proof of Stake，POS）[6]、委托权益证明算法（Delegated Proof of Stake，DPoS）[7]以及实用拜占庭容错算法（Practical Byzantine Fault Tolerance，PBFT）[8]等。

本文以目前应用广泛的联盟链为研究背景，在 PBFT 算法的基础上，提出了一种信任积分机制，增加了主节点的可信度。同时添加了 4 个子协议，在不破坏原始 PBFT 算法安全性和活性的前提下，实现了 PBFT 算法中共识节点的动态加入和退出，并实现了对作恶节点的惩罚。

## 2　相关工作

### 2.1　PBFT 算法简介

1998 年，Castro 等[8]提出多项式级别算法复杂度的实用拜占庭容错算法 PBFT，该算法在保留 BFT 算法优点的同时显著降低了 BFT 协议的开销，可以实际应用于联盟链中。PBFT 算法每次共识均发生在一个视图中，视图是连续编号的整数，每个视图对应一个主节点，其余节点都是副本节点。PBFT 的敌手模型为 $n=3f+1$（$f$ 是恶意节点的数量），网络模型为部分同步网络。客户端向共识网络发送 request 请求，共识网络达成共识之后再向客户端返回消息，如图 1 所示。

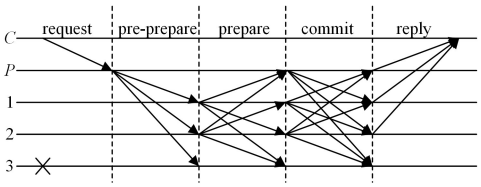


图 1　PBFT 算法流程图

Fig. 1　Process diagram of PBFT

request 阶段：客户端节点向主节点发送请求 $m$。

pre-prepare 阶段：主节点接收到来自客户端的请求后，给该请求分配一个序列号 $n$，生成 pre-prepare 消息并向副本节点广播。

prepare 阶段：副本节点对 pre-prepare 消息进行验证，若验证通过，则向其他节点广播 prepare 消息。

commit 阶段：节点收到来自其他副本节点的 prepare 消息时，对消息内容进行验证，若验证通过则向所有节点发送 commit 消息。

reply 阶段：确认阶段完成后，节点向客户端回复 reply 消息，当客户端接收到 $f+1$ 个不同的节点发来的有效消息时，reply 阶段完成。

同时，PBFT 通过视图转换方式来替换失效的主节点以保证共识的持续进行，为算法提供活性。当主节点出现问题不能及时处理数据请求时，其他副本节点会发起视图转换，转换成功后新的主节点开始工作，接着进入下一个视图，开始新一轮的三阶段共识流程。主节点以轮次交换的方式进行选取。

### 2.2　PBFT 算法的改进研究

目前 PBFT 算法还存在很多不足。随着大规模区块链系统的发展和兴起，研究人员开始着眼于简化 PBFT 协议，优化拜占庭协议的性能，从而提高容忍数据产生的多种错误的能力。当前 PBFT 算法存在的主要问题有以下几方面：

（1）当参与共识的节点数量增加时，PBFT 算法共识交互需求急剧增加，会导致网络拥塞，影响共识效率；

（2）主节点选取过于随意，很有可能选取出恶意节点，造成主节点频繁切换，影响共识效率；

（3）没有实现节点动态加入或者退出的机制，同时也没有对作恶节点进行惩罚，节点作恶成本过低。

目前 PBFT 算法的改进目标主要有以下几方面：

（1）改进 PBFT 算法过程或优化网络节点结构，减少网络中的信息交互次数，降低通信复杂度；

（2）优化主节点选取方法，增加诚实节点当选主节点的概率，降低主节点作恶的可能性，避免因为主节点作恶而共识失败；

（3）对 PBFT 协议进行改进，使其能够实现节点的动态加入和退出。

由于 PBFT 在拜占庭容错算法中具有非常重要的地位，因此近年来国内外针对 PBFT 算法的改进研究明显增多。Zhan 等[9]为了进一步提高共识过程的效率和可靠性，在 PBFT 的基础上提出了一种基于委托机制的随机化拜占庭容错共识协议 DRBFT，即提出一种新的随机算法来改进投票机制，有效减少参与共识的节点的数量。Yu 等[10]基于 PBFT 共识算法，在节点的组织和属性上修改算法模型，提出了一种基于置信度的动态分组拜占庭容错机制 DGBFT。Li 等[11]提出了一种可扩展的多层 PBFT 协议，然而，增加层数的方式会导致每一层的节点数量减少，削弱了系统的安全性能，同时增加了通信时延。为了解决可扩展性的问题，Jalalzai 等[12]基于与一组窗口节点通信的机制，提出一种 Musch 协议，该协议无须知道系统中恶意节点的实际值，就能够自动调整到每个时期表现出错误行为的节点的实际数量。Wang 等[13]提出了一种优化拜占庭容错算法来解决联盟链的性能瓶颈，其根据最新的局部节点，用块高来选举领袖节点并结合动态节点列表来替换上一轮的共识节点，实现了节点数量的无限并行扩展，使得系统的性能不受节点数量增加的影响。Du 等[14]针对恶意节点成为主节点之后干扰共识过程而导致性能下降的问题，提出了一种 SVM 和 PBFT 相结合的主节点选取策略，将动态调整的信任度机制引入区块链网络，提高了共识系统的安全性。Li 等[15]提出了一种基于可验证随机函数 VRF 的共识节点选举方法，使得其算法适用于动态网络，但是这种动态网络仅仅是参与共识的节点发生了变化，而节点的总数并没有改变。Fang 等[16]引入环签名机制，减少了视图切换，解决了节点动态加入和退出的问题，但是节点总数没有发生变换，因此不能算是真正意义上的节点的动态加入和退出。Xu 等[17]按照节点角色进行分类，每类选出一个节点参与共识，实现了一种两层的 PBFT 共识机制，降低了通信复杂度。Hao 等[18]解决了节点数量固定以及不能动态移除失效节点

的问题，但是其引入了可信的中心节点 $Node_{CA}$ 来管理成员节点，违背了区块链去中心化的原则。

### 2.3 本文算法思想

本文设计的协议基于 PBFT，在 PBFT 中添加 4 个子协议，解决了 PBFT 中节点无法动态加入、退出的问题。在正常运行情况下依然使用 PBFT 来达成共识。由于在节点加入、退出时需要修改系统中的一些参数，因此，当有节点加入或者退出网络时会中断 PBFT 三阶段协议，转而去执行节点加入或退出的协议。

本文提出了一种基于信任度积分的主节点选取方法，对原始 PBFT 的主节点选取方法进行优化，增加了诚实节点成为主节点的概率，提升了共识效率。

## 3 算法实现

### 3.1 主节点选取

在协议中，一方面允许参与共识的节点动态加入/退出，网络中的节点总数 $n$ 并不是常数，也就不能使用 $p=v \bmod n$ 来选取主节点；另一方面，原始 PBFT 算法的主节点选取的方法比较粗糙，对网络的共识效率有较大影响。因此，需要重新制定主节点选取的方法。

每个节点维护的都有一个节点信息列表（Node Information List，NIL），NIL 中包含每个节点的 ID，IP，Reputation，State。如表 1 所列。

表 1 节点信息

Table 1 Node information

| ID | IP | Reputation | Class | State |
|---|---|---|---|---|
| 1 | 192.168.59.1 | 5 | honest | normal |
| 2 | 192.168.59.3 | 3.2 | common | normal |
| 3 | 192.168.59.10 | 4.7 | honest | normal |
| 4 | 192.168.59.160 | 2.8 | common | normal |
| 5 | 192.168.59.68 | 0.9 | doubtful | normal |
| … | … | … | … | … |

系统中节点的状态可分为 3 种：normal（正常）、absent（缺失）、malicious（恶意）。系统中的节点按照信任度得分可分为 3 类：honest（诚实）、common（一般）、doubtful（可疑）。其中，honest 节点和 common 节点都可以成为主节点。如果当前网络中存在 honest 节点，则 honest 节点优先成为主节点；如果当前网络中没有 honest 节点，则在 common 类型节点中随机选择一个主节点。honest 节点、common 节点、doubtful 节点都可以参与共识，初始化时所有节点都是 common 类型，新加入网络中的节点也是 common 类型；如果节点长期处于 doubtful 类型则会被标记为 malicious 状态；被划分为 malicious 类型的节点不能参与共识，会被踢出网络，实现对作恶节点的惩罚。

依据信任度划分节点类型的方式如表 2 所列。

表 2 节点信任度分类

Table 2 Classification of node trust

| honest | common | doubtful |
|---|---|---|
| [mid, high] | [low, mid) | (0, low) |

其中，low，mid，high 都是阈值，在实际应用中可以动态调节。

每个节点本地维护的都有一个恶意节点列表（Malicious Node List，MNL），MNL 中存储的是状态为 malicious 的节点信息，当节点 $i$ 发现某个节点的类型在连续 $\tau$（阈值）轮中都是 doubtful 并持续作恶时，就将该节点的信息写入 MNL 中，并将其状态改为 malicious，当 $i$ 发现 MNL 中节点数量超过阈值 $\eta$ 时，就启动 RCLEAR 子协议，将这一部分节点清除出共识网络。

网络中不同的节点有不同的行为，有的节点表现活跃，积极参与共识；有的节点表现懒惰，未及时参与共识；有的节点表现恶劣，提供虚假信息。基于此，将节点在共识过程中的行为分为以下 3 类。

积极参与共识：如果在某轮共识中某节点的行为与大多数节点一致，则称该节点积极参与共识。

消极参与共识：如果在某轮共识中某节点的行为与大多数节点不一致，则称该节点消极参与共识。

未参与共识：如果在某轮共识中其他节点未收到来自该节点的消息，则称该节点未参与共识。

通过对节点的历史共识参与度、历史共识行为、行为奖惩等因素对节点的信任度进行评分。

**定义 1**（节点信任度评价模型） 用于对节点现有信任得分的评估。节点 $i$ 的信任度评价模型为：

$$R_{i,k}=\alpha * A_i+\beta * C_i+\gamma * L_{i,k} \tag{1}$$

其中，$R_{i,k}$ 是节点 $i$ 在第 $k$ 轮共识过程中的信任度得分，由 3 部分组成：$A_i$ 是节点 $i$ 的历史共识行为因子；$C_i$ 是节点 $i$ 的历史共识参与度因子；$L_i$ 是节点 $i$ 的奖惩因子，当节点 $i$ 在第 $k$ 轮做出诚实行为时，会得到一定的奖励，做出消极行为时，会得到一定的惩罚。$\alpha$，$\beta$，$\gamma$ 分别是三者的权重值。通过对这些指标赋予不同的权重来计算节点现有的信任度得分，作为节点等级划分的依据。

**定义 2**（历史共识参与度因子） 用于评价节点的活跃程度。节点 $i$ 的历史参与共识度 $C_i$ 可以表示为：

$$C_i=\frac{T_i}{T} \tag{2}$$

其中，$T$ 是共识网络在过去一段时间内进行的共识过程次数的总和，$T_i$ 是该段时间内节点 $i$ 参与共识过程的总次数。$C_i$ 可以反映出节点 $i$ 的活跃程度，节点 $i$ 参与程度越高，活跃度越高，对信任值评价的正面影响也就越大。

**定义 3**（历史共识行为因子） 用于评价节点 $i$ 的历史共识行为对信任度的影响程度。节点 $i$ 的历史共识行为因子 $A_i$ 可以表示为：

$$A_i=\begin{cases}\dfrac{\sum_{k=1}^{m} h(k) * T_{iN}}{m}, & m\neq 0\\ 0, & m=0\end{cases} \tag{3}$$

其中，$m$ 是共识进行的轮数，$T_{iN}$ 是节点 $i$ 的共识参与标识 $T_N$ 值，是节点是否成功参与共识的一种评估方式，可以表示为：

$$T_N=\begin{cases}1, & \text{成功参与共识}\\ 0, & \text{未参与共识}\\ -1, & \text{未成功参与共识}\end{cases} \tag{4}$$

其中，$h(k)$是时间衰减因子，用来衡量共识轮次的先后对信任度得分的影响，可以表示为时间衰减函数。

$$h(k)=e^{-(m-k)},(1\leqslant k\leqslant m) \tag{5}$$

在每轮共识过程中都有一个时间衰减因子与之对应，共识轮次距离当前的轮次越远，时间衰弱越大，对节点的信任度得分影响就越小，即近期的共识过程行为是更为重要的评价指标。

**定义 4**(奖惩因子)　对节点的当前行为进行奖惩评价，有利于激励节点做出诚实行为。第 $k$ 轮奖惩因子$L_{i,k}$可以表示为：

$$L_{i,k}=\begin{cases}f(R_{i,k-1}), & T_N=1\\ 0, & T_N=0\\ f(R_{i,k-1})-1, & T_N=-1\end{cases} \tag{6}$$

其中，$R_{i,k-1}$是第 $k-1$ 轮共识后节点的信任度得分。$f(x)$是标准正态函数，表达式为：

$$f(x)=\frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}} \tag{7}$$

对于信任度高的节点，其做出诚实行为时，所获得的信任奖励相对于信任度低的节点所获得的奖励较低。同时，信任度高的节点行为出错时，其所受的信任惩罚较大；而信任度低的节点行为出错时，其所受到的惩罚相对较小。这样可以避免高信任节点权利集中，并激励节点做出诚实行为。

**定义 5**(依据积分选择主节点的概率)　当存在 honest 节点时，依据节点积分占总 honest 节点积分的比值来计算概率。honest 节点 $i$ 成为主节点的概率可以表示为：

$$P_i=\frac{R_i}{\sum_{j=1}^{t}R_j} \tag{8}$$

为所有 honest 类型的节点进行统一编号，便于计算。其中，$R_i$是 honest 节点 $i$ 的当前信任得分，$t$ 是 honest 类型节点的个数。

如果不存在 honest 节点，则在所有 common 节点中随机选择一个主节点。如果存在 $t$ 个 honest 节点，则依据积分计算这 $t$ 个 honest 节点成为主节点的概率，并按照以下步骤选取主节点：

(1)现任主节点生成一个[0,1]之间的随机数 $s$ 并广播给所有副本节点；

(2)若 $0\leqslant s\leqslant P_1$，则$P_1$对应的节点成为主节点，若$P_1<s\leqslant P_1+P_2$，则$P_2$对应的节点成为主节点，以此类推，若$P_1+P_2+\cdots+P_{(t-1)}<s\leqslant 1$，则$P_t$对应的节点成为主节点。

### 3.2　JOIN 子协议

节点的加入和退出的时间是未知的，如果在三阶段共识执行过程中有节点退出和加入共识网络，则需要中断三阶段共识，并且在子协议执行之后还需要保证能够继续执行，因此需要在子协议传递的消息中包含之前三阶段协议时请求 $m$ 的唯一的序列号 $r$。这样一来，在子协议结束之后就能够重新启动三阶段共识协议，完成对 $m$ 的共识，不需要在每次子协议执行过程中都通过执行视图转换来恢复三阶段协议的执行，提升了效率，降低了资源开销。

(1)要加入共识的节点 $j$ 是一个已经获得授权的节点，它会向全网所有节点广播〈JOIN-REQ，$j$，IP，PK〉$_{S_j}$ 消息，主节点检查节点 $j$ 的资质，如果验证通过，则主节点 $p$ 将 $j$ 的相关信息写入 NIL 中，其中〈 〉内包含的是消息体，JOIN-REQ 是消息名称，$j$ 是节点的 ID，IP 是节点 $j$ 的网络 IP，PK 是 $j$ 的公钥，$S_j$表示该消息经过了发送者 $j$ 的签名，接收到该消息的节点可以使用 PK 来验证发送节点的签名，以此来验证消息发送者的身份。节点在收到消息后会先将消息存入本地日志中。本文后续出现的消息均使用这种格式来表示。

(2)主节点 $p$ 向所有副本节点广播〈JOIN-PRE，$j$，IP，PK，$p$，$r$，$t$〉$_{S_p}$ 消息，其中，$p$ 是主节点的 ID，$r$ 是原有三阶段共识时最新的请求的序列号，$t$ 是主节点为本次子协议消息所分配的序列号，副本节点 $i$ 收到 JOIN-PRE 消息后，将其与自己收到的节点 $j$ 的请求信息进行比较验证。

(3)如果验证通过，节点 $i$ 就在所有副本节点之间多播〈JOIN，$r$，$t$，$v$，$h$，$C$，$j$，$i$〉$_{S_i}$ 消息并中断 PBFT 三阶段共识，其中，$h$ 是节点 $i$ 记录的最新的稳定的检查点 $s$ 的序列号，$C$ 是 $2f+1$ 个证明 $s$ 正确性的检查点的消息。

(4)至此，网络中的所有节点就节点 $j$ 的加入达成了共识，并将 $j$ 的相关信息写入自己的 NIL 中。节点 $i$ 向 $j$ 回复〈JOIN-REPLY，$r$，$t$，$v$，$j$，$B$，$i$，NIL〉$_{S_i}$ 消息，其中，$B$ 是一组来自 $2f+1$ 个节点的 JOIN 消息(在实际应用中，为了降低通信开销，通常发送消息的摘要以及消息发送者的 ID)。节点将 $B$ 和 NIL 回复给 $j$，使得 $j$ 能够快速与共识网络中其他节点达成一致。如果 $j$ 收到 $f+1$ 个有效的 JOIN-REPLY 消息，那么它就加入共识网络。

如果节点 $j$ 由于某些原因(自身网络故障、未成功收到 $f+1$ 条有效消息等)此时未能加入共识网络，导致节点 $j$ 与共识网络中其他节点的数据不一致，这也不影响协议的安全性，因为共识网络中一个节点长时间未积极参与，它的信任积分会降低，最终被踢出网络。算法流程如图 2 所示。

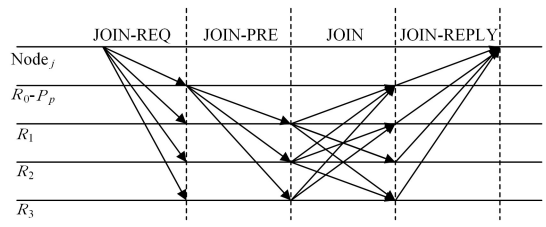


图 2　JOIN 算法流程图

Fig. 2　Process diagram of JOIN

### 3.3　EXIT 子协议

每个节点本地维护的还有一个节点撤销列表(Node Revocation List，NRL)。

(1)当节点 $j$(副本节点)想要退出网络时，它首先向全网节点广播〈EXIT-REQ，$j$，IP，PK〉$_{S_j}$，在收到来自其他节点的足够的回复之前，节点 $j$ 依旧要继续参与三阶段共识。

(2)主节点 $p$ 收到 EXIT-REQ 消息时,会将节点 $j$ 的相关信息放入 NRL 中,并将相关信息存入自己的日志中,备份节点收到 EXIT-REQ 消息后,会先将相关信息存入自己的日志中。主节点 $p$ 每隔一段时间 $T$ 会检查自己的 NRL,如果 NRL 不为空,$p$ 会向所有副本节点多播〈EXIT-PRE,NRL,$p$,$r$,$t$〉$_{S_p}$ 消息,其中 NRL 为 $p$ 维护的节点撤销列表,$p$ 为主节点的 ID。

(3)其他副本节点收到 EXIT-PRE 消息后,会对消息进行验证(验证主节点发送的 NRL 中的节点信息与自己本地日志中存储的请求信息是否一致),若验证通过,则将相应的节点信息放入自己的 NRL 中,并启动 EXIT 协议,向所有节点广播〈EXIT,$r$,$t$,$v$,$h$,$C$,NRL,$i$〉$_{S_i}$ 消息,如果节点 $i$ 收到 $2f+1$ 条经过验证的 EXIT 消息,就将自己的 NRL 列表中的节点状态标记为 absent。

(4)至此,所有节点就 $j$ 及其他若干节点的退出达成了共识,它们会再向 $j$ 发送〈EXIT-REPLY,$r$,$t$,$v$,$j$,$i$,$B$〉$_{S_i}$ 消息。其中,$B$ 是一组来自 $2f+1$ 个节点的 EXIT 消息(在实际应用中,为了降低通信开销,通常发送消息的摘要以及消息发送者的 ID)。如果 $j$ 收到 $f+1$ 条有效的 EXIT-REPLY 消息,那么它就主动退出共识网络,不再参与共识。

EXIT 算法流程如图 3 所示。

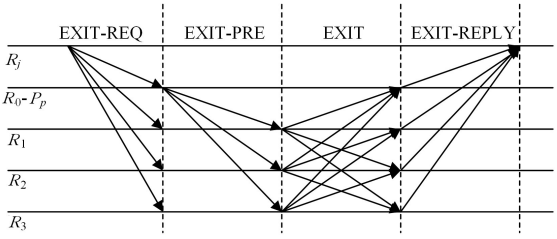


图 3 EXIT 算法流程图

Fig. 3 Process diagram of EXIT

### 3.4 PCLEAR 子协议

虽然我们会尽最大努力来保证主节点是诚实的,但是仍然不能排除主节点作恶或宕机的可能性。其余副本节点一旦发现主节点作恶,就会试图启动 PCLEAR 子协议,将作恶节点踢出共识系统。

(1)当副本节点 $i$ 发现主节点 $p$ 作恶时,$i$ 会向其他副本广播〈PCLEAR,$v+1$,$t$,$h$,$C$,$p$,$i$〉$_{S_i}$ 消息,其中,$p$ 是作恶主节点的 ID。所有节点在收到 $2f+1$ 条相同的 PCLEAR 消息后都会进行验证。

(2)当新的主节点 $p_1$ 收到 $2f+1$ 个有效的 PCLEAR 消息后,就会广播〈NEW-VIEW,$v+1$,$B$,$O$,$t$,$p_1$〉$_{S_{p_1}}$ 消息(除了恶意主节点)。其中,$p_1$ 是新的主节点的 ID,$B$ 是一组来自 $2f+1$ 个节点的 PCLEAR 消息(在实际应用中,为了降低通信开销,通常发送消息的摘要以及消息发送者的 ID),$O$ 是一组 pre-prepare 消息,分为两步计算。首先,$p$ 根据 $B$ 中最后一个稳定检查点选择序列号 min-$s$,并在 $B$ 中的 prepare 消息中选择最高序列号 max-$s$;然后,主节点为视图 $v+1$ 的 min-$s$ 和 max-$s$ 之间的每个序列号创建一个 pre-prepare 消息。节点 $i$ 收到 NEW-VIEW 消息后,会进行相关的检查。此时所有节点都关于 $p$ 是恶意节点这一信息达成了共识,所有节点都会在自己的 NIL 中将 $p$ 的状态改为 malicious,不允许该节点再申请加入。

(3)至此,$p$ 已经是所有节点公认的恶意节点且永远都不能再参与共识。然后,作为一个完整的交互过程,节点 $i$ 应该向作恶的主节点 $p$ 回复〈PCLEAR-REPLY,$v+1$,$t$,$p_1$,$i$〉$_{S_i}$ 消息。

主节点一旦作恶,将直接被划分为恶意节点。如果 PCLEAR 没有通过,还需要恢复节点的状态。算法流程如图 4 所示。

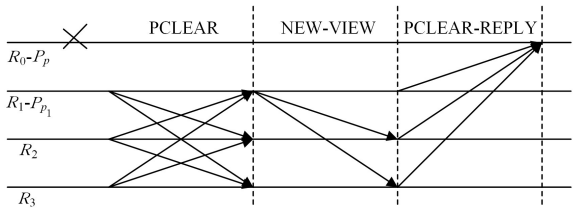


图 4 PCLEAR 算法流程图

Fig. 4 Process diagram of PCLEAR

### 3.5 RCLEAR 子协议

根据节点信任度积分以及节点最近 $\tau$ 轮的表现来判断是否将节点的状态设置为 malicious,并将节点信息写入 MNL 中。当节点 $i$ 发现 MNL 中存储的节点超过 $\eta$ 时,就启动 RCLEAR 子协议。

(1)当副本节点 $i$ 发现某个其他的副本节点 $j$ 作恶时,它就向当前的主节点 $p$ 发送〈RCLEAR-REQ,$i$,$PK$,$j$〉$_{S_i}$ 消息,其中,$j$ 是作恶节点的 ID。

(2)$p$ 收到 $2f+1$ 条有效的 RCLEAR-REQ 消息时,便向除了节点 $j$ 以外的其他副本节点多播〈RCLEAR-PRE,$j$,$PK$,$r$,$t$〉$_{S_p}$,其中,$r$ 是当前执行三阶段共识中消息的序号,$t$ 是主节点为该子协议消息分配的序号。将 $j$ 的状态标记为 malicious,并将 $j$ 放入 MNL 中。

(3)$i$ 收到 RCLEAR-PRE 消息后会对 $j$ 的状态进行验证,如果验证通过,则广播〈RCLEAR,$r$,$t$,$v$,$h$,$C$,MNL,$i$,$j$〉$_{S_i}$ 消息。

(4)节点收到 $2f+1$ 条有效的 RCLEAR 消息时,会将 $j$ 的状态标记为 malicious,并将 $j$ 放入 MNL 中 。

(5)此时,所有节点就清除恶意节点的事情达成了共识,向恶意节 $j$ 发送〈RCLEAR-REPLY,$r$,$t$,$v$,MNL,$i$,$B$〉$_{S_i}$ 消息。其中,$B$ 是一组来自 $2f+1$ 个节点的 EXIT 消息(在实际应用中,为了降低通信开销,通常发送消息的摘要以及消息发送者的 ID)。强制恶意节点退出共识且不再接受该节点的加入申请。

若节点长期没有答复(宕机)导致信任度积分过低,则状态置为 absent 并踢出网络,允许其后续再申请加入网络;若节点消极参与共识导致长期信任度低,则状态置为 malicious,不允许其后续再加入网络。算法流程如图 5 所示。

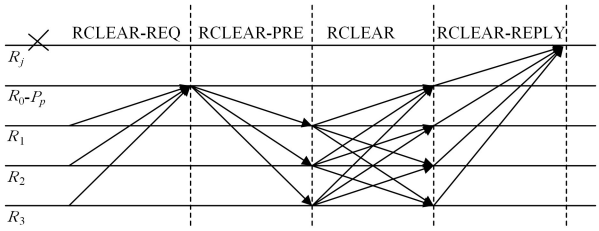


图 5　RCLEAR 算法流程图

Fig. 5　Process diagram of RCLEAR

## 4　正确性分析

根据 CAP(Consistency-Availability-Partition Tolerance)原理[19]，一个分布式系统最多只能同时满足一致性、可用性和分区容忍性 3 项中的两项，此处的一致性指强一致性。但是在满足可用性和分区容忍性的同时可以采取适当的方式达到最终一致性，即所有副本在经过一定的时间之后，最终能够达成一致的状态[2]。

首先，在进行三阶段共识的过程中，没有对原始 PBFT 算法进行修改，因此自然能够保证这一阶段的一致性。其次，在子协议阶段，各个节点之间的 MNL，NIL，NRL 每隔一段固定的时间也会相互交换，这就能够保证共识网络中节点的数据的最终一致性。各个节点本地都保存了日志信息，当新加入的节点进入网络时，其余节点向新节点回复的信息中有之前网络中进行的请求，同时新节点也会从主节点处进行日志同步，从而也能够保证所有节点的操作、日志的一致性。

### 4.1　安全性

本文在不改动原始 PBFT 算法的前提下，添加了 4 个子协议。在 PBFT 阶段利用其原有的共识特性就能够保证系统的安全性。因此，本文的安全性论述主要包括 4 个子协议本身的安全性证明，以及在加入子协议之后原始 PBFT 共识协议的安全性的证明。

对于 JOIN 子协议，其本质就相当于一个客户端向共识网络发起请求，唯一不同的是该请求可以中断原有的三阶段共识。本文提出的主节点选取机制能够最大程度地保证主节点的诚实性，且能够对作恶主节点做出及时惩罚，保证了系统的安全性。在联盟链场景中，节点在向共识网络发送 JOIN 请求之前需要先获得授权证书并在请求消息中附带上其授权证书，从而保证新加入的节点的诚实性。网络中的节点会将要加入节点的信息存入本地日志，若发现有节点频繁请求加入，则拒绝其请求，从而避免因频繁启动 JOIN 子协议而影响三阶段共识的效率。

对于 EXIT 子协议，其本质相当于一个客户端向网络中其他节点发送请求消息。它不会使得节点立即退出，而是先将要退出的节点收集起来，当要退出的节点超过一定数量时再一并处理，这样可以避免频繁启动 EXIT 子协议。并且规定每个节点都需要将要退出的节点的信息写入自己的日志中，如果中途更换主节点，那么新的主节点可以从自己的日志中读取要退出的节点的信息，生成 NRL。

对于 PCLEAR 子协议，副本节点发现主节点故障后会发送 PCLEAR 消息，然后会进行一次 PBFT 中的视图转换，从而保证系统的安全性。

对于 RCLEAR 子协议，节点 $i$ 发现某个副本节点的信任度长期较低，且在作恶时会考虑将其状态置为 malicious 并将其写入 MNL 中，当 MNL 中节点数量较多时，会发送 RCLEAR-REQ 消息，请求开启 RCLEAR 子协议。这样做一方面能够避免由于节点某一次网络不通畅造成的误判，另一方面也能减少启动 RCLEAR 的次数。

接下来将论述在加入 4 个子协议之后原始 PBFT 的安全性问题。

在 JOIN 子协议(EXIT，RCLEAR 子协议也类似)中：

(1)当节点 $i$ 已经在它的本地日志中插入了消息 $m$、消息 $m$ 在视图 $v$ 中序列号为 $n$ 的 pre-prepare 消息和其他 $2f+1$ 个节点的 prepare 消息时，$prepared(m,v,n,i)$为 true；

(2)当存在 $f+1$ 个诚实节点认为 $prepared(m,v,n,i)$为 true 时，$committed(m,v,n)$为 true；

(3)当 $pre\text{-}prepared(m,v,n,i)$为 true 且节点 $i$ 已经收到 $2f+1$ 个不同的与 $m$ 的 pre-prepare 消息相匹配的节点 commits 消息时，$committed\text{-}local(m,v,n,i)$为 true。

当且仅当 $committed(m,v,n)$为真时，$m$ 会在视图 $v$ 中序列号为 $n$ 的诚实节点上进行本地提交。因此就存在一个节点集合$R_1$，其中至少包含 $f+1$ 个诚实的副本节点，并且对于其中每一个节点 $i$，都有 $prepared(m,v,n,i)$为 true。然后在 JOIN 子协议中，当 JOIN 子协议被 PCLEAR 子协议中断时，每个 PCLEAR 子协议中都包含视图转换步骤，而每个正确的 NEW-VIEW 消息中都包含来自副本节点集合$R_2$中的经过验证的 $2f+1$ 个副本的 JOIN 消息的唯一序列号。$R_1$和$R_2$中至少有一个相同的节点 $e$。否则，它们总的节点个数为$(f+1)+(2f+1)=3f+2$，这与我们假定的系统总节点个数 $3f+1$ 相悖。因此，假设集合$R_1$和$R_2$中公共的诚实节点数量为 $w$。节点 $e$ 发送的 JOIN 消息可以将 $m$ 在之前的视图中的 prepare 消息传播到新的主节点，那么新的主节点将在 NEW-VIEW 消息中包含具有相同序列号和新视图号的 $m$，要求副本节点为消息 $m$ 重新启动 PBFT 三阶段协议。该方法能够有效防止旧视图中具有相同序列号的不同的消息 $m^1$的提交。

在 PCLEAR 子协议中，相当于进行了一个 PBFT 协议的视图转换。唯一与视图转换不同的是，PCLEAR 子协议会修改作恶主节点的状态从而将它踢出共识系统。与前面的叙述类似，同样可以在视图转换机制中保证系统的安全性。

综上所述，该协议在三阶段共识和子协议中都可以保证系统的安全性。

### 4.2　活性

在 PBFT 共识阶段，网络通过前文所述的信任度的方式来确定主节点，视图编号从 0 开始递增。当副本节点的计时器超时或者收到 $f+1$ 条有效的视图转换消息时，意味着当前视图的主节点已经无法完成共识，副本必须进入下一个视图重新进行共识服务。同时，视图转换只在需要时才进行。这就需要保证以下两点：

(1)系统中至少有 $2f+1$ 个正常副本节点处于同一视图中时,应尽可能长时间地保持这种状态;

(2)每次视图变更时,上述时间间隔需要快速增长,比如,以指数形式增长。

子协议通过以下几种规则来保证活性:

(1)为了防止子协议启动过快而影响原有三阶段共识的效率,当一个节点发送 JOIN(或者 EXIT, PCLEAR, RCLEAR)请求后,它会等待其他节点的 JOIN 消息,在它等待接收 $2f+1$ 个 JOIN 请求消息时,会同时启动定时器并设置超时时间为 $T$。若其没有在 $T$ 时间内接收到 $2f+1$ 个 JOIN 请求消息,则会重新发送 JOIN 消息并重启计时器。此时,算法会以指数递增的方式调整超时时间,将其设置为原来的两倍,即 $2T$。

(2)除了上述的定时器超时触发节点发送 JOIN 消息外,当其接收到来自 $f+1$ 个不同节点的有效的 JOIN 消息,该节点会立即向其他节点发送 JOIN 消息,从而防止节点过晚启动子协议。

(3)基于上述第二点的规则,恶意节点无法通过故意发送 JOIN 消息来触发频繁地中断三阶段协议从而干扰系统的正常运行。因为恶意节点最多只能发送 $f$ 条消息,达不到 $f+1$ 的触发条件。然而,当主节点是恶意节点时,可以故意发送恶意消息造成子协议以及三阶段协议的失败,但是主节点作恶的代价非常高。

(4)JOIN, EXIT, RCLEAR 这 3 个子协议启动之前都由主节点为其分配唯一递增的序列号,前一个序列号的子协议未完成之前不允许启动下一个子协议,这 3 个子协议在执行之前都会先查看共识网络中是否正在运行其他的子协议,如果正在运行其他的子协议,则需等待其他子协议运行完毕之后才能启动。PCLEAR 子协议可以中断其他子协议,在 PCLEAR 初始的发送的消息中会包含当前子协议的编号以及当前三阶段协议的编号,PCLEAR 在执行完毕之后会重新启动被中断的子协议和三阶段协议。

综上所述,系统在实际应用中可以保证活性。

## 5 实验及结果分析

实验使用 simpy 模拟搭建联盟链环境,操作系统为 Windows 10, CPU 为 Intel(R) Core(TM) i5-9400 CPU @ 2.90GHz,内存为 8GB,算法实现语言为 Python。在不考虑网络拥塞的前提下,模拟 100 个共识节点组成共识网络,并比较 PBFT 算法和 DPBFT 算法的共识效率。

### 5.1 新节点申请加入网络的实验

本实验中比较了 PBFT 算法和 DPBFT 算法完成共识的耗时情况,即在网络中拜占庭节点占比 10%且当新节点以不同的概率申请加入共识网络时,完成 5000 轮共识所耗费的时间。实验结果如图 6 所示。从图 6 可以看出,PBFT 算法无法应对节点的频繁加入,一旦有新的节点申请加入,PBFT 共识网络就需要重启,即 PBFT 本身并不能应对节点的动态变化,耗费的资源巨大,时延增加也较快;而 DPBFT 在节点频繁加入时,所耗费的时间增加缓慢且变化相对不明显。由此可见,在应对节点频繁加入的问题上,DPBFT 算法较 PBFT 算法有明显改进。

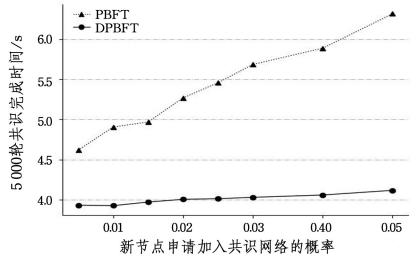


图 6 PBFT 和 DPBFT 完成 5000 轮共识的耗时比较

Fig. 6 Comparison of time consumption between PBFT and DPBFT for 5000 rounds of consensus

### 5.2 PBFT 与 DPBFT 完成共识的时间效率的比较

本实验中比较了 PBFT 算法和 DPBFT 算法在网络中拥有不同数量的拜占庭节点时,完成若干轮共识所耗费的时间,以及当网络中有相同数量的拜占庭节点时,PBFT 和 DPBFT 完成若干轮共识所耗费的时间。实验结果如图 7 所示。

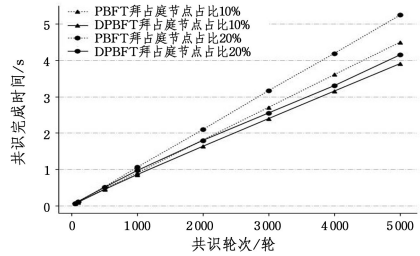


图 7 PBFT 与 DPBFT 的耗时比较

Fig. 7 Comparison of time consumption between PBFT and DPBFT

从图 7 可以看出,当 PBFT 和 DPBFT 网络中的拜占庭节点相同且进行的共识轮次相同时,PBFT 完成共识的耗时比 DPBFT 多;当拜占庭节点数量相同时,随着共识轮次的增加,DPBFT 算法耗时相比 PBFT 算法增加较慢。PBFT 与 DPBFT 的 1000 轮共识耗时比较如表 3 所列。

表 3 PBFT 与 DPBFT 的 1000 轮共识耗时的比较

Table 3 Comparison of time consumption between PBFT and DPBFT for 1000 rounds of consensus

(单位:s)

| 拜占庭节点占比 | PBFT | DPBFT |
|---|---|---|
| 10% | 0.897 | 0.758 |
| 20% | 1.040 | 0.755 |

可以看出,当拜占庭节点的数量占比从 10%增加到 20%时,PBFT 1000 轮共识的时间增加 15.9%,而 DPBFT 1000 轮共识的时间基本不变。由此可见,DPBFT 算法的优势更加明显。

**结束语** 本文以目前应用广泛的联盟链为研究背景,提出了一种对 PBFT 进行改进的算法——DPBFT,使用 4 个子协议来对 PBFT 算法进行优化,对原始 PBFT 的主节点的选取方式进行改进,通过计算节点的信任度积分使得选取到的主节点更为可信;在不影响原始 PBFT 安全性和活性的前提

下，解决了节点动态加入、退出的问题，并实现了对作恶节点的惩罚。

本文提出的 DPBFT 算法目前未对新申请加入的节点的可信度进行审核，需要依赖于联盟链的审核机制。在今后的研究工作中，将考虑利用改进算法自动审核新加入节点的可信度，并将该算法应用于某一实际场景中，为区块链技术的应用和发展做出更大贡献。

## 参 考 文 献


[1] NAKAMOTO S. Bitcoin：a peer-to-peer electronic cash system. [EB/OL]. https://bitcoin.org//bitcoin.pdf.

[2] CAI X Q, DENG Y, ZHANG L, et al. The Principle and Core Technology of Blockchain [J]. Chinese Journal of Computers, 2021, 44(1): 84-131.

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

[4] ZHANG P Y, SONG J. Research Advance on Efficiency Optimization of Blockchain Consensus Algorithms [J]. Computer Science, 2020, 47(12): 296-303.

[5] XIA Q, DOU W S, GUO K W, et al. Survey on Blockchain Consensus Protocol [J]. Journal of Software, 2021, 32(2): 277-299.

[6] KING S, NADAL S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake [J/OL]. http://inpluslab.sysu.edu.cn/files/blockchain/proof_of_stake.pdf.

[7] HERLIHY M P, WING J M. Linearizability: A correctness condition for concurrent objects [J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 1990, 12(3): 463-492.

[8] CASTRO M, LISKOV B. Practical Byzantine fault tolerance [J]. Operating Systems Review, 2002, 20(4): 398-461.

[9] ZHAN Y, WANG B C, LU R X, et al. DRBFT: Delegated randomization Byzantine fault tolerance consensus protocol for blockchains [J]. Information Sciences, 2021, 559: 8-21.

[10] YU G, WU B, NIU X X. Improved Blockchain Consensus Mechanism Based on PBFT Algorithm[C]// 2nd International Conference on Advances in Computer Technology, Information Science and Communications (CTISC 2020). 2020: 14-21.

[11] LI W Y, FENG C L, ZHANG L, et al. A Scalable Multi-Layer PBFT Consensus for Blockchain [J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(5): 1146-1160.

[12] JALALZAI M M, BUSCH C. Window Based BFT Blockchain Consensus[C]// 11th IEEE International Congress on Conferences on Internet of Things, 14th IEEE International Conference on Green Computing and Communications, 11th IEEE International Conference on Cyber, Physical and Social Computing, 4th IEEE International Conference on Smart Data, 1st IEEE International Conference on Blockchain and 18th IEEE International Conference on Computer and Information Technology, iThings/GreenCom/CPSCom/SmartData/Blockchain (CIT 2018). 2018: 971-979.

[13] WANG F L, JI Y P, LIU M S, et al. An Optimization Strategy for PBFT Consensus Mechanism Based on Consortium Blockchain[C]// 3rd ACM International Symposium on Blockchain and Secure Critical Infrastructure (BSCI 2021). 2021: 71-76.

[14] DU N, LIANG Z, HUANG Y, et al. Performance optimisation Method of PBFT Consensus for Supply Chain Integration SVM [C]// 2020 7th International Conference on Dependable Systems and Their Applications (DSA). 2020: 371-377.

[15] LI Y, WANG Z, FAN J, et al. An Extensible Consensus Algorithm Based on PBFT[C]// 2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). 2019: 17-23.

[16] FANG Y, DENG J Q, CONG L H, et al. An Improved Scheme for PBFT Blockchain Consensus Algorithm Based on Raing Signature [J]. Computer Engineering, 2019, 45(11): 32-36.

[17] XU X L, ZHU D W, YANG X X, et al. Concurrent Practical Byzantine Fault Tolerance for Integration of Blockchain and Supply Chain [J]. ACM Transactions on Internet Technology, 2021, 21(1): 1-17.

[18] HAO X, YU L, ZHI Q L, et al. Dynamic Practical Byzantine Fault Tolerance[C]// 2018 IEEE Conference on Communications and Network Security (CNS). 2018: 1-8.

[19] FOX A, BREWER E. A Harvest, yield, and scalable tolerant systems[C]// Proceedings of the Seventh Workshop on Hot Topics in Operating Systems. 1999: 174-178.


**XIE Zhuo**, born in 1997, postgraduate, is a member of China Computer Federation. His main research interests include blockchain and so on.

**LI Lei**, born in 1974, Ph. D. His main research interests include computer network, network secirity and so on.

（责任编辑：何杨）