基于哈希值分组和信任主节点选取的共识机制

韩昊澎[1,2],金　瑜[1,2]
(1. 武汉科技大学 计算机科学与技术学院,湖北 武汉 430065;
2. 湖北省智能信息处理与实时工业系统重点实验室,湖北 武汉 430065)

摘　要:PBFT(Practical Byzantine Fault Tolerance)算法是当前最流行的共识机制之一,但其采用广播的通信模式导致该算法具有高通信复杂度;将节点分层进行共识的改进方式虽降低了通信规模,但改进后的算法在安全性和效率等方面仍存在不足。基于此,提出一种 HBFT 改进算法。首先,根据节点 MAC 地址的哈希值进行分组,增加节点分组的随机性并使拜占庭节点的分布更均匀;其次,优化算法的共识流程、通信内容和视图更换协议,进一步降低通信规模并提高主节点更换和故障处理的速度;最后,引入信誉机制并据此选取主节点,提升主节点的可靠性和算法的安全性。从理论、实验和安全的角度进行分析验证,结果表明 HBFT 算法的共识效率较 PBFT 算法和基于分层的改进算法分别提高 96.1% ~98.6%,51.3% ~89.7%,且 HBFT 算法具有更高的安全性。

关键词:区块链;共识机制;哈希函数;信誉模型;共识效率
中图分类号:TP339　　**文献标识码**:A　　**文章编号**:1673-629X(2024)05-0016-08
doi:10.20165/j.cnki.ISSN1673-629X.2024.0035

Consensus Mechanism Based on Hash Grouping and Trust Primary Node Selection

HAN Hao-peng[1,2],JIN Yu[1,2]
(1. School of Computer Science and Technology,Wuhan University of Science and Technology,
Wuhan 430065,China;
2. Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System,
Wuhan 430065,China)

Abstract:Practical Byzantine Fault Tolerance algorithm (PBFT) is one of the most popular consensus mechanisms. However,the communication mode of its broadcast leads to high communication complexity. Although the improved method of hierarchical node consensus reduces the communication scale of the algorithm,there are still shortcomings in security and efficiency of the algorithm. Based on this,an improved HBFT algorithm is proposed. Firstly,the nodes are grouped according to the hash value of MAC address,which increases the randomness of node grouping and makes the distribution of Byzantine nodes more uniform. Secondly,the consensus process,communication content and view replacement protocol of the algorithm are optimized to further reduce the communication scale and improve the speed of master node replacement and fault handling. Finally,the credit mechanism is introduced and the master node is selected based on it. It improves the reliability of the master node and the security of the algorithm. From the perspective of theory,experiment and safety,it is showed that the consensus efficiency of HBFT algorithm is improved by 96.1% ~98.6% and 51.3% ~89.7% compared with PBFT algorithm and layer-based improved algorithm. In addition,HBFT algorithm has higher security.

Key words:blockchain;consensus mechanism;hash function;reputation model;consensus efficiency

0　引　言

区块链的诞生可追溯到 2008 年中本聪发表的《比特币:一种点对点的电子现金系统》[1]一文。因其具有去中心化、防篡改和可追溯等特征[2],被认为是一种新兴技术[3],在金融[4]、教育[5]、物联网[6]以及医疗数据共享[7]等领域有着广阔的应用前景。区块链技术的核心是共识机制[8]。Castro 和 Liskov 在 1999 年提出的 PBFT[9](Practical Byzantine Fault Tolerance)算法,

收稿日期:2023-09-11　　**修回日期**:2024-01-11
基金项目:国家自然科学基金项目(61802286)
作者简介:韩昊澎(1999-),男,硕士研究生,CCF 会员(P6611G),研究方向为区块链、共识机制;通信作者:金　瑜(1973-),女,博士,教授,CCF 会员(14140M),研究方向为网络安全、信任模型、分布式计算等。

因其可有效解决拜占庭将军问题[10]广泛应用于联盟链中。伴随联盟链中网络节点规模的迅猛增长,PBFT 算法的共识效率受到严重限制。目前,很多 PBFT 改进算法通过分层等措施降低算法的通信规模,进而提升算法的性能,如 SPBFT[11]、hBFT[12]、kPBFT[13]、lPBFT[14]、tPBFT[15]、DLBFT[16]等,但它们整体上沿用 PBFT 算法的通信方式和视图更换机制,无法从根本上突破性能瓶颈,主要表现在以下三点:(1)沿用 PBFT 算法的广播通信模式,导致通信量大;(2)仅由系统视图序号决定下任主节点导致安全性差;(3)拜占庭节点担任系统主节点将开启复杂的视图更换流程导致共识效率低。

基于此,提出一种基于哈希值分组和信任主节点选取的共识机制,作如下改进:(1)根据节点 MAC 地址的哈希值对节点进行排序和分组,增加节点分组的随机性并使拜占庭节点的分布更加均匀;(2)依据节点信誉值选取高可靠性节点担任主节点,提升算法安全性;(3)优化算法的共识流程和通信内容,减少共识阶段节点间的通信量;(4)改进视图更换协议,减少主节点的更换时间和共识失败的处理时间,提升共识效率。

1　相关工作

区块链分为公有链、联盟链和私有链[17]。工作量证明(Proof of Work, PoW)、权益证明[18](Proof of Stake, PoS)及授权股份证明[19](Delegated Proof of Stake, DPoS)是公有链主要的共识机制。私有链通常使用 Raft[20]协议。联盟链主要使用 PBFT 共识机制,但 PBFT 算法广播的通信方式导致通信量极大,为提高该算法的性能很多改进算法被提出。

2018 年,Feng 等人[11]基于分层提出一种动态多代理 PBFT 算法,将节点分为多个自治系统虽提升了算法的可拓展性和灵活性,但附加的消息传递次数和签名加密过程加剧了计算开销。2019 年,Thai 等人[12]由管理组代替主节点,有效降低了算法的时间复杂度。但将一个私钥分为多份加深了密钥管理复杂性,并降低了私钥的完整性和安全性。Chen 等人[13]基于 K-medoids 聚类算法将节点按照特征相似性进行分组,虽提高了算法的共识效率,但该算法计算复杂度高且需人为预设初始簇心。2020 年,Li 等人[14]精简了 PBFT 算法的共识流程,虽降低了节点间共识的通信量,但节点对消息验证次数的增加提高了等待时延。2021 年,Qushtom 等人[15]提出一种基于分层和请求批处理的改进算法,由上层节点对底层节点的审核结果进行批准,虽改善了算法的安全性和请求处理速度,但该算法通信复杂度仍很大。对此,Li 等人[16]提出一种双层改进算法 DLBFT,将节点分层分组使改进后算法的通信复杂度接近线性,改进后的算法性能有效提升,但该算法缺乏信誉机制,选举出的主节点可靠性较低。

上述改进机制与 PBFT 算法的对比如表 1 所示。

表 1　共识机制特征对比

共识机制	共识及视图更改协议	主要改进措施	主节点选取依据	分组依据
PBFT	3 阶段共识	无	轮流担任	无
SPBFT	PBFT	引入代理	轮流担任	节点序号
hBFT	PBFT	引入管理组	投票	节点序号
kPBFT	PBFT	聚类分组	聚类簇心	节点相似性
lPBFT	2 阶段 PBFT	减少共识阶段	轮流担任	节点序号
tPBFT	PBFT	请求批处理	轮流担任	节点序号
DLBFT	双层 PBFT	分层共识	轮流担任	节点序号

2　HBFT 算法

2.1　算法预设

2.1.1　相关定义

定义 1:节点。

节点为区块链共识的参与者,拥有 credit(信誉值)、MAC(MAC 地址)、index(序号)、v(投票得分)和 Hash(哈希值)等属性。系统中节点总数为 N,分组数为 x,各组节点数为 y,节点信息用 < node > 表示,节点分组信息用 $G=\{g_1,g_2,\cdots,g_x\}$ 表示。

定义 2:主节点。

各组主节点用 g_i. pri_node 表示;主共识层表示为 < PRI_NODE > ;全局主节点用 PRI 表示。

定义 3:REQUEST 消息。

$<\text{REQUEST},o,t,c>_{\sigma_c}$,其中 o 为请求的操作,t 为时间戳,c 为客户端的身份标识,σ_c 为客户端对消息的签名。

定义 4:PREPARE 消息。

$<<\text{PREPARE1},n,d,s,i>m>_{\sigma_i}$,其中 m 为请求的内容,n,d 分别为请求的序号和摘要,i 为节点序号,$s=(n,d)_{\sigma_i}$。$<<\text{PREPARE2},D,n,i>m>_{\sigma_i}$,其中 D 包含节点接收到的所有 PREPARE1 消息中的 s。

定义 5:REPLY 消息。

$< \text{REPLY}, r, t, i, c >_{\sigma_i}$, r 为请求执行的结果。

定义 6:SUCCESS 消息。

$< \text{SUCCESS}, r, c, d, t, \text{CO} >_{\sigma_c}$,其中 r 与数量超过主共识层节点数 2/3 的一致 COMMIT 消息中的 r 一致,CO 为客户端接收到的所有 COMMIT 的消息。

定义 7:COMMIT 消息。

$< \text{COMMIT}, r, t, i, c >_{\sigma_i}$,其中 r 与该组主节点接收到的由本组节点发来且数量超过本组节点数 1/2 的一致 REPLY 消息中的 r 一致。

定义 8:REFRESH 消息。

$< \text{REFRESH}, \text{Credit}, i >_{\sigma_i}$,其中 Credit 为集合 $\{\text{node}_1 . \text{credit}, \cdots, \text{node}_n . \text{credit}\}$,指系统中各节点更新后的信誉值。

定义 9:VIEW_CHANGE 消息。

$< \text{VIEW\_CHANGE}, m, i, \text{num} >_{\sigma_i}$,其中 num 是当前故障主节点的序号。

定义 10:NEW_VIEW 消息。

临时组主节点向主共识层广播的 NEW_VIEW 消息格式为 $< \text{NEW\_VIEW}, C, m, i, \text{num} >_{\sigma_i}$;向本组节点广播的消息格式为 $< \text{NEW\_VIEW}, m, i, P, C >_{\sigma_i}$,其中 P 为生成的新 PREPARE2 消息,C 为接收到的 VIEW_CHANGE 消息集合。

定义 11:VOTE 消息。

$< \text{VOTE}, \text{vote}, i >_{\sigma_i}$,在投票阶段生成且在本组内进行广播,其中 vote 代表节点 i 向其它节点投票的分值,表示为 $\{\text{vote}_{i1}, \cdots, \text{vote}_{iy}\}$。

定义 12:节点综合得分。

根据信誉模型为每个节点计算产生的分值,符号表示为 S_j。相关符号 T_i 为节点 i 的信誉值,Rand 为可设置的整数值,α 和 β 分别为节点获得的投票总分和自身信誉值对综合得分的影响权重。

2.1.2 HBFT 算法的改进分析

HBFT 算法从节点分组依据、主节点选取、共识流程等方面进行改进,其改进策略与 PBFT 算法的对比如表 2 所示。

表 2 HBFT 算法的改进分析

共识机制	共识及视图更改协议	信誉模型	主节点选取依据	分组依据	主节点更换周期
PBFT	3 阶段共识	无	轮流担任	无	1 轮共识
HBFT	6 阶段共识	有	信誉模型组内投票	节点 MAC 地址哈希值	多轮共识

2.2 系统概述

HBFT 算法由四部分组成:哈希值分组策略、信誉机制、共识流程和故障处理。首先,通过各节点的 MAC 地址计算哈希值,根据哈希值对节点进行排列并分组。其次,各节点使用投票算法对本组节点投票,并根据节点信誉值及获得的投票值计算综合得分,选出各组主节点和全局主节点。最后,共识达成则在下轮共识过程中根据节点的历史行为按照信誉模型更新节点信誉值;共识未达成则开启故障处理,更换拜占庭主节点并对未达成共识的请求进行处理。HBFT 算法框架如图 1 所示。

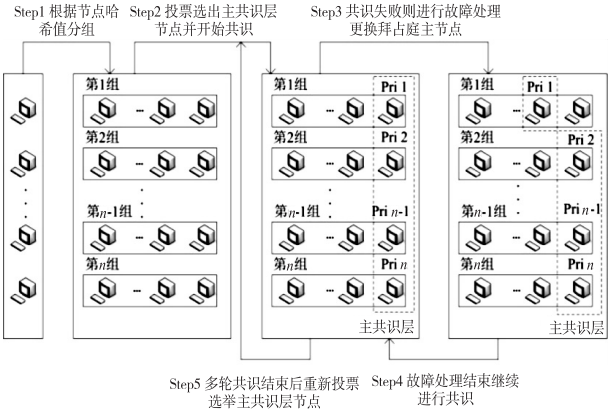

图 1 HBFT 算法框架

2.3 哈希值分组策略

提出一种依据哈希值进行节点分组的策略。由于网络设备的 MAC 地址具有唯一性,且哈希函数计算的结果具有抗冲突的性质,利用节点 MAC 地址的哈希值进行排序保证了节点哈希值的唯一性,又可有效打乱节点的顺序。这提高了节点分组的随机性,使拜占庭节点在各组均匀分布,降低了拜占庭节点对共识结果的干扰。具体分组流程如下:首先,对各节点的 MAC 地址进行散列获得各节点的唯一哈希值。其次,将节点按照哈希值由大到小的顺序进行排列并分组。当系统中节点总数和组数分别为 N 和 x,则每组节点数为 $y = N/x$,存在余数则放在最后一组。哈希值分组的流程如算法 1 所示。

算法 1:哈希值分组算法

输入:<node>, N, x

输出:G

(1) Initialize $G \leftarrow \{g_1, g_2, \cdots, g_x\}$

(2) $y \leftarrow N/x$

(3) for i from 1 to N

(4) $\text{node}_i . \text{Hash} \leftarrow \text{hash}(\text{node}_i . \text{MAC})$

(5) for i from 1 to N

(6) for j from 1 to N

(7) if $\text{node}_i . \text{Hash} < \text{node}_j . \text{Hash}$

(8)　$node_i \leftrightarrow node_j$
(9) $k \leftarrow 1$
(10) for i from 1 to N
(11)　　if $i \bmod y \neq 0$ or $k = x$
(12)　　g_k. add($node_i$)
(13)　　else
(14)　　g_k. add($node_i$)
(15)　　$k \leftarrow k + 1$
(16) return G

2.4　信誉机制

信誉机制包括信誉模型、投票算法和主节点选取三部分。为减少频繁更换主节点产生的时延,设置每共识 Rand 轮后再开启一次投票进行主节点的更换。

2.4.1　信誉模型

令节点初始信誉值为 50。共识成功时,子层节点 REPLY 消息中的 r 与 SUCCESS 消息中的 r 一致则信誉值增加 10,否则扣除 15;主共识层节点 COMMIT 消息中的 r 与 SUCCESS 消息中的 r 一致则信誉值增加 15,否则扣除 20、判定该节点为故障节点并进行故障处理。共识失败时,根据某组临时主节点广播的 NEW_VIEW 消息,于共识结束后扣除该组故障主节点 20 信誉值。节点信誉值的变化储存在 REFRESH 消息中。一轮共识结束后,节点信誉值 T_i 的变化可表示如下:

$node_i \notin <$ PRI_NODE $>$:

$$T_i = \begin{cases} T_i + 10 & node_i.\ \text{REPLY}.\ r = \text{SUCCESS}.\ r \\ T_i - 15 & node_i.\ \text{REPLY}.\ r \neq \text{SUCCESS}.\ r \end{cases}$$

$node_i \in <$ PRI_NODE $>$:

$$T_i = \begin{cases} T_i + 15 & node_i.\ \text{COMMIT}.\ r = \text{SUCCESS}.\ r \\ T_i - 20 & node_i \text{行为异常} \end{cases}$$

(1)

2.4.2　投票算法

节点参考被投票节点的信誉值,采用广播的方式对组内节点进行投票;接收到投票消息的节点根据消息内容分别计算组内各节点获得的投票总分,广播的内容为 VOTE 消息。投票算法令组主节点的产生既依赖于节点信誉值,又可满足组内大多数节点的意愿,使选举过程更加公平、民主。节点投票的分值如下:

$$vote_{ij} = \begin{cases} 0 & node_j.\ \text{credit} \leqslant node_i.\ \text{credit} \\ 1 & node_j.\ \text{credit} \in (node_i.\ \text{credit}, 2 * node_i.\ \text{credit}] \\ 2 & node_j.\ \text{credit} > 2 * node_i.\ \text{credit} \end{cases}$$

(2)

2.4.3　主节点选取

投票结束后,所有节点根据公式 3 计算组内各节点的综合得分 S_j,得分最高的节点成为所在组的主节点,若综合得分最高的节点存在多个则令哈希值排序最靠前的节点担任组主节点。各组主节点构成主共识层,主共识层中综合得分最高的节点担任全局主节点,负责接收客户端发来的请求和反馈。

$$S_j = \alpha \cdot \text{Rand} \cdot (1/y) \cdot \sum_{i=1}^{y} vote_{ij} + \beta \cdot T_j \qquad (3)$$

其中,$\alpha + \beta = 1$ 且 α,β 均大于 0。此处不直接使用节点获得的投票总分是为避免各组节点数量不一致对全局主节点的选取产生影响。各组主节点及全局主节点的选举流程如算法 2 所示。

算法 2:各组主节点及全局主节点选举算法
输入:G,N,y,$<$ node $>$,x
输出:$<$ PRI_NODE $>$,PRI
(1)　Initialize Global_Score $\leftarrow 0$, PRI
(2)　Initialize $<$ PRI_NODE $> \leftarrow$ NULL
(3)　for i from 1 to N
(4)　　$S_i = \alpha \cdot \text{Rand} \cdot (1/y) \cdot node_i.\ v + \beta \cdot T_i$
(5)　for i from 1 to x
(6)　　Initialize node $\leftarrow$ NULL
(7)　　Initialize score $\leftarrow 0$
(8)　　for j from 1 to y
(9)　　if $g_i.\ S_j >$ score
(10)　　score $\leftarrow g_i.\ S_j$
(11)　　node $\leftarrow g_i.\ node_j$
(12)　g_i. pri_node $\leftarrow$ node
(13)　$<$ PRI_NODE $>$. add(g_i. pri_node)
(14)　if score $>$ Global_Score
(15)　　Global_Score $\leftarrow$ score
(16)　　PRI $\leftarrow g_i$. pri_node
(17) return $<$ PRI_NODE $>$,PRI

2.5　共识流程

HBFT 算法的共识协议共 6 个阶段,共识流程如图 2 所示。

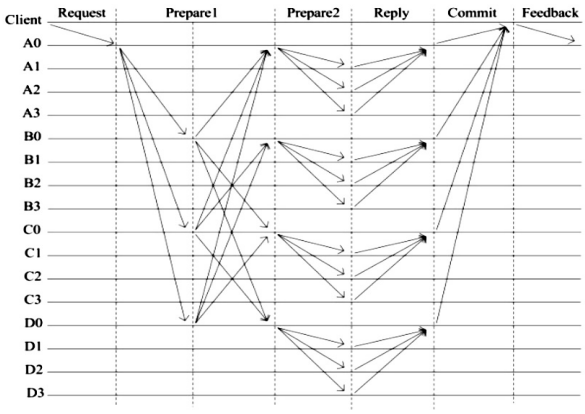

图 2　HBFT 算法共识流程

Request 阶段:客户端向系统全局主节点发送 REQUEST 消息。

Prepare1 阶段:全局主节点为接收到的请求分配序号并生成 PREPARE1 消息;随后根据信誉模型对主共识层节点和本组其它节点在上轮共识中的行为进行

判定并生成 REFRESH 消息。最后,将 PREPARE1 消息、REFRESH 消息和在上轮共识中接收到的 SUCCESS 消息广播到主共识层。

主共识层节点接收到 PREPARE1 消息后对其进行验证。当 d = digest(m) 且 c , σ_i 均验证通过生成 PREPARE1 消息;接收到 SUCCESS 消息的主节点根据上轮共识接收到的 REPLY 消息对本组节点的行为进行判定并生成 REFRESH 消息,最后将 PREPARE1 消息和 REFRESH 消息在主共识层广播。

Prepare2 阶段:主共识层节点对 PREPARE1 消息进行验证。当节点接收到包括自己在内的、验证通过且数量为主共识层节点数的一致 PREPARE1 消息后生成 PREPARE2 消息。最后将 PREPARE2 消息以及生成和接收到的所有 REFRESH 消息广播到本组并生成本轮共识的 REPLY 消息存储到本地。

Reply 阶段:节点对 PREPARE2 消息进行验证,仅当 D 中所有 s 中的 n 都与 PREPARE2 消息中的 n 相同、所有 d 都满足 d = digest(m) 时验证通过。验证通过后节点执行请求 m 并生成 REPLY 消息发送给本组主节点。此外,节点根据接收到的 REFRESH 消息更新本组各节点信誉值。

Commit 阶段:各组主节点若接收到来自本组不同节点且数量大于本组节点数 1/2 的一致 REPLY 消息,则生成 COMMIT 消息发送给客户端。

Feedback 阶段:当客户端接收到来自主共识层不同节点且数量大于主共识层节点数 2/3 的一致 COMMIT 消息时,客户端生成 SUCCESS 消息并发送给全局主节点,即共识达成。

由于主节点更换可能会导致主共识层成员的变更,因此,每次开启投票更换主节点的前一轮共识只对上轮共识中节点的行为进行判定并更新节点信誉值而不接收处理新的 REQUEST 消息。

2.6 故障处理

2.6.1 共识失败的原因分析及处理

若客户端在规定时间内未接收到足量的一致 COMMIT 消息,则本轮共识失败,原因分析如下。

情形 1:节点已收到来自组主节点的 PREPARE2 消息,但超过 1/3 组主节点用以生成 PREPARE2 消息的 PREPARE1 消息与其在主共识层广播的 PREPARE1 消息不一致;超过 1/3 组主节点未向本组节点广播 PREPARE2 消息或未向客户端发送 COMMIT 消息。

情形 2:存在组主节点生成错误的 PREPARE1 消息在主共识层广播。

上述情形 1 将由故障组节点发起视图更换请求;情形 2 将由主共识层其它诚实节点向故障组广播该组主节点的故障证明,故障证明为主共识层各节点产生的带有节点签名的 PREPARE1 消息集合。

2.6.2 故障处理流程

步骤 1:客户端重复广播未达成共识的请求和已回复 COMMIT 消息的节点名单。

步骤 2:接收到该请求的节点查询本地是否存在与该请求对应的 REPLY 消息,存在则将其发送给组主节点;不存在、本组主节点未在名单中或节点收到本组主节点的故障证明,对消息验证通过后则在本组广播 VIEW_CHANGE 消息请求更换视图。

步骤 3:临时组主节点在接收到数量大于本组节点数 2/3 的一致 VIEW_CHANGE 消息后,向主共识层广播 NEW_VIEW 消息。

步骤 4:主共识层节点对该 NEW_VIEW 消息的身份签名及集合 C 进行验证,通过则将请求 m 对应的 PREPARE1 消息发送给该临时组主节点。

步骤 5:该临时组主节点利用接收到的 PREPARE1 消息生成 PREPARE2 消息,随后生成 NEW_VIEW 消息广播到本组。

步骤 6:接收到 NEW_VIEW 消息的组内节点先对集合 C 以及临时组主节点的身份进行验证,验证通过则执行 Reply 阶段的共识流程。

3 算法分析

本章从算法通信量、安全性、容错及实验等方面对 PBFT 算法、DLBFT 算法和 HBFT 算法的性能进行分析验证。

3.1 理论分析

3.1.1 无故障情形下完成单轮共识的通信量

(1)PBFT 算法。

请求阶段,客户端仅向主节点发送请求;预准备阶段,主节点向系统广播预准备消息;准备阶段和确认阶段,各节点向系统分别广播一次准备消息和提交消息;回复阶段,每个节点向客户端发送回复。综上,PBFT 算法完成一次共识的通信量为:

$$Q_1 = 2N^2 - N + 1 \tag{4}$$

(2)DLBFT 算法。

主共识层节点在预准备阶段、准备阶段和确认阶段都需向主共识层广播一次共识消息;回复阶段,各节点仅向客户端发送回复。子层节点在准备阶段和确认阶段仅向本组广播一次共识消息,在回复阶段仅向本组主节点发送回复消息。综上,DLBFT 算法完成一轮共识的通信量为:

$$Q_2 = 2x^2 + (2N^2 - 4N + 2)/x - N + 2x + 3 \tag{5}$$

(3)HBFT 算法。

请求和反馈阶段,客户端仅向全局主节点发送一

次消息;准备阶段,主共识层节点向主层广播准备消息和更新消息,并将准备消息广播到本组;回复阶段,各节点向本组主节点发送一次回复;提交阶段,各组主节点向客户端发送提交消息。综上,HBFT 算法完成一轮共识的通信量为:

$$Q_3 = (x-1)^2 + 2N + 1 \tag{6}$$

由公式 4 ~ 6 可得公式 7 和公式 8:

$$Q_1 - Q_3 = (N^2 - x^2) + (N-1)^2 - N + 2(x-1) \tag{7}$$

$$Q_2 - Q_3 = x^2 + (2N^2 - 4N + 2)/x - 3N + 4x + 1 \tag{8}$$

由 $N > x$, $x > 1$ 和基本不等式 $a + b \geqslant 2\sqrt{ab}$ 可得公式 7、公式 8 的结果恒大于 0。综上,HBFT 算法在完成一轮共识的通信量上均恒小于 PBFT 算法和 DLBFT 算法。

3.1.2　完成一次主节点更换的通信量

(1)PBFT 算法。

节点向系统广播视图更换消息,随后新主节点接收到足量视图更换请求后向系统中广播新视图消息;当节点接收到新视图消息后对其进行验证,通过则开始 PBFT 的三阶段共识过程。综上,PBFT 算法完成一次主节点更换的通信量为:

$$W_1 = 3N^2 - 3N + 1 \tag{9}$$

(2)DLBFT 算法。

节点先向所在组广播视图更换请求,当新组主节点接收到足量视图更换请求后,先后向主共识层和所在组广播主节点更换消息;最后,该组节点对新组主节点发来的消息进行验证,通过则开始本组的 PBFT 三阶段共识。综上,DLBFT 算法完成一次故障主节点更换产生的通信量为:

$$W_2 = 3y^2 + 3y + x + 1 \tag{10}$$

(3)HBFT 算法。

系统中某组节点发现本组主节点故障,先向所在组广播主节点更换请求;当临时组主节点接收到足量视图更换请求后向主共识层广播打包好的视图更换消息并生成新的准备消息广播到本组;随后该组节点执行正常的共识过程。综上,HBFT 算法完成一次主节点更换的通信量为:

$$W_3 = y^2 + 2x - 1 \tag{11}$$

由公式 9 ~ 11 可得公式 12、公式 13:

$$W_1 - W_3 = 3N^2 - 3N - y^2 - 2x + 2 \tag{12}$$

$$W_2 - W_3 = 2 + \frac{2N^2 + 3x + 3 - 3Nx - 6N - x^3}{x^2} \tag{13}$$

由 $N \geqslant 4x$ 可将公式 13 变换为求公式 14 的值小于 0 时的条件。

$$W_4 = x(x^2 - 20x + 21) \tag{14}$$

计算得仅当 $x \in (10 - 2\sqrt{79}, 10 + 2\sqrt{79})$ 时公式 14 的值小于 0,此时 DLBFT 算法完成一次故障处理的通信量小于 HBFT 算法的。由于 $N > y$, $N \geqslant 16$,所以公式 12 的结果恒大于 0。综上,HBFT 算法在完成一次主节点更换的通信量上恒小于 PBFT 算法,且在极大多数情况下小于 DLBFT 算法。

3.2　容错改进分析

HBFT 算法要求各组主节点接收到大于本组节点数 1/2 的一致回复时开始向客户端发送 COMMIT 消息,同时要求客户端接收到大于主共识层节点数 2/3 的一致 COMMIT 消息时才判定本次共识成功,因此 HBFT 算法的最大容错可表示为:

$$\frac{N}{3} + (\frac{2N}{3} - \frac{2x}{3}) * \frac{1}{2} = \frac{2N}{3} - \frac{x}{3} > \frac{N}{3} \tag{15}$$

综上,HBFT 算法的容错能力强于 PBFT 算法的容错能力。

3.3　安全性分析

HBFT 算法基于信誉机制选取主节点,各节点对组内节点的投票依赖于被投票节点的信誉值。因此,被选举出的主节点普遍具有较高的可靠性。此外,多轮共识之后再更换主节点更有利于诚实节点积累信誉值。而 PBFT 算法和 DLBFT 算法的新任主节点仅由视图序号决定且主节点更换过于频繁,因此 PBFT 算法和 DLBFT 算法有更大的概率由故障节点担任主节点,不利于算法的安全性。综上,HBFT 算法较其它两种算法的安全性更高。

3.4　实验分析

3.4.1　实验环境

实验环境为 Intel Core i5-8250U CPU1.60 GHz 和 16 GB 内存,操作系统为 64 位 Windows10;编程语言为 golang;测试工具有 Goland 2022.3、go1.19.2。HBFT 算法的参数设置如下:

$$\begin{cases} \alpha = 0.3 \\ \beta = 0.7 \\ \text{Rand} = 20 \end{cases} \tag{16}$$

3.4.2　实验对比

实验一:系统中节点总数为 200,令节点分组数为 10,16,22,28,34,40,通过增加节点分组数比较三种算法共识 1 小时的吞吐量及完成两轮共识的共识时延。其中 HBFT-1、HBFT-2 分别为系统中无故障节点和有故障节点时的情况。实验结果如图 3、图 4 所示。

由图 3、图 4 可知,相同时间内 HBFT 算法较其他两种算法能够完成更多轮次的共识,且无论系统中是否存在故障节点,HBFT 算法的共识时延都小于另外两种算法的共识时延。这证明 HBFT 算法具有更高的

共识效率。

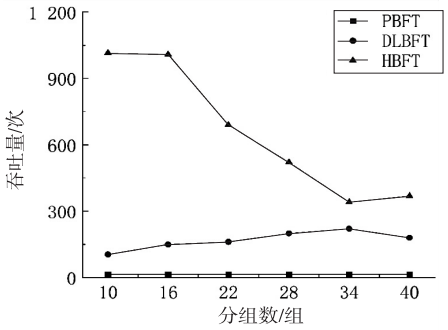

图 3 算法吞吐量对比

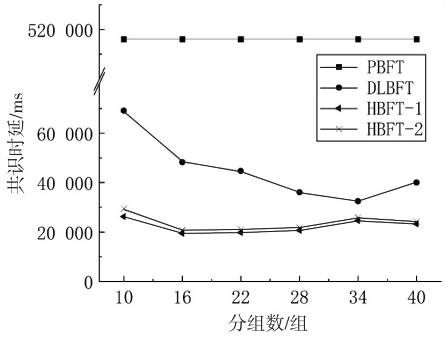

图 4 不同分组数时算法共识时延对比

实验二:节点分组数为 4 组,令系统节点总数为 32,48,64,80,96,112,通过增加节点总数比较三种算法的单轮共识时延。实验结果如图 5 所示。其中 HBFT-1、HBFT-2 分别为系统中无故障节点和有故障节点时的情况。

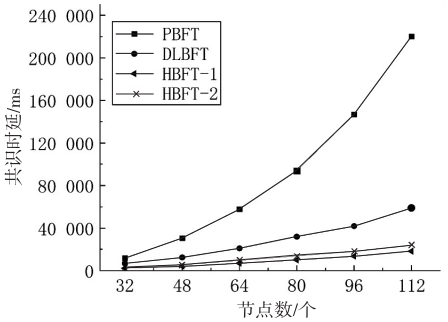

图 5 不同节点总数时算法共识时延对比

由图 5 可知,无论系统中是否存在故障节点,HBFT 算法的共识时延都小于另外两种算法的共识时延。这证明 HBFT 算法更能适应网络中节点总数的增长且能够以更快的速度达成共识。

实验三:系统中节点总数为 200,令拜占庭节点总数为 10,30,50,70,90,110,通过增加拜占庭节点数比较三种算法执行 100 轮共识发生故障的概率、完成 100 次主节点更换拜占庭节点担任系统主节点的概率,概率取 100 次实验结果的平均值。实验结果如图 6、图 7 所示。其中 DLBFT-1、HBFT-1 为分组数为 4 时的情况,DLBFT-2、HBFT-2 为分组数为 10 时的情况。

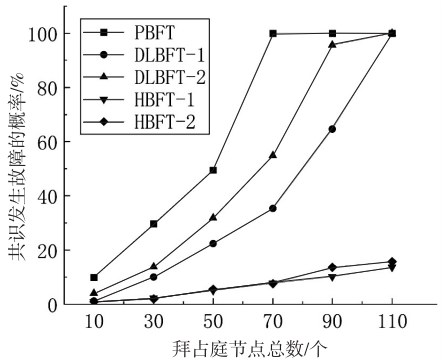

图 6 算法鲁棒性对比

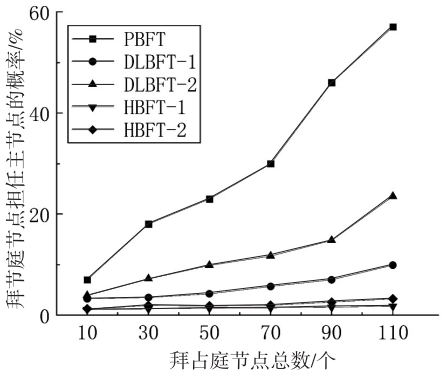

图 7 拜占庭节点担任主节点的概率

由图 6、图 7 可知,伴随系统中拜占庭节点数量的增加,HBFT 算法故障率的增长幅度较小且远低于 PBFT 算法和 DLBFT 算法;此外,HBFT 算法拜占庭节点担任主节点的概率始终保持稳定并小于另外两种算法。这表明 HBFT 算法抗拜占庭节点干扰的能力更强,算法具有更高的健壮性和容错性,并且信誉机制使诚实节点拥有更大概率担任主节点,因此 HBFT 算法具有更高的安全性。

4 结束语

为解决现有 PBFT 改进算法仍存在的通信量大、安全性差和共识效率低等问题,从节点分组原则、共识流程、通信内容和视图更换协议等方面进行改进,引入了哈希值分组策略和信誉模型,有效提高了算法的安全稳定性和共识效率。最后从理论、安全性和实验等角度,分析证明了提出的 HBFT 算法相较于 PBFT 算法和改进算法 DLBFT,有效降低了算法通信量且具有更高的共识效率、安全性和容错,更能适应大规模节点的区块链网络。

提出的 HBFT 算法在共识效率和安全性等方面具有优势,但也存在一些不足,后续将在保障系统安全性的同时进一步提升算法的共识效率。

参考文献:

[1] VACCA A,DI S A,VISAGGIO C A,et al. Asystematic lit-

erature review of blockchain and smart contract development:techniques,tools,and open challenges[J]. Journal of Systems and Software,2021,174(4):110891-110910.

[2] MATHUR S,KALLA A,GUR G. A survey on role of blockchain for IoT:applications and technical aspects[J]. Computer Networks,2023,227(3):47-128.

[3] OYINLOYE D P,TEH J S,JAMIL N,et al. Blockchain consensus:an overview of alternative protocols[J]. Symmetry,2021,13(8):1363-1398.

[4] RIJANTO A. Blockchain technology adoption in supply chain finance[J]. Journal of Theoretical and Applied Electronic Commerce Research,2021,16(7):3078-3098.

[5] ALALYAN M S,JAAFARI N A,HUSSAIN F K,et al. A systematic review of blockchain adoption in education institutions[J]. International Journal of Web and Grid Services,2023,19(2):156-184.

[6] AUHL Z,CHILAMKURTI N,ALHADAD R,et al. A comparative study of consensus mechanisms in blockchain for IoT networks[J]. Electronics,2022,11(17):2694-2717.

[7] LEE J S,CHEW C J,LIU J Y. Medical blockchain:data sharing and privacy preserving of EHR based on smart contract[J]. Journal of Information Security and Applications,2022,65(1):103117-103131.

[8] LASHKARI B,MUSILEK P. A comprehensive review of blockchain consensus mechanisms[J]. IEEE Access,2021,9:43620-43652.

[9] CASTRO M,LISKOV B. Practical byzantine fault tolerance and proactive recovery[J]. ACM Transactions on Computer Systems,2002,20(4):398-461.

[10] LAMPORT L,SHOSTAK R,PEASE M. The byzantine generals problem[J]. ACM Transactions on Programming Languages and Systems,1982,4(3):382-401.

[11] FENG L,ZHANG H,CHEN Y,et al. Scalable dynamic multi-agent practical byzantine fault-tolerant consensus in permissioned blockchain[J]. Applied Sciences,2018,8(10):1919-1941.

[12] THAI Q T,YIM J C,YOO T W,et al. Hierarchical byzantine fault-tolerance protocol for permissioned blockchain systems[J]. The Journal of Supercomputing,2019,75(11):7337-7365.

[13] 陈子豪,李　强. 基于 K-medoids 的改进 PBFT 共识机制[J]. 计算机科学,2019,46(12):101-107.

[14] LI X,LV F,XIANG F,et al. Research on key technologies of logistics information traceability model based on consortium chain[J]. IEEE Access,2020,8(8):69754-69762.

[15] QUSHTOM H,MISIC J,CHANG X,et al. A scalable two-tier pbft consensus for blockchain-based IOT data recording[C]//Proceedings of the 2021 IEEE international conference on communications. Montreal:IEEE,2021:1-6.

[16] LI W,FENG C,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.

[17] 冷基栋,吕学强,姜　阳等. 联盟链共识机制研究综述[J]. 数据分析与知识发现,2021,5(1):56-65.

[18] WANG Y L,YANG G Y,BRACCIALI A,et al. Incentive compatible and anti-compounding of wealth in proof-of-stake[J]. Information Sciences,2020,530(24):85-94.

[19] ZHANG S J,LEE J H. Analysis of the main consensus protocols of blockchain[J]. ICT Express,2020,6(2):93-97.

[20] ONGRAO D,OUSTERHOUT J. In search of an understandable consensus algorithm[C]//Proceedings of the 2014 USENIX annual technical conference. Berkeley:USENIX,2014:305-319.