文章编号：2095-2163(2023)07-0191-11 中图分类号：TP212.9 文献标志码：A

基于节点可信度的改进 PBFT 算法

韩 璐，翟健宏，杨 茹
（哈尔滨工业大学 网络空间安全学院，哈尔滨 150001）

摘 要：本文主要是对区块链的核心技术—共识算法进行深入了解，在联盟链中难以避免会有恶意节点的存在，在节点达成共识的过程中，恶意节点将会散布不实信息，影响数据的一致性。本文对区块链防作弊技术进行研究，提出了一种基于节点可信度的 NRPBFT(node-reliability-based PBFT consensus algorithm)共识算法，利用节点的可信度将所有的节点分成普通节点和共识节点，提高了共识的效率，降低了主节点作恶的可能性。

关键词：区块链；NRPBFT；共识算法；可信度计算

Improved PBFT algorithm based on node credibility

HAN Lu, ZHAI Jianhong, YANG Ru
(School of Cyberspace Science, Harbin Institute of Technology, Harbin 150001, China)

【Abstract】 This paper mainly focuses on the in-depth understanding of the core technology of blockchain - consensus algorithm. It is difficult to avoid the existence of malicious nodes in the alliance chain. In the process of nodes reaching consensus, malicious nodes can spread false information and affect the consistency of data. In this paper, the anti cheating technology of blockchain is studied, and a node-reliability-based PBFT consensus algorithm (NRPBFT consensus algorithm) is proposed. By using the reliability of nodes, all nodes are divided into ordinary nodes and consensus nodes, which improves the efficiency of consensus and reduces the possibility of the master node engaged in malicious activity.

【Key words】 blockchain; NRPBFT; consensus algorithm; credibility calculation

0 引 言

共识机制[1]作为区块链[2]的核心技术，能够在缺乏中央控制的网络中，通过自组织、大规模协作的方式实现账本的分布式一致性，也是区块链通过技术手段而非中心化机构构建去中心化“信任”网络的关键算法。良好的共识算法[3]有助于提高区块链系统的性能效率，对维护系统的稳定运行和节点相互信任起着重要作用。然而，现有的共识算法各有优劣，普遍存在扩展性差、交易确认时间长、应用场景受限等问题，成为制约区块链项目落地的主要因素[4]。

实用拜占庭共识算法(Practical Byzantine Fault Tolerance，PBFT)[5]作为目前联盟区块链使用较多的共识算法，不仅实现了节点之间状态的信任，而且很大程度上降低了系统因拜占庭错误产生的巨大开销[6]，但存在对带宽要求较高，节点数量固定等缺陷。本课题对 PBFT 算法在 fabric 联盟链[7]中的实现进行研究，利用 PBFT 算法的特点，保证了网络中节点的一致性。通过部署链码在 docker[8]环境中对算法的性能进行检测，并根据检测结果对 PBFT 算法进行改进。

本文对区块链防作弊技术进行研究，提出了一种基于节点可信度的 NRPBFT(node-reliability-based PBFT consensus algorithm)共识算法，主要贡献如下：

(1)提出了 NRPBFT 算法，采用选举集群的方式，将所有的节点分成普通节点和共识节点。

(2)利用 PBFT 算法的特性实现了共识过程的防作弊，同时优化了算法的性能，提高了算法的效率。

(3)利用 20 个端口模拟区块链节点，利用 tcp

基金项目：黑龙江省自然科学基金联合引导项目(JJ2019LH0412)。

作者简介：韩 璐(1997-)，女，硕士研究生，主要研究方向：区块链共识算法；翟健宏(1968-)，男，副教授，主要研究方向：网络安全、区块链、工控安全；杨 茹(1970-)，女，教授，主要研究方向：计算机应用。

通讯作者：翟健宏 Email：zhaijh@hit.edu.cn

收稿日期：2022-06-05

实现数据在端口之间的同步,通过实验验证改进后的共识算法的可行性。

本文的组织结构如下:第 1 节介绍了 NRPBFT 算法方案;第 2 节阐述了代码优化的过程及结果;第 3 节结合实验说明了该方案的可行性与高效性;第 4 节总结了全文,并讨论了后续的工作与改进之处。

1 NRPBFT 算法方案

1.1 改进方法介绍

针对 PBFT 算法存在问题的分析,结合调研的结果和文献资料汇总,本节提出了一种 NRPBFT 算法方案。这个方案主要是从缩小网络的规模的角度出发,利用选举集群的方式,将所有的节点分成普通节点和共识节点。利用节点可信度评估公式对节点的可信度进行评估,并对节点的可信度进行排序,选可信度最高的前p个节点作为共识集群($2f \leqslant p < 3f$, f表示网络中恶意节点的数量)。同时,对主节点的选举做了一些改动,选取可信度最高的节点作为主节点,大大降低了恶意节点成为主节点的概率。主要从以下 2 点进行改进:

(1)根据可信度将网络中的节点分成 2 种角色。普通节点只负责完成交易,共识节点集群中可信度最高的节点会成为主节点。主节点接收客户端的消息,在有 p 个共识节点的共识集群中进行广播,对消息进行签名排序后,广播给其他所有节点。节点收到至少 $p/2$ 个相同的消息,认为这个信息正确,写入内存,并将确认信息回复给主节点。

(2) 采用可信度评估的方式来选举共识集群和主节点。在每次共识结束后,都需要根据可信度评估算法重新计算每个节点的可信度,这样就实现了节点动态加入或者退出共识集群。对于共识集群中作恶的节点,研究在评估的时候采取相应的惩罚机制,加上惩罚系数,降低其可信度评估结果。这种方式不仅能降低主节点随意选取带来的风险,也能保证系统的动态性。

网络拓扑设计如图 1 所示。

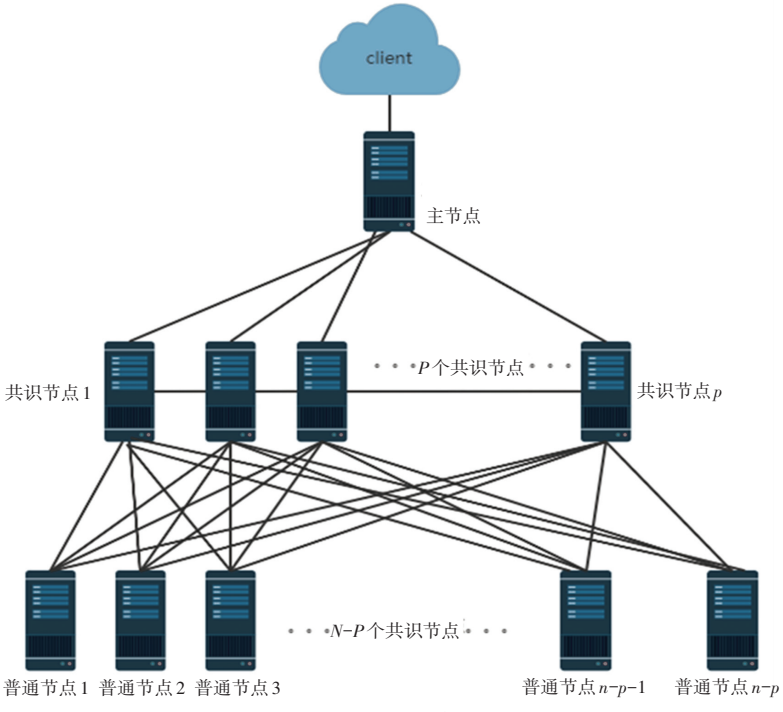


图 1 改进后 PBFT 算法拓扑图

Fig. 1 Topological diagram of the improved PBFT algorithm

1.2 NRPBFT 优化算法流程

(1)客户端 c 将请求信息发送到主节点,主节点验证,提取消息摘要并签名后广播给所有共识节点,广播的格式为 $<<PREPARE,n,S,D>,data>$,其中 $PREPARE$ 是命令名称,代表信息处于的阶段是主节点在共识集群中广播信息阶段,n 是主节点为信息分配的序列号,S 是主节点的签名,D 是主节点提取的信息摘要,$data$ 是待共识信息。

(2) 认证共识阶段。共识集群中的所有节点收到请求后,首先验证交易签名和摘要信息,验证成功后将交易数据存储到内存中,对信息进行签名后,将信息广播给所有的节点,广播信息格式为 $<<AUTHENTICATION,credit\_i,n,S,D>,data>$,其中 $AUTHENTICATION$ 是这一阶段的命令名称,

credit_i 是节点的可信度的值。信息广播到所有节点后，进入提交阶段。如果交易信息验证不成功，则认为主节点可能出了问题，重新对主节点进行选举。

(3) 提交阶段。所有的节点都会收到多个由共识节点发来的信息，节点接收到信息后，对签名和序列号进行验证，有大于等于 $p/2$ 个信息验证成功后，将信息存储到内存中，并向主节点进行提交。信息提交的格式为 $<SUBMIT,n,S,D,i>$，其中 i 是节点的编号。主节点在一定时间内接收到 $2f+1$ 个一致的信息，认为提交阶段完成，主节点将达成共识的信息发送给客户端，完成共识。

算法的流程如图 2 所示。NRPBFT 算法步骤具体如下。

算法 1　新 PBFT 算法

输入　主节点编号 N_0，P 个共识节点 $(N_1 - N_p)$，$n-p$ 个普通节点 (N_p+1, N_n)，节点可信度的值 *credit* − *i*，客户端发送的交易信息 *data*

输出：节点对客户端的 *reply*

1：for 主节点 N_0 接收客户端发送的交易信息 *data*

2：N_0 提取消息摘要 D，对 D 签名得到 S，将信息广播给共识节点，广播信息格式为 $<<PREPARE, n,S,D>, data>$，其中 n 是消息序列号

3：共识节点 N_p 验证交易的签名和摘要

4：if 签名 S 和摘要 D 验证成功 then

5：　共识节点 N_p 将交易信息储存到内存中

6：　共识节点 N_p 对交易信息进行签名，广播给所有节点，广播格式为 $<<AUTHENTICATION, credit\_i,n,S,D>, data>$

7：else then

8：　触发主节点更换协议，按照可信度排序更换主节点，重复步骤 1

9：　普通节点 N_n 接收到共识节点 N_p 的广播信息，验证交易的签名和摘要

10：if 节点有 $P/2$ 个签名 S 和摘要 D 验证成功 then

11：普通节点 N_n 将交易信息储存到内存中

12：　向主节点 N_0 发送确认信息，信息提交的格式为 $<SUBMIT,n,S,D,i>$，其中 i 是节点的编号

13：else then

14：　重新评估节点的可信度，重新选举共识节点集群

15：if 主节点 N_0 接收到 $2f+1$ 个一致的信息 then

16：　主节点 N_0 将确认信息提交给客户端

17：end

18：else then

19：　触发主节点更换协议，按照可信度排序更换主节点，重复步骤 1

还要一值的是，这里的节点 1 和节点 2 均属于共识节点，节点 3 属于拜占庭节点。

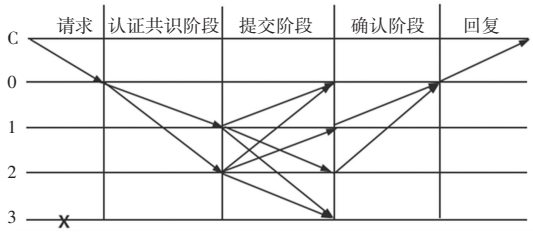


图 2　改进的 PBFT 算法流程图

Fig. 2　Flow chart of improved PBFT algorithm

1.3　节点可信度计算模块

节点可信度计算模块是以往的研究中和研发团队一起提出的方法。可信度是信任的量化表示，表示一个节点中数据可以被另一节点接受的信任程度。可以通过节点自身的行为来判断节点的可信度。

将生成一个区块的时间定义为一个评审周期，其中评审内容包括 6 项，节点每接收到一个区块会动态更新该表格中的矿工节点、该表格中的矿工节点、节点参与验证的区块总数、节点正确验证的区块数、节点最新打包区块字段，每个周期结束后更新信任度和新的起始区块字段，各参数设置见表 1。

表 1　模型参数设置

Tab. 1　Model parameters settings

参数	计算方法
矿工节点账户	负责区块打包的节点账户
区块验证总数	在当前周期内参与区块验证的次数
正确验证总数	将合法区块验证通过并在本地账簿同步次数
打包区块总数	在当前周期内由该节点打包的区块总数
起始区块	当前周期内产生的第一个区块高度
信任度	通过信任度计算公式进行周期性计算

将表 1 中各字段值带入式 (1) 中进行计算，检验计算结果是否达到可信度阈值：

$$trust_{cur}^{(i)} = \frac{1}{1+e^{-\frac{\omega}{\tau}\left(\sum_{x=1}^{\tau} v_x - \sigma \sum_{x=1}^{\tau} \mu_x\right)}} \tag{1}$$

其中，$trust_{cur}^{(i)}$ 表示第 i 个节点在当前评审周期的可信度；τ 表示在一个评审周期内产生的区块数；ω 表示第 i 个节点积极参与争夺记账权并投票的区块数。对于在该评审周期内产生的第 x 个区块，当

第 i 个节点正常参与投票时，将 ν_x 设置为 1，μ_x 设置为 0，否则 ν_x 设置为 0，μ_x 设置为 1，σ 是恶意投票对于节点可信度影响所占的比重，该值越大，惩罚权重越大。研究可知，式(1) 主要用来衡量节点在当前评审周期内的表现情况。

由于式(1)基于 logistics 模型[9]提出，而该模型在对数增长期间函数值增长较快，导致对数增长期间节点信任度相较之前值偏高，因此使用式(2)进行修正：

$$trust(i)_{cur} = \lambda \cdot trust(i)_{cur} + (1-\lambda) \cdot trust(i)_{pre} \quad (2)$$

其中，$trust(i)_{cur}$ 表示当前周期的信任度，$trust(i)_{pre}$ 表示上个周期的信任度。通过 λ 来对当前周期信任度计算进行修正，添加节点信任历史相关性，该参数可由用户定义。研究可知，修正方法主要是中和了该节点在上轮评审周期中的表现情况

一旦某个节点出错，则在下一轮对该节点可信度进行评审的时候，恶意的节点会被乘以一个特别小的系数来降低下一轮评审中此节点的可信度。

1.4 方案分析

本方案的核心特点在于实现了一个基于可信度的 PBFT 算法优化方案、即 NRPBFT，该方案具有以下优势：

(1)高可靠性：选可信度最高的前 p 个节点作为共识集群，在一定程度上保证了系统的安全性和可靠性。研究中也对主节点的选举做了一些改动，选取可信度最高的节点作为主节点，大大降低了恶意节点成为主节点的概率。从节点接收到主节点的广播信息后，首先会对消息的签名、消息的序号和消息的摘要进行验证，3 个信息都验证通过后才能接收这个需要共识的内容，这一过程也大大地保证了信息的可靠性和安全性。

(2)高可用性：原本的 PBFT 算法在进行共识的过程中需要经过 1 次单点全广播和 2 次全点全广播，基于可信度的 PBFT 优化算法只需要经过一次一对多的传输、一次多对一的传输及一次全点全广播。区块链中节点与节点之间的传输过程消耗的资源非常多，优化后的方法减少了节点与节点之间信息传输的次数，因此可以降低达成共识的时间且提升系统的性能。

(3)兼容性：本方案是对区块链的共识算法进行改进，共识算法是区块链的核心技术之一，可以在以太坊中、联盟链、或者其他区块链平台中直接进行应用。

本方案的缺点是由于主节点一方面要接收来自客户端的信息，另一方面还要对节点共识的结果进行统计并回复给客户端，一旦节点数量非常大，主节点容易过载。除此以外，如果主节点是恶意节点，会对整个系统的共识结果造成重大的影响，但是由于主节点是整个网路中可信度最高的节点，因此主节点是恶意节点的可能性大大降低。

2 NRPBFT 算法整体流程

本节以上一节提出的 NRPBFT 算法方案为基础，描述了 NRPBFT 算法的整体流程，给出了更详细的设计方案，包括一些关键模块的实现方法和代码分析，同时介绍了 Hyperledger Fabric 的搭建，以及编写链码验证算法改进的结果。

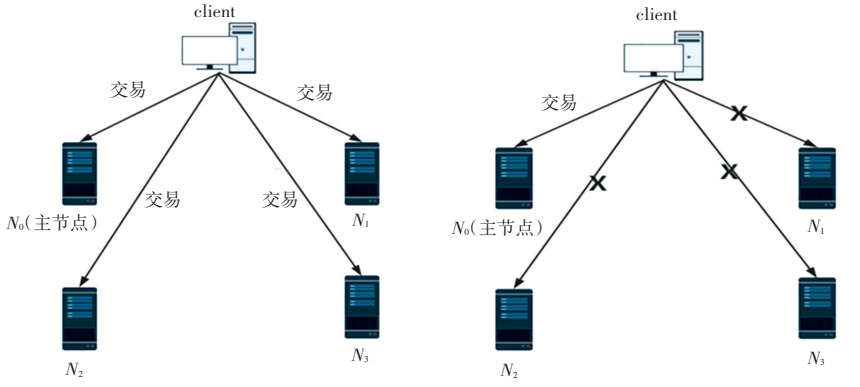
2.1 NRPBFT 算法整体流程

NRPBFT 算法按照运行过程可以分成 2 部分。一部分是共识过程，另一部分是视图更换过程。一旦共识过程出现问题会立即触发视图更换过程。对此拟展开研究分述如下。

(1)共识过程

① 准备阶段。这一步主要目的是让客户端知道哪个节点是主节点。在交易请求阶段，客户端将交易发送给所有节点，节点接收到交易信息后，验证签名，验证成功后判断交易是否为 query 类型的交易，如果是，则执行智能合约，查询本地账本的余额信息并返回结果，这个过程就结束了。如果是 deploy 或 invoke 类型的请求交易，则需要利用共识机制来得到结果。对于需要共识的交易，普通节点是无法处理的，只有主节点可以接收并进行回复，因此，在交易请求阶段相当于客户端把需要共识的交易信息发送给了所有节点，在验证阶段，只有主节点能接收，也就是选出了主节点。主节点接收到交易信息后，对信息进行编号，提取摘要并签名后发送给共识集群中的节点进行共识，进入认证共识阶段。准备阶段拓扑见图 3。

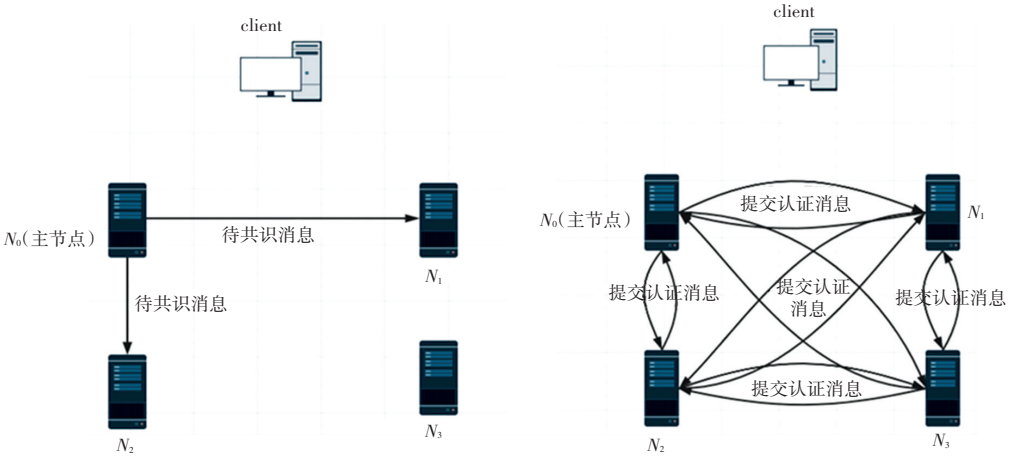
② 认证共识阶段。在认证阶段，共识集群中节点收到主节点发送的信息后，对信息的签名、摘要、编号进行验证，验证通过后放入内存，对信息进行签名并打包生成提交报文，在整个网络的节点中进行广播，进入提交认证阶段。在任何一个阶段如果验证失败或者接收信息超时，则认为主节点可能是恶意节点，触发视图更换协议对主节点和共识集群中的节点进行更换并重启共识过程。认证共识阶段拓扑见图 4。



(a) 交易请求阶段　　(b) 验证阶段

图 3　准备阶段拓扑图

Fig. 3　Topology diagram of preparation phase



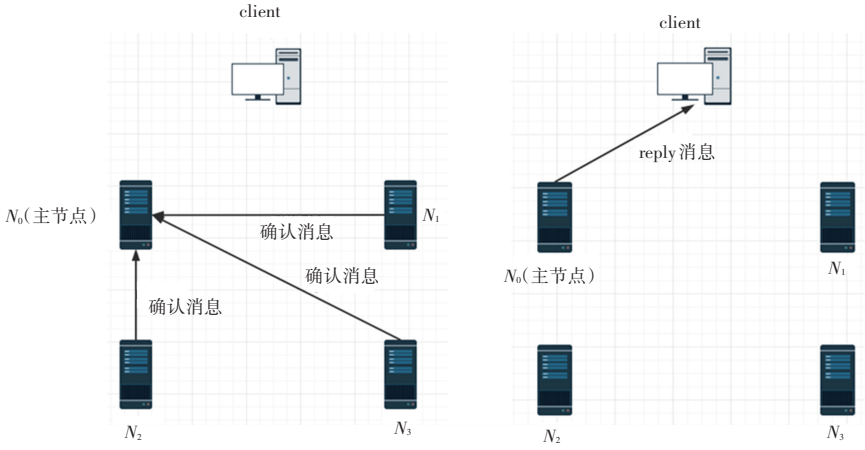
(a) 认证阶段　　(b) 提交认证阶段

图 4　认证共识阶段拓扑图

Fig. 4　Topology diagram of authentication consensus phase

在提交认证阶段，共识节点将信息在整个网络中进行广播，普通节点收到广播信息后，对信息的签名、摘要、视图编号进行验证，验证通过后放入内存，一旦普通节点收到了 $p/2$ 个验证通过的提交认证消息，生成提交报文，提交给主节点，进入确认提交阶段。在任何一个阶段如果验证失败或者接收信息超时，则认为主节点可能是恶意节点，触发视图更换协议对主节点和共识集群中的节点进行更换并重启共识过程。

③ 确认提交阶段。确认提交阶段拓扑见图 5。



(a) 确认阶段　　(b) 提交阶段

图 5　确认提交阶段拓扑图

Fig. 5　Topology diagram of confirmation commit phase

(2)视图更换过程。触发节点更换的条件主要有以下几点:

① 过了指定时间,普通节点或者共识集群的节点未收到主节点的共识请求。

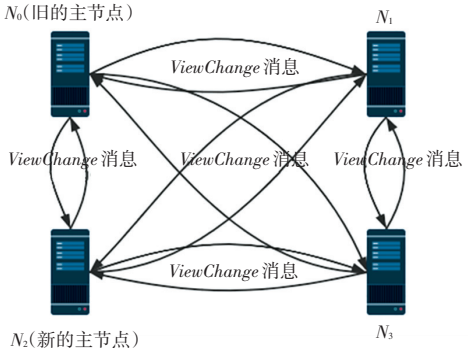
② 信息的格式不合法。

③ 没有在规定时间内完成主节点更换。

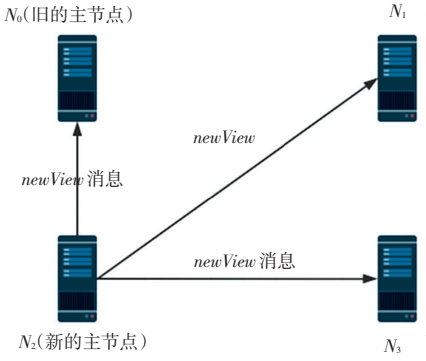
④ 在共识过程中,多个节点的视图编号是旧的编号或者签名摘要验证失败。

在 NRPBFT 算法中,由原 PBFT 算法根据节点编号顺序选举主节点改变为利用节点的可信度来选择主节点,这种做法避免了主节点随意选举带来的风险,同时大大降低了主节点成为恶意节点的可能性。

在主节点更换阶段,视图编号自加 1 作为新的视图编号,所有节点封装 ViewChange 报文,报文中封装了各个节点的可信度和签名,在网络中进行广播,如图 6(a)所示。



(a) 主节点选举阶段



(b) 主节点确定阶段

图 6 视图更换拓扑图

Fig. 6 View replacement topology

节点接收到其他节点的广播信息后,选出可信度最高的节点作为主节点,如果 $2f+1$ 个节点选出的可信度最高的节点是同一个,则认为这个节点就是下一个主节点。在这里,每个节点都会计算除了主节点外可信度最好的 p 个节点,组成该视图下的共识集群。在主节点确认阶段,新的主节点封装 *newView* 报文并签名,在网络中进行广播。如此,新的主节点就更换完成,如图 6(b)所示,N_2 节点是可信度最高的节点,就会被选举为下一个主节点,接下来重新进行共识过程。

2.2 NRPBFT 算法结构及实现

整个 NRPBFT 算法的代码各部分功能见表 2。

表 2 NRPBFT 各模块功能表

Tab. 2 Function table of each module of NRPBFT

方法名称	方法功能
client.go	模拟并运行客户端
cmd.go	封装各阶段标志位和计算摘要方法
main.go	定义了节点及其端口号,主程序入口,可以启动客户端和节点
newPbft.go	封装了 NRPBFT 运行的各个阶段的代码,是整个程序最核心的模块
rsa.go	生成 *RSA* 公私钥对,生成或验证签名
tcp.go	定义了节点使用 *tcp* 进行客户端监听及消息发送
voteNotes.go	定义了主节点的更换过程和共识集群的选举过程

下面对 NRPBFT 算法的具体实现进行描述,包括数据结构的类型定义和相关的实现,还有关键变量和关键函数的调用过程。

(1)数据结构定义。节点的定义如图 7 所示。其中,*nodeID* 用来存储属于节点的编号,*addr* 用来存储节点的 ip 地址+端口号,*rsaPrivKey* 和 *rsaPubKey* 分别存储节点的公私钥,*credit* 存储节点的可信度的值。

```
type node struct {
    nodeID string
    addr string
    rsaPrivKey []byte
    rsaPubKey []byte
    credit int64
}
```

图 7 节点定义图

Fig. 7 Node definition graph

RPBFT 信息结构的定义如图 8 所示。其中,*node* 存储的是上述的节点的数据结构信息,*sequenceID* 存储的是交易的序号,会随着交易数量的增加而自增。*Lock* 定义了一个锁,在多线程操作的过程中可以保证获取数据的一致性和准确性。*messagePool*、*prePareConfirmCount*、*commitConfirmCount* 分别存储消

息本身、准备阶段的消息和确认消息。*isCommitBordcast* 是用来判断是否进行过广播的标志位，*isReply* 是用来判断是否回复信息给主节点的标志位。

```
type pbft struct{
    node node
    sequenceID int
    lock sync.Mutex
    messagePool map[string]Request
    prePareConfirmCount map[string]map[string]bool
    commitConfirmCount map[string]map[string]bool
    isCommitBordcast map[string]bool
    isReply map[string]bool
}
```

图 8　RPBFT 信息结构定义图

Fig. 8　RPBFT information structure definition diagram

(2)变量定义。该部分介绍在代码中定义的全局变量及其含义。变量定义见表 3。由表 3 可知，*nodeCount* 定义了节点的数量，根据这个变量值，生成 *nodeCount* 个节点的公私钥对。*nodeCredit* 定义了共识集群节点的数量，在共识过程中也是根据这个值判断是否收到了所有共识节点的消息。*clientAddr* 定义了客户端的监听地址，*nodeTable* 和 *credit_nodeTable* 都是 *map* 类型的数据，分别存储节点和地址的键值对及共识集群中的节点和地址的键值对。*primaryNode* 记录主节点的 *id*，记录每一个视图下主节点的 *id*。*localMessagePool* 是一个未定义的数组，用于存储客户端发送的消息。*start* 和 *end* 分别记录算法的开始时间和结束时间，二者作差便可以得到算法的运行时间，*prefixCMDLength* 是命令名称的长度，凭借这个值取得消息的前 12 位，分析前 12 位的消息即可得出目前处于共识的哪个阶段。

表 3　合约变量定义表

Tab. 3　Contract variable definition table

变量名称	数据类型	说明
nodeCount	const	定义节点数量
nodeCredit	const	定义共识节点数量
clientAddr	string	客户端的监听地址
nodeTable	map[string]string	储存所有节点和地址的键值对
credit_nodeTable	map[string]string	储存共识节点和地址的键值对
primaryNode	string	记录主节点的 id
localMessagePool	[]Message{}	存储客户端发送的消息
start	time.Time	记录算法开始时间
end	time.Time	记录算法结束时间
prefixCMDLength	const	定义前 12 位为命令名称

(3)功能方法。关键方法定义及介绍见表 4。

3　实验结果

3.1　实验环境

本文的 Windows 实验环境的参数见表 5，基于此环境配置，利用端口号模拟节点，实现了 NRPBFT 算法的运行。

表 4　关键方法定义表

Tab. 4　Key method definition table

函数名称	输入参数	说明
clientSendMessageAndListen	无	开启客户端的本地监听，封装并发送待共识信息给主节点
handleClientRequest	*content* []*byte*	该方法传入的 *content* 就是待共识的消息，主要是主节点验证处理待共识消息，并打包后广播给共识集群中的节点
handlePrePrepare	*content* []*byte*	该方法传入的 *content* 是主节点封装后的待共识消息，共识集群的节点验证并处理消息，打包后在整个网络中进行广播
handlePrepare	*content* []*byte*	该方法传入的 *content* 是共识节点封装后的待共识消息，所有节点收到 $p/2$ 个共识节点发送的消息，认为共识成功，发送共识成功的消息给主节点
handleCommit	*content* []*byte*	该方法传入的 *content* 是所有节点共识成功后发送给主节点的信息，主节点收到至少 $2f+1$ 个节点的共识信息后，回复给客户端，共识完成

表 5　设备参数表

Tab. 5　Equipment parameters table

Name	CPU/内存	CPU 型号	操作系统	系统类型
客户端	2 核/8.00 GB	Intel(R) Core(TM) i7-9700	Windows 10	64 位操作系统
共识节点	2 核/8.00 GB	Intel(R) Core(TM) i7-9700	Windows 10	64 位操作系统
普通节点	2 核/8.00 GB	Intel(R) Core(TM) i7-9700	Windows 10	64 位操作系统

运行上一节描述的 NRPBFT 算法和原 PBFT 算法,这里以 4 个网络节点为例,分别运行优化前后的算法。在 PBFT 算法中, N_0 是主节点,N_1、N_2、N_3 是普通节点。在 NRPBFT 算法中,设置 4 个节点情况下,只能存在一个恶意节点,根据 p 的范围为 $2f \leqslant p < 3f$,这里选择节点 N_1 和节点 N_2 组成共识集群,主节点为节点 N_0,节点 N_3 是普通节点。

PBFT 算法 4 个节点时的执行时间见图 9。

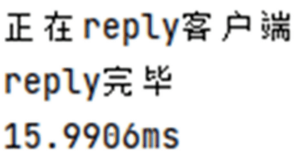


图 9 PBFT 算法执行时间

Fig. 9 Algorithm execution time of PBFT

NRPBFT 算法 4 个节点时的执行时间见图 10。

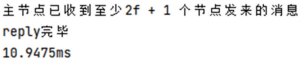


图 10 NRPBFT 算法执行时间

Fig. 10 Algorithm execution time of NRPBFT

从上面的对比结果可以看出,本项研究对 PBFT 算法的修改可以有效减少节点之间达成共识的时间,提高了共识效率。但是节点数量只有 4 个的情况太过简单,尤其是在 NRPBFT 算法中,普通节点只有 N_3,并不能很好地体现算法优化的程度。PBFT 算法的特点是随着网络规模的增大,通信时延呈指数增长,因此可以增大网络规模做后续进一步研究。

将节点的数量分别设置成 4 个、6 个、8 个、10 个、12 个、14 个、16 个、18 个、20 个,查看在不同节点数量下算法改进前后达成共识所需的时间。节点数量和达成共识的时间见表 6。

表 6 改进前后算法执行时间表

Tab. 6 Algorithm execution schedule before and after improvement

节点数量/个	改进前时间/ms	改进后时间/ms
4	15.990 6	10.947 5
6	23.907 6	10.702 1
8	37.899 7	11.968 2
10	57.379 1	13.016 8
12	80.095 0	14.960 0
14	111.745 2	15.957 5
16	143.911 8	20.944 0
18	155.184 9	30.920 2
20	187.754 3	36.873 4

将改进前和改进后的算法运行时间绘制在一张图上,可以更直观地看出改进前后性能的优化,节点数量和达成共识的时间对比如图 11 所示。

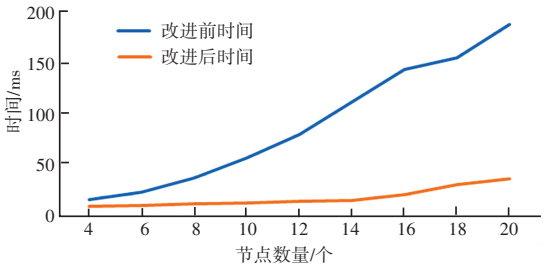


图 11 改进前后 PBFT 算法执行时间与网络规模关系图

Fig. 11 The relationship between the execution time of the PBFT algorithm and the network size before and after the improvement

通过实验结果可知,NRPBFT 算法相比 PBFT 算法的改进效果比较明显。从整体上分析,在相同节点数量和相同交易信息的条件下,NRPBFT 算法的时延总是低于 PBFT 的时延。从趋势上看,随着网络中节点数量的增加,网络中节点与节点之间的通信数量增加,NRPBFT 算法和 PBFT 算法的时延都增大了。这时候影响算法效率的是节点之间的通信开销,并且 NRPBFT 算法减少通信次数的优点也体现出来了。从图 11 中可以明显看出,随着节点数量的增加,NRPBFT 算法时延增长速度比 PBFT 的时延增长速度慢。在 4 个节点 1 笔交易的条件下,NRPBFT 算法的时延为 10.063 8 ms,PBFT 算法的时延为 16.709 6 ms,NRPBFT 算法相对 PBFT 算法时延降低了 39.77%。

为了说明其防作弊的特点,这里还需要模拟恶意节点,在网络中节点总数为 4 的情况下进行模拟。具体来说, N_0 为主节点,N_1 和 N_2 为正常节点,N_3 为恶意节点,模拟的时候故意不开启 N_3,模拟节点宕机的情况。

在 PBFT 中,当不开启 N_3 节点的时候,经测试,节点依然可以达成共识,达成共识的时间见图 12。

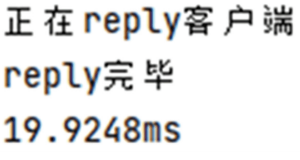


图 12 有恶意节点时 PBFT 算法执行时间

Fig. 12 PBFT algorithm execution time when there are malicious nodes

在 NRPBFT 中,当不开启 N_3 节点的时候,经测试,节点依然可以达成共识,达成共识的时间见图 13。

```
主节点已收到至少2f + 1 个节点发来的消息
reply完毕
19.9474ms
```

图 13　有恶意节点时 NRPBFT 算法执行时间

Fig. 13　NRPBFT algorithm execution time when there are malicious nodes

从图 13 可以看出，在 N_3 为恶意节点的情况下，2 个都能达成共识，并且由于在 4 个节点的情况下，如果有一个恶意节点，则剩余的节点都是共识集群中的节点，和 PBFT 算法没有区别，因此达成共识的时间也非常接近。实验结果说明，当网络中存在 $f(n > 3f)$ 个节点时，2 种算法都可以防止节点作弊，在规定时间内达成共识。

3.2　通信次数分析

除了通信时间上有了直观改善，在通信数量上也减少了很多。这里定义网络中节点总数为 n，定义 NRPBFT 算法中共识集群中节点数量为 p。

在 PBFT 算法中，核心有 3 个阶段。在预准备阶段，主节点将待共识信息进行广播，这一阶段的通信次数为 $n-1$ 次；进入到准备阶段，每个节点进行一次广播，一共 n 个节点，因此这一阶段的通信次数为 $n(n-1)$；进入提交阶段，依然是每个节点进行广播，通信次数为 $n(n-1)$。因此，PBFT 算法总的通信次数为 $2n^2-n-1$。

在 NRPBFT 算法中，核心阶段也是有 3 个。在认证阶段，主节点向共识集群中的节点发送待共识信息，这一阶段的通信次数为 p 次；进入到共识阶段，每个共识节点进行一次广播，一共 p 个共识节点，因此这一阶段的通信次数为 $p(p-1)$；进入确认阶段，这里是共识节点和主节点在整个网络中进行广播，通信次数为 $n(p+1)$。因此，NRPBFT 算法总的通信次数为 p^2+np+n，这里 p 取 $(2/3)n$，遇到小数上取整，即总通信次数为：$(10/9)n^2+n$。

将节点的数量分别设置成 4 个、6 个、8 个、10 个、12 个、14 个、16 个、18 个、20 个，查看在不同节点数量下的通信次数。节点数量和通信次数的关系见表 7。

将改进前和改进后的算法通信次数绘制在一张图上，可以更直观地看出改进前后的优化，节点数量和通信次数的时间对比如图 14 所示。

通过实验结果可知，NRPBFT 算法相比 PBFT 算法的改进效果比较明显。从整体上分析，在相同节点数量和相同交易信息的条件下，NRPBFT 算法的通信次数总是比 PBFT 的通信次数少。从趋势上看，随着网络中节点数量的增加，网络中节点与节点之间的通信数量都在增加，但是 NRPBFT 算法中通信次数增加得比较缓慢。

表 7　改进前后算法通信次数表

Tab. 7　Algorithm communication times table before and after improvement

节点数量/个	改进前通信次数/次	改进后通信次数/次
4	27	21
6	65	46
8	119	79
10	189	121
12	275	172
14	377	231
16	495	300
18	629	378
20	779	464

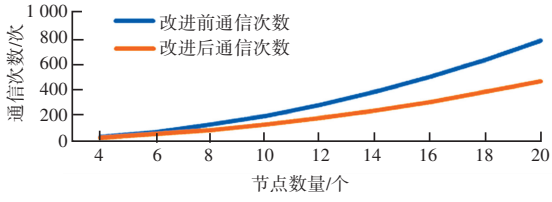


图 14　改进前后 PBFT 算法执行时间与通信次数关系图

Fig.14　The relationship between the execution time of the PBFT algorithm and the number of communications before and after the improvement

3.3　ubuntu 环境下性能测试

编写链码，进行交易的初始化和转账操作，部署链码，进行测试。PBFT 算法共识时间的运算见图 15。由图 15 可知，在采用原 PBFT 算法的时候，交易转账共识所需要的时间为 0.029 s。

```
ex02 Invoke
Query Response:{"Name":"A","Amount":"678"}
ex02 Invoke
Query Response:{"Name":"B","Amount":"567"}
PASS
ok      _/home/hanlu/go/src/github.com/hyperledger/fabric/examples/chaincode/go/chaincode_example02     0.029s
```

图 15　PBFT 算法共识时间

Fig. 15　Consensus time of PBFT algorithm

NRPBFT 算法共识时间的运算见图 16。由图 16 可知，采用优化后的 NRPBFT 算法的时候，交易转账共识所需要的时间为 0.005 s。

```
ex02 Invoke
Query Response:{"Name":"A","Amount":"678"}
ex02 Invoke
Query Response:{"Name":"B","Amount":"567"}
PASS
ok      _/home/hanlu/go/src/github.com/hyperledger/fabric/examples/chaincode/go/chaincode_example02     0.005s
```

图 16　NRPBFT 算法共识时间

Fig. 16　Consensus time of NRPBFT algorithm

在采用原 PBFT 算法的时候，交易转账共识过

程的性能指标见图 17。

图 17 PBFT 算法性能

Fig. 17 Performance of PBFT algorithm

采用优化后的 NRPBFT 算法的时候,交易转账共识过程的性能指标见图 18。

图 18 NRPBFT 算法性能

Fig. 18 Performance of NRPBFT algorithm

从图 18 可以看出, NRPBFT 算法的吞吐量更大,性能也提升了。

4 本文研究结论

4.1 共识节点数量取值证明

在本项研究中, 选举了 p 个除了主节点外可信度最高的节点组成共识集群, p 的取值范围是 $2f \leqslant p < 3f$。接下来,对 p 的取值范围进行证明,说明为什么在这个范围内可以保证系统中的节点正常达成共识。

要想证明 P 在 $2f \leqslant p < 3f$ 的情况下能保证网络在有 f 个恶意节点的情况下达成共识,首先要证明 PBFT 满足的如下条件:

证明 假设节点总数为 N, f 为拜占庭错误节点,N 满足:$N \geqslant 3f + 1$。

如果网络中一共有 N 个节点,其中拜占庭节点的数量为 f,那么非拜占庭节点的数量就是 $N - f$。在共识的时候,如果一个普通节点收到 $N - f$ 个信息,节点无法判断这 $N - f$ 个信息来自拜占庭节点、还是非拜占庭节点。考虑最坏的情况,这 $N - f$ 个信息中有 f 个信息来自拜占庭节点,在这种情况下,只有 $N - 2f$ 个信息来自非拜占庭节点。对于这个普通节点,在收到 2 种不同的信息,节点会按照少数服从多数的原则,选择信息数量多的信息作为待共识信息。因此,要想保证待共识信息是非拜占庭节点发送的信息,就需要保证 $N - 2f > f$,即 $N > 3f$,也就是 $N \geqslant 3f + 1$。

在 NRPBFT 算法中, p 的范围是 $2f \leqslant p < 3f$,在认证共识阶段,主节点只会将待共识信息发送给 p 个共识集群中的节点进行共识。由于这 p 个节点是网络中除了主节点外可信度最高的 p 个节点,所以在一定程度上可以保证 p 个节点中拜占庭节点的数量比较少,也就是可以保证 $p - f > f$,即 $p > 2f$。在提交阶段,p 个节点会分别向 N 个节点发送信息,也就是进行 p 次单点全广播的过程,当一个节点收到至少 $p/2$ 个相同的信息时,也就是至少 f 个相同的消息,显然这些消息不可能全是恶意节点发出的,只能是非拜占庭节点发出的消息。在确认阶段的证明和 PBFT 算法一致,这里不再重复。

由上述论证可以说明,在公式集群 p 的节点数量满足 $2f \leqslant p < 3f$ 时, NRPBFT 算法可以保证能在有 f 个恶意节点的情况下,网络中的非拜占庭节点达成共识。

4.2 算法防作弊特性说明

接下来对算法的防作弊特点进行说明。由于本项研究中默认恶意节点数量 f 和网络中节点总数 n 之间满足 $n > 3f$ 的关系,且默认诚实节点一定能正常进行共识,不会出现信息丢失和宕机等情况,因此,本算法一定可以抵御 51%攻击。

在区块链中,DDoS 攻击的主要目的是大量占用网络中的节点资源,使得这些节点无法提供正常的服务,如果受害的节点过多,很可能会影响整个区块链网络的运行。DDoS 攻击是通过攻击手段占用了受害者的大量资源,使得受害者不能提供正常服务。

在区块链中,DDOS 攻击主要是计算集群对网络中的节点进行攻击,占用节点资源,使得节点宕机,无法进行共识。本课题中,最多只有 f 个节点可以遭受 DDOS 攻击,而剩余的 $n - f$ 个诚实节点依然可以正常达成共识,不会对最后的结果造成影响。因此,本算法可以抵挡 DDOS 攻击。

拜占庭将军问题所描述的正是共谋攻击的情境,恶意节点联合起来修改信息或者丢失信息,以此影响网络中的节点达成共识。在上述证明中,已经说明为什么 PBFT 算法和 NRPBFT 算法能够在共谋攻击的情况下达成共识,但是前提必须是 $n > 3f$,恶意节点的数量再多,就无法抵御共谋攻击了。

女巫攻击是一种专门针对联盟链的攻击,恶意节点可以伪装自己的身份来进行攻击。由于 PBFT 机制的延展性不高,一般只能用在网络规模小的网络中,所以更易受到女巫攻击。在 NRPBFT 算法中,对节点的可信度进行评估,选取可信度最高的节点作为主节点,除主节点外可信度最高的 p 个节点作为共识集群,核心的阶段都在共识集群中进行,最后只需要将共识集群共识完成的信息发送给普通节点,避免了恶意节点过多参与共识过程,减少了恶意节点作恶的机会。