第51卷 第7期
2021年7月

中国海洋大学学报
PERIODICAL OF OCEAN UNIVERSITY OF CHINA

51(7):092~100
July, 2021

一种可应用于联盟链的拜占庭容错混合共识机制*

周 炜[1,2], 袁晓伟[1]**, 魏志强[3], 翟翌立[2], 王 超[1], 杜丙瑜[1], 朱文印[2], 王金龙[1]

(1.青岛理工大学信息与控制工程学院, 山东 青岛 266100; 2.海尔集团技术研发中心, 山东 青岛 266100;
3.中国海洋大学信息科学与工程学院, 山东 青岛 266100)

摘 要: 拜占庭容错(BFT)在提供分布式系统的可靠性方面将变得越来越重要,其中实用性拜占庭容错(PBFT)是目前用的最佳算法之一,但是面对具有众多节点的分布式系统,PBFT并不令人满意。由于PBFT共识失败率接近1/3,共识失败率较高会造成主节点切换频繁,拖慢共识效率,概率分组算法降低了共识失败率,从而提高了共识效率。为了防止拜占庭节点串通攻击分组,基于可验证随机函数VRF抽签方式让拜占庭节点无法预判分组,进一步提高共识安全性。混合共识机制组内采用拜占庭容错chain-raft加快共识效率,组间采用PBFT保证对现有PBFT共识的兼容性。本文提出的共识机制在16个节点分为四组的情况下消息复杂度比PBFT降低了92%,吞吐量是PBFT的3.6倍。

关键词: 实用拜占庭容错;联盟链;分组混合共识;概率分组;随机分组

中图法分类号: TP311.13 **文献标志码:** A **文章编号:** 1672-5174(2021)07-092-09

DOI: 10.16441/j.cnki.hdxb.20200250

引用格式: 周炜, 袁晓伟, 魏志强, 等. 一种可应用于联盟链的拜占庭容错混合共识机制[J]. 中国海洋大学学报(自然科学版), 2021, 51(7): 92-100.

Zhou Wei, Yuan Xiaowei, Wei Zhiqiang, et al. A byzantine fault tolerant hybrid consensus mechanism applicable to consortium blockchain[J]. Periodical of Ocean University of China, 2021, 51(7): 92-100.

伴随着区块链的发展,截至目前块链大致可以分为三类:公有链、联盟链、私有链。而不同类型的链会有不同的共识,公有链共识大致可以分为三类:基于算力竞争记账权的POW共识[1]、基于权益证明的POS共识[2]、基于POW共识和POS共识的改进,例如Casper FFG共识[3]、POA共识[4]、POSV共识[5]。联盟链共识大致分为两类:一类是非拜占庭容错共识,这类共识不考虑节点采取恶意攻击行为,只考虑崩溃节点不工作不回应的情况,例如PAXOS共识[6]、RAFT共识[7]。另一类是拜占庭容错共识,这类共识会考虑超时错误、回应错误信息、节点串通攻击等情况,例如PBFT共识。私有链与以上两种链不同,它大部分被使用于组织与集体内部不对外公开,不过共识大部分采用PBFT、PAXOS、RAFT等共识。

目前无论是社会结构还是网络服务结构都是由几个大型组织作为中心,完全去中心化显然不符合目前发展阶段,多个组织合作的联盟链比较符合当下的发展。目前一部分联盟链使用非拜占庭容错共识。非拜占庭共识的优点是交易吞吐量大、延迟低。但是非拜占庭容错共识无法防止节点对系统的恶意攻击,考虑到联盟链是多个组织参与,因此这类共识在实际场景中安全性很低,没有实用价值。考虑到联盟链系统安全性问题,本文针对拜占庭容错算法进行研究,选取目前应用最普遍的PBFT共识[8]。PBFT是一种基于部分异步网络模型的拜占庭容错算法,它通过三阶段提交过程保证了节点之间的一致性,通过节点之间相互传递签名消息保证安全性,通过超时检测和主节点切换保证活性。一致性保证了节点之间确认的消息是一样的,安全性保证了节点可以判断消息是否正确,活性保证了系统不会出现一直等待不工作的状况。

虽然拜占庭容错PBFT共识目前发展的比较成熟,但目前还存在一些问题。首先,虽然节点两两之间相互通信可以保证节点判断消息的正确性,但造成通信复杂度过高($(O(n^2))$)。由于消息确认需要两次且随着节点数量变多,通信复杂度会增长过快,进而影响系统可扩展性。其次,虽然PBFT共识采用主从模式保证了交易顺序的一致性,但是导致了通信会集中于主节点,引起主节点通信压力过大。最后,PBFT共识失败率高。采用主节点轮次交换的方式,而在$3f+1$个节点中,如果有f个拜占庭节点,轮换到错误节点的概

* 基金项目:国家自然科学基金项目(61502262);青岛市博士后应用研究项目资助
Supported by the National Natural Science Foundation of China(61502262); the Qingdao Postdoctoral Applied Research Project
收稿日期:2020-08-30;修订日期:2020-10-02
作者简介:周 炜(1981-),男,副教授,研究方向为区块链技术、信息安全。E-mail:zhouwei@qut.edu.cn
** 通讯作者:E-mail:yuan199501@hotmail.com

率接近 1/3，共识失败率较高，会导致主节点切换频繁，共识效率下降。

随着现代计算机和互联网的发展，数据规模不断增长，处理的数据量越来越大，单一节点处 理数据和存储数据压力越来越大，分布式系统在提高数据处理效率同时保证了系统可靠性，缓解节点存储压力。不过分布式系统在多节点合作协调一致性方面存在技术挑战，因此分布式系统中如何保证共识是个经典的技术问题。

共识协议从大的方面可分为自比特币诞生以来产生的 POW 等中本聪共识协议和拜占庭容错（BFT）类共识协议这两大类协议。其中，在计算机科学的分布式系统研究领域，近年来，越来越多的联盟链平台大部分集中在对 BFT 类共识协议的研究。由此本文对 BFT 类协议进行了综述性概览分析，并从改进思路上将现有 BFT 类共识协议分为三大类：针对无拜占庭错误场景优化的协议、针对拜占庭错误场景优化的协议以及为公链应用而优化的协议。

针对无拜占庭错误场景进行优化又分为基于协约方法的优化，例如 Chain 协议[9]、Ring 协议[10]、BFT-SMaRt 协议[11]；基于 Quorum 方法的优化，例如 Q/U 协议（Query/Update）[12]、HQ 协议（Hybrid Quorum）[13]、Quorum 协议[14]；基于 Speculation 方法的优化，例如 Zyzzyva 协议[15]、Zeno 协议[16]、ZZ 协议[17]；基于客户端方法的优化，例如 OBFT（Obfuscated BFT）协议[18]；基于可信组件方法的优化，例如 CheapBFT 协议[19]；基于拜占庭锁方法的优化，例如 Zzyzx 协议[20]；基于分离一致性与执行请求方法的优化，例如 ODRC 协议[21]。

上面介绍的一类协议均是针对没有错误的场景对 BFT 协议进行简化而设计，因此当遇到拜占庭错误时，这类协议的性能一般都会下降比较多，甚至很难保证系统活性。而另一类针对有错误的场景对 BFT 协议的改进目的是为了有效对拜占庭行为（甚至是一些罕见的行为）进行容错，降低系统在有无拜占庭错误这两种场景下的表现差异。其中比较典型的协议，例如 Aardvark 协议[22]、Prime 协议[23]、Spinning 协议[24]、Redundant-BFT 协议[25]。

传统 PBFT 及类似协议，其自身的特性导致应用场景有较多限制，例如消息复杂度随节点数成平方级别上升、主节点容易成为系统性能瓶颈、节点列表需要提前固定且节点间相互已知。所以在分布式账本系统中，更多应用于联盟链或私有链场景中。为了适应公有链场景中的大规模扩展需求，有不少项目进行了有益尝试，具体方式可主要包括与公链共识结合以及基于密码学机制等两大类方式。与公链共识结合的典型协议，例如 DPOS＋BFT 协议[26]、DBFT 协议[27]、Tendermint BFT 协议[28]、FBA 协议[29]；基于密码学优化的典型协议，例如基于聚合签名的 Zilliqa[30]、基于可验证随机函数的 VBFT 协议[31]、基于门限签名的 Honey-Badger BFT[32]。

通过以上分析发现，区块链共识算法及其变种都集中在 PBFT 算法上，所以本文选取了部分同步网络模型的 PBFT 算法作为研究对象。

PBFT 算法是一种状态机副本复制算法，所有副本在视图编号 v 的轮转中运行，视图是连续编号的整数，在同一视图编号下，节点角色被分为客户端、主节点和从节点。客户端是请求的发送方，主节点负责接收客户端的请求，并且按照顺序对请求进行排序，从节点主要是负责检查从主节点接收到的请求，并且通过超时机制检测主节点是否宕机，当发生出节点宕机时，触发视图切换机制。

结合 PBFT 共识存在拜占庭节点情况下，共识失败率较高，节点数量多主节点通信压力过大，整体共识通信复杂度增长过快的问题，本文提出了一种分组混合共识机制，首先通过概率分组算法找出共识概率达成最高的分组方式，然后所有节点基于可验证随机函数 VRF[31]通过两轮消息投票方式确认分组信息，接下来分组内部开始进行拜占庭容错 chain-raft 共识，按照下文中设计的算法流程在分组内达成共识，最后，分组主节点之间按照 PBFT 共识流程达成最终共识。分组混合共识机制其中概率分组算法提高共识达成概率，减少主节点切换次数从而提高在存在拜占庭错误情况下共识性能；随机分组过程利用可验证随机函数 VRF 确保节点之间不能串通，保证了共识的活性与安全性；混合共识算法组内使用拜占庭容错 chain-raft 共识，引入聚合签名使得通信复杂度为 $O(n)$ 并且优化了共识流程，缩短了整体共识时间；组间继续使用 PBFT 共识，保证分组之间拜占庭容错，进一步加强共识安全性。

针对 PBFT 共识存在的问题，本文提出了一种适合联盟链的分组混合共识机制。之所以适合联盟链，是因为联盟链是准入制可以知道系统中有多少个参与节点，有利于分组构建。之所以分组，是因为节点进行分组，虽然增加了一个分组过程，但是重新分组只会发生在最终共识达成失败的情况，根据本文概率分组算法，共识失败率低于 PBFT 共识，因此缓解了共识失败主节点切换频繁造成的通信压力，并且此分组过程也保证了系统活性。然后分组之后每组都有主节点，此方式会缓解主从模式主节点作为通信中心的压力。随机分组让拜占庭节点无法预知分组结果进而串通攻击分组，保证了系统安全性。之所以要混合共识，一方面

本文拜占庭容错 chain-raft 算法在拜占庭容错的基础上通信复杂度为($O(n)$),并且借鉴 HotStuff 共识[33]流水线化的共识流程,让先后两笔交易不同阶段同时进行,缩短了上述共识算法的时间。另一方面,考虑到联盟链目前大部分使用 PBFT 共识,本文设计的共识机制可以作为 PBFT 共识的一种扩展,这样可以很好的兼容当前的 PBFT 共识。

本文的贡献在于:一是采用概率分组算法,提高共识达成概率;二是采用随机分组过程,防止拜占庭节点串通,保证共识安全性;三是提出一种新的组内拜占庭容错 chain-raft 共识,在拜占庭容错的基础上引入共识流程并行化,优化共识流程,组间继续使用 PBFT 共识算法,保证了分组拜占庭容错和对已有 PBFT 共识算法的兼容性。

1 分组混合共识机制

本文构建了基于分组混合共识机制,如图 1 所示其中分为三个阶段,分别进行介绍。

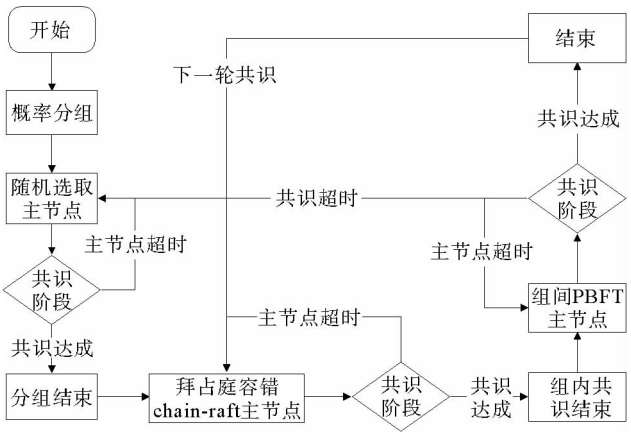

图 1 分组混合共识机制架构图

Fig.1 Architecture diagram of grouped hybrid consensus mechanism

概率分组阶段:PBFT 共识算法若有 $3f+1$ 个节点,最多可容纳 f 个拜占庭节点,共识失败率最高接近 1/3,导致频繁触发 viewchange 切换主节点拖慢系统性能。根据概率分组可以降低共识失败率提高系统性能。

随机化分组阶段:PBFT 共识算法随着节点数量的增加,节点间通信复杂度增长过快,主节点通信压力会越来越大,分组后先组内共识,然后组间共识,降低了通信复杂度同时减缓了通信复杂度增长速度,组内有主节点,组间有主节点,降低了单一主节点通信压力。随机化分组的目的是为了防止拜占庭节点可能会串通攻击一定数量的分组破坏共识达成,因此使用可验证 VRF 随机函数随机分组让每个节点无法预先判断分到哪个组,从而保证共识活性。

混合共识阶段:如果只是 PBFT 共识分组,组内还是 PBFT 共识,通信复杂度为 $o(n^2)$,因此组内使用拜占庭容错的 chain-raft 共识,通过聚合签名确认消息,并且并行化共识阶段,共识通信复杂度为 $o(n)$,进一步降低了系统通信复杂度,而且组间 PBFT 共识,保证分组间拜占庭容错和已有 PBFT 共识算法的兼容性。

1.1 概率分组算法

概率分组算法的步骤:

第一步:先求解出组数满足 $N>=3f+1$ 和组内节点数满足 $N>=3f+1$ 的分组方式求解算法如表 1。

表 1 分组算法

Table1 Grouping algorithm

```
def divN(n,k): //n是节点个数 k是组数
global times
    for num in range(4,n+1):
        if(num >= res[k-1]):
            res[k] = num
            rest = n-num
                if(rest == 0):
                    times = times+1
                    ifk >= 4:
                        foriinrange(1,k+1):
                            print(res[i],end = ")
                        print('\n')
                else:
                        divN(rest,k+1)
```

第二步:根据已经求出的分组方式,求出该分组方式的拜占庭容错节点个数,求出该分组方式达成共识的概率,例如总共有 16 个节点,分为 4 组,每组有 4 个节点,该分组方式拜占庭容错节点个数为 5 个,达成共识的概率为

$$\left(1-\frac{c_5^2\ c_3^2\ c_{12}^2\ c_{10}^2\ c_8^4}{c_{16}^4\ c_{12}^4\ c_8^4}\right)\approx 90\%\ 。\tag{1}$$

第三步:在满足达成共识概率最高和容纳拜占庭节点最多的条件下得出分组方式。

本文概率分组算法不仅提高共识达成概率,还发现在同样的节点数,不同的分组方式会影响达成共识的概率。根据拜占庭容错,$N\geqslant 3f+1$,其中 N 是节点总数,f 是拜占庭节点数。如果有 28 个节点分为 4 组,一组为 7 个节点,这种分组方式若有 9 个拜占庭节点,达成共识的概率为:

$$\left(1-\frac{c_9^3\ c_6^3\ c_{22}^4\ c_{18}^4\ c_{14}^7}{c_{28}^7\ c_{21}^7\ c_{14}^7}\right)\approx 73\%\ 。\tag{2}$$

若分为 7 组每组 4 个节点，这种分组方式若为 9 个拜占庭节点共识达成概率为

$$\left(1-\frac{c_9^2\ c_7^2\ c_5^2\ c_{22}^2\ c_{20}^2\ c_{18}^2\ c_{16}^4\ c_{12}^4\ c_8^4}{c_{28}^4\ c_{24}^4\ c_{20}^4\ c_{16}^4\ c_{12}^4\ c_8^4}\right)\approx 95\%\ 。\quad (3)$$

同样节点分组变多会提升共识的达成概率。

1.2 随机分组过程

如果分组按照一定的规则，例如选取分组，分组结果就是可预测的，拜占庭节点非常容易串通破坏共识，例如，有 16 个节点，分成 4 组，每组有 4 个节点，根据拜占庭共识结论，$N>=3f+1$，所以总体最多可以允许 5 个拜占庭节点，每个分组只能允许 1 个拜占庭节点，允许一个分组为拜占庭分组，假如这 5 个节点串通，分成 2 和 3 分别分入两个分组之中，就会导致两个拜占庭分组出现从而导致整体无法达成共识。为此本文提出一种结合可验证随机函数 VRF[31] 的随机分组过程，分为随机分组信息生成和随机分组信息的确认。

哈希函数其值域是离散的、均匀分布的，给定不同的输入值，其输出值没有规律，随机的洒落、分布在值域区间内，因此，哈希函数的输出值是无法预知的，从而保证分组信息的随机性。VRF 是一种结合非对称密钥技术的哈希函数，具体流程：

①证明者计算 $proof=VRF\_PROVE(sk,info)$，sk 是私钥。

②证明者把 $proof$ 和 $info$ 公布出去。

③验证者计算 $True/False=VRF\_Verify(proof,info,pk)$，pk 是公钥。

之所以采用上述流程，因为可以保证随机分组信息的可验证性。本文在 VRF 可验证随机函数基础上设计了一种抽签方式随机分组信息生成。具体原理：$proof$ 是哈希函数的输出值，值长度一般为 256，如果把输出值作为一个大整数 $proof\in[0,-1]$，可得 $\frac{proof}{2^{256}-1}\in[0,1]$，设置抽签阈值 $t\in[0,1]$，当 $\frac{proof}{2^{256}-1}>=t$ 时，抽签成功可以公布 $proof$ 和 $info$，假设有 16 个节点，可以分为 4 组，分组方式有 $c_{16}^4c_{12}^4c_8^4$ 种，分组方式对应到区间 $[t,1]$，第 n 种分组方式对应的区间为 $[t+\frac{n(1-t)}{c_{16}^4c_{12}^4c_8^4},t+\frac{n+1(1-t)}{c_{16}^4c_{12}^4c_8^4}]$，$(n\in[0,c_{16}^4c_{12}^4c_8^4-1])$，当验证者接收到 $proof$ 和 $info$ 通过了 VRF_Verify 验证确定 $\frac{proof}{2^{256}-1}$ 的值便可得出分组信息。上述分组方式与区间对应是均匀的，但也可以设置权重扩大某类分组方式的对应区间，因为由上小节概率分组算法的规律可得，分组越少共识达成概率越高。

随机分组信息的确认采用二次确认过程，这样保证了部分同步网络模型下节点之间的一致性。

步骤如图 2 所示：

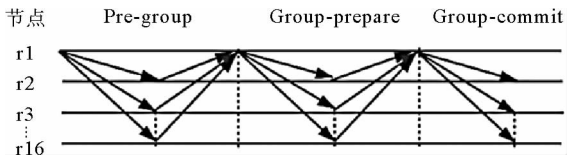

图 2　随机分组信息确认流程

Fig.2　Random group information confirmation process

第一步：主节点根据自己的私钥 sk 签名接收到 client 的信息作为 data-sk，当前轮数作为 n，分组信息作为 VRF_PROVE 的 info，不断随机 info 计算出自己这一轮的 proof。如果满足抽签阈值，就广播 $<pre\text{-}group,n,info,proof,data\text{-}sk>$ 到其他节点验证。

第二步：副本节点收到验证信息，查看轮数 n 是否比之前的大，然后验证 $VRF\_Verify(proof,info,pk)$，如果验证通过就用私钥签名确认信息 $<pre\text{-}group\text{-}ack,nodeID,data\text{-}sk>$ 并回复。

第三步：主节点接收到回复，如果收到 2f+1 的节点签名确认就组合成 data-sk-list，然后广播 $<group\text{-}prepare,n,info,proof,data\text{-}sk\text{-}list>$ 到其他节点，其他副本节点验证消息是否得到了 $2f+1$ 个节点签名并且验证 $VRF\_Verify(proof,info,pk)$ 是否通过，若通过再次用私钥签名确认回复。

第四步：主节点收集到 $2f+1$ 个节点签名确认信息，发送信息到其他副本节点，副本节点通过最终的 proof 确认对应的分组信息。上述四步消息格式如表 2。

表 2　发送接受对象之间的消息格式

Table 2　Message format between sending and receiving objects

发送对象 Sender	接收对象 Recipient	信息 Message
leader	backup	$<pre\text{-}group,n,info,proof,data\text{-}sk>$
backup	leader	$<pre\text{-}group\text{-}ack,nodeID,data\text{-}sk>$
leader	backup	$<group\text{-}prepare,n,info,proof,data\text{-}sk\text{-}list>$
backup	leader	$<group\text{-}prepare\text{-}ack,nodeID,data\text{-}sk>$
leader	backup	$<group\text{-}commit,info,proof,data\text{-}sk\text{-}list>$

1.3 mix-raft-pbft 混合共识算法

如图3所示，mix-raft-pbft共识流程分为组内共识和组间共识，其中组内共识是拜占庭容错chain-raft算法分为两个阶段保证部分同步网络模型下数据的一致性，其中包含组内预确认阶段和组内确认阶段。组内共识阶段：①组内主节点用私钥签名接收到的client交易数据data，生成交易数据哈希值data-hash保证数据的完整性，为这笔交易数据标记唯一data-ID，组成＜*pre-commit*，*data-ID*，*data-sk*，*data-hash*＞消息发送到副本节点；②副本节点接收到消息，用主节点公钥pk解密消息，首先验证交易数据data-ID是否已经存在，若已存在拒绝接收，若不存在继续验证交易数据data是否与data-hash一致，若一致用私钥签名确认回复＜*pre-commit-ack*，*data-ID*，*data-sk*，*data-hash*，*node-ID*＞消息；③主节点接收到其他副本节点消息确认当超过$2f+1$个确认主节点组合签名data-sk-list，然后发送其他副本节点＜*commit*，*data-ID*，*data-sk*，*data-hash*，*data-sk-list*＞消息；④副本节点接收到消息，确认data-ID是否存在pre-commit消息，验证主节点签名，验证data-sk-list签名集合是否超过$2f+1$，验证通过回复＜*commit-ack*，*data-ID*，*data-sk*，*data-hash*，*node-ID*＞消息到主节点。组间共识阶段组间节点之间完成三阶段PBFT共识。

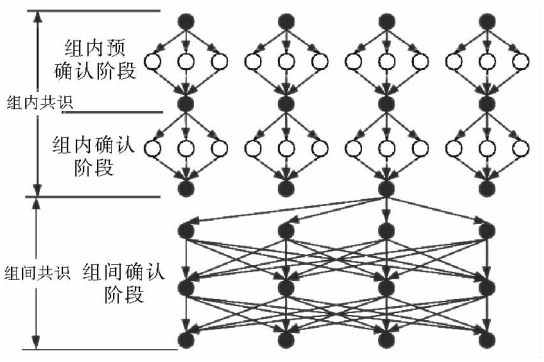

图3 mix-raft-pbft共识流程

Fig.3 mix-raft-pbftconsensus process

以上描述的拜占庭容错chain-raft算法共识流程是一笔交易先进行pre-commit阶段，主节点收集签名集合data-sk-list，然后进入commit阶段主节点再次收集签名集合data-sk-list，这个流程交易是一笔交易走完两个阶段然后下一笔交易再次走完这两个阶段，其实当主节点正在进行第一笔交易commit阶段的时候，可以同时发送第二笔交易的pre-commit消息，这样主节点就可以同时收到副本节点第一笔commit-ack回复和第二笔交易的pre-commit-ack回复，优化了共识流程，如图4所示。

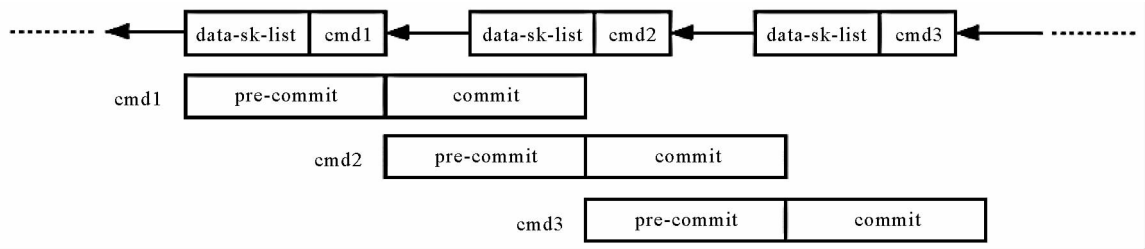

图4 chain-raft流程优化图

Fig.4 chain-raft process optimization diagram

1.4 共识算法分析

1.4.1 通信分析 在分布式网络中假设由r个节点，若在不分组的情况下直接采用PBFT共识，通信复杂度为$(r-1)+2r(r-1)$，若使用本文共识机制，假设分为g组每组节点个数为m，组内通信复杂度为$max_{i=1}^{i=g}\{2(m_i-1)\}$，组间通信复杂度为$(g-1)+2g(g-1)$，总的通信复杂度为$2g^2-g-1+max_{i=1}^{i=g}\{2(m_i-1)\}$。由于组内通信复杂度为$o(n)$，而组间通信复杂度为$o(n^2)$，所以尽可能减少分组会降低整体通信复杂度。但是当组间通信复杂度小于组内通信复杂度时，继续分组通信复杂度反而会上升。因此最优解条件是$2g^2-g-1=max_{i=1}^{i=g}\{2(m_i-1)\}$。

1.4.2 安全性分析 随机分组防止了拜占庭节点之间的串通，分组需要$2f+1$个节点确认，保证在最多f个拜占庭节点的条件下，分组可以正确完成。

组内chain-raft共识不存在同一个交易ID中不同的交易都能收到足够多投票的情况，保证最终存储的一致性。

证明：假设$N=3f+1$为节点总数，f为拜占庭节点最大数量，那么当收到$2f+1$投票为足够多投票。因两个区块都收到至少$2f+1$，投票总量至少为$2(2f+1)=N+f+1$，能看到至少有$f+1$对同一个交易ID中不同的交易都投了票，与f个拜占庭节点假设矛盾。

1.4.3 活性分析 活性指的是所有好的事情一定发生，就是要在一定时间范围内完成整个共识过程，并保证所有诚实节点提交一致性结果。共识算法活性出现

的原因主要是节点之间可能存在的传输延迟，传输延迟会导致节点无法产生一致性结果。

随机分组过程活性主节点必须在一定时间内收集到 2/3 个节点签名，如果在一定时间内没有收集到足够的签名，其他节点会代替主节点，在 $N>=3f+1$ 的条件下，一定时间内是可以保障主节点活性。

组内拜占庭 chain-raft 共识算法中，活性主要体现在随机超时时间 t 的设立。$t\in[T,2T]$ 并且满足 $T>=randomtime$ 的条件，以避免因为消息延迟而导致的不必要的主节点切换，同时 t 的设立又可以让系统在有限的时间内产生一个主节点，保证系统的正常运行。

2　仿真实验

本文通过 go 语言实现 PBFT 和分组混合共识机制，使用线程监听不同的端口代替节点，线程之间的交互使用 TCP 协议，模拟分布式系统共识过程。环境中包含 RSA 信息加解密过程和网络信息传递延迟，如表 3 所示。

表 3　实验环境

Table 3　Experimental environment

配置 Configuration	详细信息 Details
系统信息 System Information	Ubuntu16.04
CPU	Intel i7 2.8GHz 双核
内存 RAM	4GB 2400MHz DDR4
硬盘 Hard Disk	60GB HDD
Golang 版本 Golang Version	Golang1.4

2.1 模拟网络信息通信延迟

本文混合共识适用于联盟链，联盟链使用者大部分为企业，因此模拟企业最普遍的网络拓扑结构星型结构，依照此结构计算网络信息传递延迟。

图 5 为 n 个节点网络拓扑结构，任意两个节点 n_i，n_j 的延迟，若 $(n_i,n_j)\subset$同一个 hub，延迟为 30～50 ms，若 $(n_i,n_j)\not\subset$同一个 hub，若相隔 hub 个数为 k，延迟为 30～50 kms。节点延迟是一个包含几个值的集合，这些值用于模拟节点路由到不同节点之间的延迟。

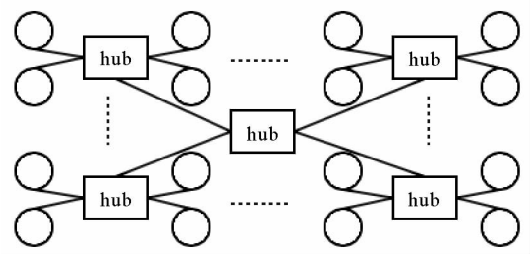

图 5　星型网络结构

Fig.5　Star network structure

2.2 共识实验结果

实验从共识时延和吞吐量两个方面对比 PBFT 与分组混合共识机制，结果如下

图 6 是在本文算法在 100 笔交易，随着节点数量增加，共识时间对比。chain-raft 共识随着节点数增加，共识时间接近线性增长趋势，而 pbft 共识接近多项式级平方增长。本文混合共识机制组内使用拜占庭容错 chain-raft 共识，降低了整体共识的消息复杂度和共识时间，提高了系统性能。

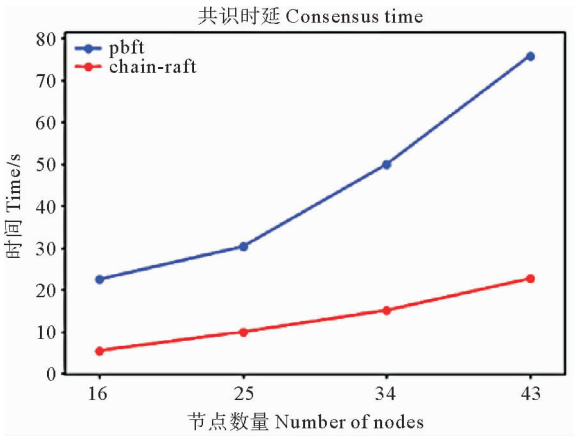

图 6　共识时延对比

Fig.6　Consensus delay comparison

图 7 是本文共识在 100 笔交易分为四组的情况下与 PBFT 共识在不同节点下的对比，可以看出 PBFT 随着节点数量增加，共识时间迅速增加，而本文共识由于分组，在分为 4 组的情况下，组间共识时间基本不变，而组内共识时延增长为线性、组内节点数量增加缓慢并且分组之间共识并行化，最终整体达成共识时间增加缓慢。随着节点的增加与 PBFT 共识时间的优势会越来越大。

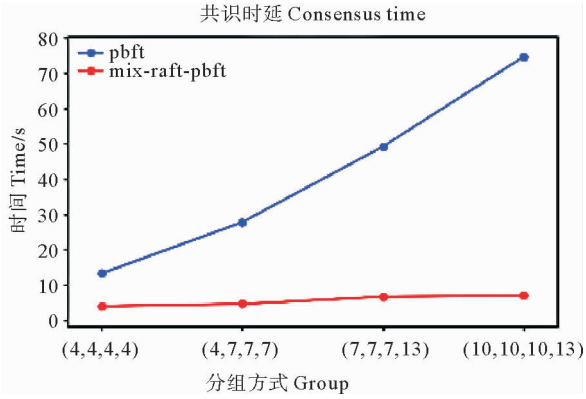

图 7　共识时延对比

Fig.7　Consensus delay comparison

图 8 是本文共识算法在 10 笔交易，100 个节点分为不同组数情况下的共识时间。因为组间是 PBFT 共识，通信复杂度 $o(n^2)$，所以随着分组变少，共识时间变少，但随着组内节点变多，对整体共识时间影响变大，因此最终分组的确定需要与达成共识概率做一个平衡。

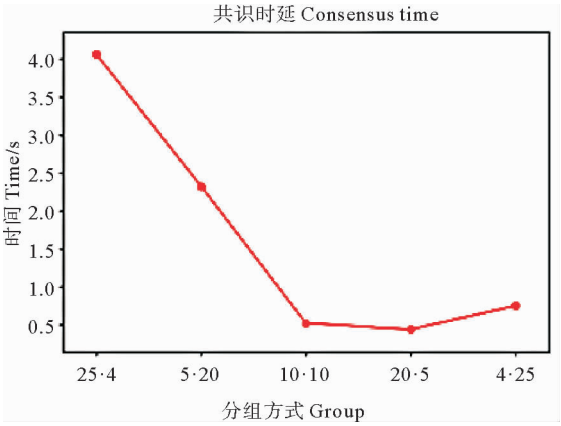

图 8 mix-raft-pbft 不同分组共识时延

Fig.8 Different group consensus delay

图 9 是在不同的节点数量分为四组的情况下，mix-raft-pbft 与 PBFT 吞吐量的对比，mix-raft-pbft 由于组内节点数量增加导致整体吞吐量下降，不过整体吞吐量容然有较大提升。

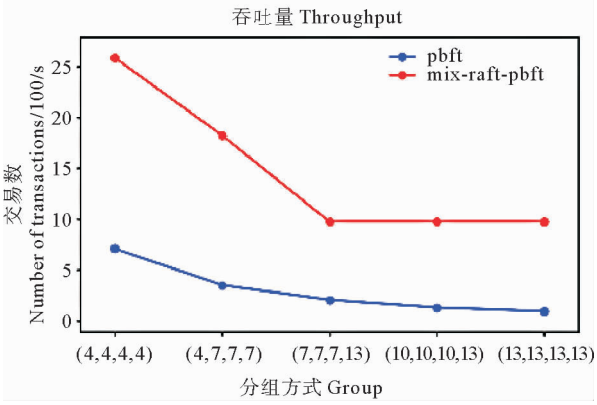

图 9 mix-raft-pbft 与 PBFT 吞吐量对比

Fig.9 The throughput comparison between mix-raft-pbft and PBFT

3 结语

在本文中，简要介绍了 PBFT 算法。针对 PBFT 的共识效率，提出了一种基于 PBFT 的分组混合共识机制来提升效率并且保证了安全性不变。分组混合共识机制包括三个阶段，第一阶段概率分组算法，提升了共识达成概率；第二阶段随机分组，防止拜占庭节点的串通；第三个阶段 mix-raft-pbft 混合共识算法，组内使用拜占庭容错 chain-raft 算法，组间使用 PBFT 算法，因为已经完成分组，所以分组之间可以同时进行共识，最终分组之间完成 PBFT 算法共识，组内拜占庭容错 chain-raft 算法降低了通信复杂度，分组减少了最终 PBFT 算法共识的参与节点同时减小了主节点通信压力，从而大大缩短了整体共识时间。

参考文献：

[1] Sateshi N. Bitcoin: peer-to-peer electronic cash system[J]. Consulted, 2008(1): 28.

[2] King S, Nadal S. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake[J]. self-published paper, August, 2012, 19: 1.

[3] Buterin V, Griffith V. Casper the friendly finality gadget[J]. arXiv preprint arXiv, 1710.09437.

[4] Bentov I, Lee C, Mizrahi A, et al. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y[J]. Evaluation Review, 2014, 42(3): 34-37.

[5] Wang W, Hoang D T, Hu P, et al. A survey on consensus mechanisrhs and mining strategy management in blockchain networks [J]. IEEE Access, 2019, 7: 22328-22370.

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

[7] Ongaro D, Ousterhout J. In Search of an Understandable Consensus Algorithm[C]. [s.l.]: 2014 {USENIX} Annual Technical Conference ({USENIX}{ATC} 14). 2014: 305-319.

[8] Castro M, Liskov B. Practical Byzantine fault tolerance[J]. OSDI, 1999, 99: 173-186.

[9] Van Renesse R, Schneider F B. Chain Replication for Supporting High Throughput and Availability[J]. OSDI, 2004, (4): 91-104.

[10] Vvkolić M. Rethinking Permissioned blockchanis[C]. Proceedings of the ACM Workshop on Blackchain, Cryptocurrencies and Contracts, 2017: 3-7.

[11] Bessani A, Sousa J, Alchieri E E P. State machine replication for the masses with BFT-SMART [C]. IEEE: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2014: 355-362.

[12] Abd-El-Malek M, Ganger G R, Goodson G R, et al. Fault-scalable Byzantine fault-tolerant services[J]. ACM SIGOPS Operating Systems Review, 2005, 39(5): 59-74.

[13] Cowling J, Myers D, Liskov B, et al. HQ replication: A hybrid quorum protocol for Byzantine fault tolerance[C]. [s.l.]: Proceedings of the 7th symposium on Operating systems design and implementation, 2006: 177-190.

[14] Aublin P L, Guerraoui R, Knežević N, et al. The next 700 BFT protocols[J]. ACM Transactions on Computer Systems (TOCS), 2015, 32(4): 1-45.

[15] Kotla R, Alvisi L, Dahlin M, et al. Zyzzyva: Speculative byzantine fault tolerance[C]. [s.l.]: Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, 2007: 45-58.

[16] Singh A, Fonseca P, Kuznetsov P, et al. Zeno: Eventually Con-

sistent Byzantine-Fault Tolerance[C]. NSDI，2009，9：169-184.

[17] Wood T，Singh R，Venkataramani A，et al. ZZ and the art of practical BFT execution[C]. [s.l.]：Proceedings of the sixth conference on Computer systems，2011：123-138.

[18] Shoker A，Bahsoun J P，Yabandeh M. Improving independence of failures in bft[C]. IEEE：2013 IEEE 12th International Symposium on Network Computing and Applications，2013：227-234.

[19] Kapitza R，Behl J，Cachin C，et al. CheapBFT：Resource-efficient byzantine fault tolerance[C]. [s.l.]：Proceedings of the 7th ACM european conference on Computer Systems，2012：295-308.

[20] Hendricks J，Sinnamohideen S，Ganger G R，et al. Zzyzx：Scalable fault tolerance through Byzantine locking[C]. IEEE：2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)，2010：363-372.

[21] Fan J，Yi L T，Shu J W. Research on the technologies of Byzantine system[J]. Journal of Software，2013，24(6)：1346-1360.

[22] Clement A，Wong E L，Alvisi L，et al. Making byzantine fault tolerant systems tolerate byzantine faults[J]. NSDI，2009，9：153-168.

[23] Amir Y，Coan B，Kirsch J，et al. Prime：Byzantine replication under attack[J]. IEEE Transactions on Dependable and Secure Computing，2010，8(4)：564-577.

[24] Veronese G S，Correia M，Bessani A N，et al. Spin one's wheels? Byzantine fault tolerance with a spinning primary[C]. IEEE：2009 28th IEEE International Symposium on Reliable Distributed Systems，2009：135-144.

[25] Aublin P L，Mokhtar S B，Quéma V. Rbft：Redundant byzantine fault tolerance[C]. IEEE：2013 IEEE 33rd International Conference on Distributed Computing Systems，2013：297-306.

[26] Tschorsch F，Schevemann B. Bitcon and beyond：Atechnical survey on decentralized digital currencies[J]. IEEE Communications Surveys & Tutorials，2016，18(3)：2084-2123.

[27] Aggarwal S，Challdhary R，Avjla G S，et al. Blockchain for smart communities：Applications，challenges and opportunities[J]. Journal of Network and Computer Applications，2019，144：13-48.

[28] Bvchpom E. Tendermint：Byzantime fance tolerance in the age of blockchains (2016)[J]. Query date，2018(2)：2-26.

[29] Innerbichler J，Darnjanovic-Behrendt V. Federated byzantine agreement to ensure trustworthiness of digital mannfacturing platforms[C]. [s.l.]：Proceedings of the Ist Workshop on Crypeocurrencies and Blockchains for Distributed Systems，2018：111-116.

[30] Team Z. The zilliqa technical whitepaper[J]. Retrieved September，2017，16：2019.

[31] Micali S，Rabin M，Vadhan S. Verifiable random functions[C]. IEEE：40th annual symposium on foundations of computer science (cat. No. 99CB37039)，1999：120-130.

[32] Miller A，Xia Y，Croman K，et al. The honey badger of BFT protocols[C]. [s.l.]：Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security，2016：31-42.

[33] Yin M，Malkhi D，Reiter M K，et al. Hotstuff：Bft consensus in the lens of blockchain[J]. arXiv preprint arXiv 1803.05069.

A Byzantine Fault Tolerant Hybrid Consensus Mechanism Applicable to Consortium Blockchain

Zhou Wei[1,2], Yuan Xiaowei[1], Wei Zhiqiang[3], Zhai Yili[2], Wang Chao[1], Du Bingyu[1], Zhu Wenyin[2], Wang Jinlong[1]

(1.School of Information and Control Engineering, Qingdao University of Technology, Qingdao 266100, China; 2. Haier Group Technology R&D Center, Qingdao 266100, China; 3.College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China)

Abstract: Byzantine Fault Tolerance (BFT) will become more and more important in providing the reliability of distributed systems. Among them, practical Byzantine Fault Tolerance (PBFT) is one of the best algorithms currently used, but it faces distributed systems with many nodes, and PBFT is not satisfactory. Since the PBFT consensus failure rate is close to 1/3, the high consensus failure rate will cause frequent master node switching, slowing down the consensus efficiency, and the probability grouping algorithm reduces the consensus failure rate, thereby improving the consensus efficiency. In order to prevent Byzantine nodes from colluding and attacking groups, the VRF lottery method based on the verifiable random function makes the Byzantine nodes unable to predict the grouping and further improves the security of consensus. Byzantine fault-tolerant chain-raft is used in the hybrid consensus mechanism group to accelerate consensus efficiency, and PBFT is used between groups to ensure compatibility with existing PBFT consensus. In the consensus mechanism of this paper, when 16 nodes are divided into four groups, the message complexity is 92(lower than PBFT, and the throughput is 3.6 times that of PBFT).

Key words: Practical Byzantine Fault Tolerance(PBFT); consortium blockchain; grouped hybrid consensus; grouping based on probability; random grouping

责任编辑 徐 环