计算机科学
COMPUTER SCIENCE

基于可验证随机函数的实用拜占庭共识算法

黄保华, 彭丽, 赵伟宏, 陈宁江

引用本文

黄保华, 彭丽, 赵伟宏, 陈宁江.基于可验证随机函数的实用拜占庭共识算法[J]. 计算机科学, 2023, 50(6A): 220300064-6.

HUANG Baohua, PENG Li, ZHAO Weihong, CHEN Ningjiang. Practical Byzantine Consensus Algorithm Based on Verifiable Random Functions [J]. Computer Science, 2023, 50(6A): 220300064-6.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220300064

基于可验证随机函数的实用拜占庭共识算法

黄保华　彭　丽　赵伟宏　陈宁江
广西大学计算机与电子信息学院　南宁 530004

摘　要　针对联盟链中广泛应用的实用拜占庭容错共识算法(Practical Byzantine Fault Tolerance,PBFT)主节点选取方式固定和通信成本高等问题进行了改进,提出了一种基于可验证随机函数(Verifiable Random Function,VRF)的拜占庭容错共识算法(Selection-based Byzantine Fault Tolerance,SBFT)。首先,在每轮共识后动态评测节点行为并计算节点贡献值,根据节点贡献值选取参与共识的节点。其次,结合节点贡献值和可验证随机函数进行密码抽签随机选取主节点,在减少非诚实节点成为主节点的概率的同时,使选取的主节点具有不可预测性。最后,改进了 PBFT 的一致性协议,将 PBFT 的网状通信网络拓扑变成星形通信网络拓扑,并将视图切换流程融入正常共识流程中。仿真实验结果表明,相比 PBFT 算法,所提 SBFT 算法具有更高的吞吐量、更低的共识时延和更高的算法效率。

关键词:区块链;共识算法;PBFT;VRF;节点贡献值
中图法分类号　TP391

Practical Byzantine Consensus Algorithm Based on Verifiable Random Functions

HUANG Baohua,PENG Li,ZHAO Weihong and CHEN Ningjiang
School of Computer and Electronic Information,Guangxi University,Nanning 530004,China

Abstract　To solve the problems of fixed primary node selection method and high communication cost of the practical Byzantine fault tolerance consensus algorithm widely used in alliance chain,this paper proposes a selective Byzantine fault tolerance consensus algorithm named SBFT based on verifiable random function. The first proposal is to dynamically calculate node contribution value by evaluating the node behavior after each round of consensus,and select the nodes participating in consensus based on the node contribution value. Next,a combination of node contribution value and verifiable random function is used for random selection of primary nodes by cryptographic sortation,which makes the selected primary node unpredictable while reducing the probability of non-honest nodes becoming primary node. Finally,the consistency protocol of PBFT is improved by changing the mesh communication network topology of PBFT into a star communication network topology and incorporating the view replacement process into the normal consensus process. Simulation experimental results show that the proposed SBFT algorithm has higher throughput,lower consensus latency and higher algorithmic efficiency compared with the PBFT algorithm.

Keywords　Blockchain,Consensus algorithm,PBFT,Verifiable random function,Contribution value of nodes

1　引言

区块链是一种可以实现数据一致存储的分布式账本技术。数据按区块的形式进行存储,区块之间通过记录前一区块的哈希值,链接成一个可验证、不可篡改的链。由于区块链中区块数据并不止存在于一个分布式节点中,因此在不信任区块链中任何节点的情况下,仍可信任区块链网络中记录的数据。因此,如何保证各节点存储的区块数据和区块的链接关系的一致性是区块链工作的核心,共识算法则是用于解决区块链中这一问题的方案。

1980 年,Pease 等提出了分布式系统中的一致性问题[1],即当分布式系统中存在因网络拥塞或硬件故障而不能及时响应消息的故障节点时,其中的节点要如何对某个决策或某个状态的变化达成一致。1982 年,Lamport 等提出了“拜占庭将军问题”[2],提出当分布式系统中存在“拜占庭将军”中的叛徒将军,即拜占庭节点时,要如何保证一致性。与之前提出的故障节点不同,拜占庭节点是一种恶意节点,它们可以采取任何行动以干扰系统达成共识,比如故意不响应消息或响应错误的消息。

根据能否容忍拜占庭节点,共识算法被分为了两类:拜占庭容错(Byzantine Fault Tolerance,BFT)共识算法和非 BFT 共识算法。非 BFT 共识算法中比较经典的有 1989 年由 Lamport 提出的 Paxos 共识算法[3]和由 Ongaro 提出的简化的 Paxos 算法,被称为 Raft[4]。此后大部分的非 BFT 共识算法研究都基于这两个工作展开。

第一个实用有效的能在半同步环境中解决拜占庭故障的 BFT 共识算法是由 Castro 等于 1999 年提出的实用拜占庭容错共识算法(PBFT)[5]。之后,中本聪在 2008 年提出了基于

基金项目:国家自然科学基金(61962005);国家重点研发计划(2018YFB1404404)
This work was supported by the National Natural Science Foundation of China(61962005) and National Key Research and Development Program of China(2018YFB1404404).
通信作者:黄保华 (bhhuang66@gxu.edu.cn)

哈希碰撞的工作量证明(Proof of Work,PoW)共识算法[6]。PoW 算法中,所有想要成为出块者的节点需要不断地寻找一个满足系统要求的随机数,这个过程会浪费大量哈希功率。为了减少这种浪费,权益证明共识算法(Proof of Stake,PoS)在 2011 年被提出[7]。在 PoS 中,每个节点不再根据哈希功率来寻找随机数,而是依靠自己所持有的权益来寻找随机数,在一定程度上减少了哈希功率的浪费。文献[8]将比特股(Bitshare)与委托投票制度和 PoS 相结合,提出了 DPoS(Delegation Proof of Stake)共识算法,在 PoS 的基础上进一步提高了共识效率,且不再需要消耗哈希功率,其出块时间短,效率高,但容易被中心化。相较于 PoW,PoS 和 DPoS,PBFT 是一种没有分叉的、可以容忍拜占庭节点的确定性共识算法,可以被看作是一种比较理想的共识算法。然而,PBFT 也存在着一些缺陷,如主节点选取方式固定、不能在动态网络环境中工作和需要较大的通信开销才能达成一致等。

针对此类问题,国内外学者主要从优化 PBFT 机制、改进 PBFT 架构等方面进行了研究。文献[9-11]从层次结构方面对 PBFT 进行了优化,提高了可扩展性和协商效率。文献[12]提出了线性视图改变(Linear View Change,LVC)方案,并将共识过程进行流水化处理。文献[13]提出了 SBFT 共识机制,使用收集器汇总消息,将 PBFT 变为线性 PBFT,显著降低了通信量。区块链项目瑞波(Ripple)[14]使用了 FBFT 共识机制,它弱化了主节点和备份节点的概念,只由信任节点列表(Unique Node List,UNL)中的节点对交易进行投票。Gao 等基于 EigenTrust 信任模型提出了 T-PBFT 算法[15],该算法使用主组替换单个主节点,并使用信用模型评估节点信任度,选择网络中高质量的节点来构建共识组,从而减少视图改变的概率、提高系统的容错性。Tong 等结合 PeerTrust 信任计算模型和 PBFT 共识算法提出了 Trust-PBFT 共识算法[16],通过选取信任度高的节点参与到交易中,以提升共识机制的容错能力、系统的吞吐量和系统可扩展性。

Algorand 共识算法首次提出将可验证随机函数 VRF[17]与 BFT 相结合应用到共识算法中[18-19],Algorand 共识算法结合 PoS 和 VRF 随机选择验证节点和出块节点,其安全性高、吞吐量大,能抵抗恶意攻击,但通信开销高。Ontology 项目结合 PoS 和 VRF 提出了 VBFT 共识算法[20]。VBFT 将节点分类为出块节点、验证节点、确认节点和候补节点,使用 VRF 随机选择除候补节点以外的 3 类节点。其安全性大、吞吐量大、时延低,但用户需通过质押代币才能参与到共识网络中。Wu 等基于 PoS 和 PBFT 提出了一个优化的混合共识算法[21],其使用 VRF 密码抽签将参与共识的节点个数减少到一个常数,其吞吐量、延迟和可扩展性都比 PBFT 算法更优。Zhan 等基于 PBFT 提出委托随机化拜占庭容错共识协议 DRBFT[22],并开发了一种称为 RS 的随机选择算法来配合投票机制,减少参与协商一致过程的节点数量,此方案具有不可预测性、随机性和公正性。文献[23]结合 VRF、PoS 和公证制度提出了 Dfinity 共识机制,由随机选取的公证委员会中的公证人提议提案。文献[24]使用锁机制深度改进 PBFT 共识机制,提出了 Tendermint 共识,它对预投票消息进行加锁,保证数据的正确性和一致性,因此无需视图变更协议和错误恢复协议,但 Tendermint 共识机制较为复杂,没有对应的现实世界信任模型。

在对比分析了已有的一些共识算法后,本文提出了一种改进的拜占庭共识算法 SBFT,通过结合节点贡献值和可验证随机函数 VRF 来优化共识节点和主节点的选择,使 SBFT 可以更好地抵抗恶意攻击,减少共识失败的概率,增大系统吞吐量并减少交易时延。此外,我们还简化了 PBFT 的一致性协议,降低了共识所需的交易时延和通信开销。

2 PBFT 共识算法概述

2.1 基本概念

PBFT 主要由一致性协议、视图更改协议和检查点协议组成。一致性协议用于保证客户端的请求至少被所有诚实的节点以相同的顺序处理,并且能够返回正确和一致的结果给客户端。视图更改协议用于保持共识算法的活性,保证共识失败时能继续正常工作。检查点协议用于设置检查点,节点根据检查点可以安全地丢弃日志中的信息,以减少节点的内存开销,也可以用来帮助宕机节点同步最新状态。

PBFT 是基于视图的共识协议,共识过程由一个接一个的视图组成。在一个视图中,存在一个确定的主节点来主导共识协议,其他节点为参与共识的备份节点。假设共有 N 个节点,PBFT 共识算法可以容忍的最大拜占庭节点数为 f,f 与 N 的关系如式(1)所示,即当且仅当 $N\geqslant 3f+1$ 时,PBFT 算法可以正常进行共识。

$$f=\lfloor (N-1)/3 \rfloor \tag{1}$$

在 PBFT 中,视图按顺序编号,通过视图变更以累加的形式编号,节点则按 0 到 $N-1$ 依次编号。每个视图中主节点的编号 p 由视图编号 v 和共识节点个数 N 按式(2)计算决定。

$$p=v \bmod N \tag{2}$$

当某个视图超时未能达成共识或主节点出错时,将执行视图更改协议,切换到下一个视图继续进行共识。

2.2 共识过程

PBFT 的一致性协议,即共识过程属于典型的三阶段,具体为预准备、准备和提交 3 个主要阶段。执行过程如图 1 所示,具体步骤如下。

(1)请求阶段:客户机 c 发起请求,并将请求消息〈*REQUEST*,o,t,c〉发送到共识网络。其中,o 表示请求的具体操作,t 表示请求的时间戳。

(2)预准备阶段:主节点收到验证通过的服务请求后,在当前视图 v 中为这个请求分配请求标记序号 seq,并向所有备份节点广播预准备消息〈〈*PRE-PREPARE*,v,seq,d〉,m〉。其中,m 为请求消息,d 为消息 m 的摘要。

(3)准备阶段:备份节点 i 在收到验证通过的预准备消息后,向其他备份节点广播准备消息〈*PREPARE*,v,seq,d,i〉。

(4)提交阶段:备份节点 i 在收到验证通过的准备消息后,将其写入日志。如果累积收到 $2f+1$ 个来自不同节点(包括其自身)的验证通过的准备消息,则向所有备份节点广播提交消息〈*COMMIT*,v,seq,d,i〉。

(5)答复阶段:备份节点 i 在接收到 $2f+1$ 个(包括其自身)验证通过的提交消息后,执行请求的具体操作,并向客户端发送答复消息〈*REPLY*,v,t,c,i,r〉,r 为执行请求操作的结果。客户端在收到 $f+1$ 个不同备份节点的正确答复时,则认为请求已经得到执行。

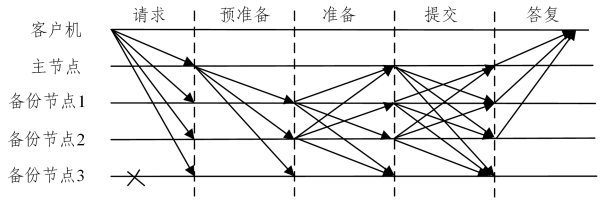

图 1 PBFT 算法共识算法执行过程

Fig. 1 Execution flow of PBFT consensus algorithm

2.3 PBFT 算法的不足

对 PBFT 的流程进行详细分析可知,PBFT 算法存在以下几点不足:

(1)主节点选取方式固定。主节点的编号由式(2)计算得到,这种轮换的方式不能保证主节点的可靠性,容易被敌手操控。在之前的视图中已知的恶意节点或故障节点随着视图变更仍会作为主节点主导共识,使系统的可靠性和安全性大幅降低。

(2)通信开销大。PBFT 的一致性协议包括两次多对多的全网广播,非常消耗网络带宽。当请求量过多时,网络极易出现拥堵。

(3)系统可扩展性差:没有灵活的共识节点加入和退出机制,无法满足系统节点变更的需求,且使得已知的恶意节点或故障节点一直存在并参与共识,系统的安全性、可用性和可扩展性差。

3 改进的共识算法

本文提出的 SBFT 算法在每轮共识后动态评测节点行为,计算节点贡献值,然后根据节点贡献值选取参与共识的节点;再结合节点贡献值和可验证随机函数进行主节点选取,在减少非诚实节点成为主节点的概率的同时,使选取的主节点具有不可预测性。最后,在选取更可靠的主节点的基础上改进了 PBFT 的一致性协议,减少了共识所需的时间和通信开销。

3.1 节点贡献值

3.1.1 节点贡献值计算

本文提出的节点贡献值(Contribution Value)根据节点累积在各轮共识中做出的贡献进行计算,在每一轮共识过程结束后,动态更新节点的贡献值。节点的初始贡献值为 CV_{init},节点 i 在参与视图 v 的共识后,其贡献值 $TotalCV_v^i$ 按式(3)进行计算:

$$TotalCV_v^i = TotalCV_{v'}^i * (1 + Succ_v * Pri_v^i * d) \tag{3}$$

其中,$TotalCV_{v'}^i$ 为节点 i 在参与视图 v 前的贡献值,v' 是节点 i 在视图 v 之前上一次参与共识的视图编号。$Succ_v$ 是与视图 v 中共识是否完成相关的参数,若在视图 v 中达成共识,$Succ_v$ 取值为正数,此时,$TotalCV_v^i$ 的值增大;反之,若未完成共识,则 $Succ_v$ 取值为负数,$TotalCV_v^i$ 的值减小。Pri_v^i 的取值由节点 i 是否作为主节点参与视图 v 的共识决定,用于区别主节点与共识节点的贡献量,保证在成功地共识后,主节点贡献值增长比共识节点大,失败时贡献值减少量比共识节点大。δ 为计算贡献值的基础增量。当节点被检测到存在两次以上不诚实行为时,其贡献值将降到最低阈值 CV_{low} 之下。

3.1.2 贡献值恢复机制

为防止节点因为贡献值过高而产生惰性行为或趋于集中化,我们设计了一个节点贡献值恢复机制。通过保持一个贡献值阈值区间 $[CV_{\text{low}}, CV_{\text{high}}]$,在执行节点贡献度恢复机制时,将贡献值大于 CV_{high} 的节点贡献值恢复到一个在区间 $[CV_{\text{init}}, CV_{\text{high}}]$ 中的随机值,将节点贡献值在区间 $[CV_{\text{low}} + (CV_{\text{init}} - CV_{\text{low}})/2, CV_{\text{init}}]$ 内的恢复到 CV_{init}。通过节点贡献值恢复机制可以保证低贡献值节点的积极性,也可以降低节点因贡献值过高而产生惰性行为或恶意行为的可能性。节点贡献值恢复机制的流程如算法 1 所示。

算法 1 节点贡献值恢复算法

输入:C,CV_{init},CV_{low},CV_{high},N

输出:C

```
1. / * C 为节点贡献值数组,N 为节点个数 * /
2. t ←CVlow + (CVinit − CVlow)/2;
3. for i←0 to N do
4.    if C [i] > CVhigh
5.       C [i] ← Random(CVinit, CVhigh);
6.    else if C [i] >= t and C [i] < CVinit
7.       C [i] ← CVinit;
8.    end if
9. end for
10. return C.
```

3.2 节点选取

在 SBFT 中,将节点分类成主节点、共识节点和候选共识节点。SBFT 中主节点由节点贡献值作为节点权重使用可验证随机函数密码抽签确定,由共识网络中所有 $TotalCV_v^i$ 大于 $CV_{\text{low}} + (CV_{\text{init}} - CV_{\text{low}})/2$ 的节点独立地进行抽签。设共有 N 个节点,每轮按贡献值排序选取前 n 个节点作为共识节点参与共识,其他候选共识节点与共识网络保持同步。

3.2.1 主节点选取

SBFT 中主节点的选取由共识网络中所有 $TotalCV_v^i$ 大于 $CV_{\text{low}} + (CV_{\text{init}} - CV_{\text{low}})/2$ 的节点使用可验证随机函数(VRF)密码抽签确定。VRF 具有可验证性、唯一性和随机性三大特性。每个节点都持有一个密钥对(公钥 pk_i,私钥 sk_i)。VRF 的计算函数 $VRF\_Evaluate$ 在输入一样的密钥对和消息 x 的情况下,只会产生唯一的输出 vrf_hash 和 $proof$。vrf_hash 是由 sk_i 和 x 唯一确定的哈希长度的伪随机字符串,证明 proof 可以供拥有 pk_i 的人使用 VRF_Verify 函数验证 vrf_hash 是否确实对应 x。SBFT 的主节点选取基于 VRF 实现密码抽签,使主节点的选取具有不可预测性,但仍不能抵抗女巫攻击,因此本文结合节点贡献值,以更好地抵抗女巫攻击。

在 SBFT 中,节点 i 使用可验证随机函数密码抽签流程如算法 2 所示。节点使用 VRF 求出哈希值 vrf_hash,vrf_hash 在区间 $[0, 2^{\text{hashlen}}]$ 内均匀分布,取 $e = vrf\_hash/2^{\text{hashlen}}$,即使 e 均匀地分布在 $[0,1]$ 之间。设置一个期望值 σ,参与共识节点的总贡献值为 CV_Sum,根据概率 $\mu = \sigma/CV\_Sum$ 及节点的贡献值 CV_i,计算节点 i 以概率 μ 在 CV_i 次伯努利实验中成功 k 次的概率,记作 $B(k; CV_i, \mu)$。将 k 取值从 0 到 j 的和相加,求出 e 落在区间 I 内时的 j 值,区间 I 的计算式如式(4)所示。

$$I = [\sum_{k=0}^{j} B(k; CV_i, \mu), \sum_{k=0}^{j+1} B(k; CV_i, \mu)] \tag{4}$$

满足条件的 j 则为节点 i 的抽签结果,当 j 为 0 时,表示

节点 i 抽签失败，大于 0 表示抽签成功。多个节点抽签成功时，抽签结果 j 值最大的成为主节点，有相同抽签值时，节点编号最小的成为主节点。

算法 2 密码抽签算法

输入：sk_i, seed, σ, CV_i, CV_Sum

输出：vrf_hash, proof, j

1. /* seed 为这一轮共识的安全参数 */
2. vrf_hash, proof ← VRF_Evaluate(sk_i, seed);
3. e←vrf_hash/$2^{hashlen}$;
4. μ←σ/CV_Sum;
5. j←0;
6. while e$<\sum_{k=0}^{j} B(k;CV_i,\mu)$
7. j++;
8. end while
9. return vrf_hash, proof, j.

在每一轮共识中需要一个不断更新的、随机选择的、公开的安全参数 *seed* 作为函数 *VRF_Evaluate* 的输入。初始的 *seed* 使用分布式随机数生成随机选择，视图 v 的 $seed_v$ 由视图 $v-1$ 中的主节点 i 使用式(5)计算得到。

$$seed_v, proof\_seed = VRF\_Evaluate(sk_i, seed_{v-1} \| v) \quad (5)$$

在视图 $v-1$ 中，抽签成功的节点需将其计算的 $seed_v$ 和 *proof_seed* 同区块一起发送到网络中，其他节点收到后需验证区块、抽签值以及下一轮的 *seed* 值。其他节点在验证节点 i 的抽签结果时，根据节点 i 发送的 *vrf_hash*, *proof* 和抽签结果 j，使用本节点保持的验证密钥 pk_i 及这一轮共识的公共参数 *seed* 来验证抽签结果，验证过程如算法 3 所示。

算法 3 验证算法

输入：pk_i, seed, σ, CV_i, CV_Sum, vrf_hash, proof, j

输出：true/false

1. if ¬VRF_Verify(pk_i, seed, vrf_hash, proof)
2. return false;
3. end if
4. e ←vrf_hash/$2^{hashlen}$;
5. μ←σ/CV_Sum;
6. j′←0;
7. while e$<\sum_{k=0}^{j'} B(k;CV_i,\mu)$
8. j′++;
9. end while
10. if j !=j′
11. return false;
12. end if
13. return true.

3.3 共识过程

相较于 PBFT 的典型的三阶段共识过程，SBFT 在选择更可信的、更可靠的主节点的基础上用一对多的通信替换了多对多的通信，并将视图切换流程融入正常共识过程中。在 PBFT 中，只在出现异常情况并有足够多的节点期望更换视图时，才执行视图更换协议。而在 SBFT 中，每次共识后，无论是否成功，都重新选取共识节点并切换到下一个视图继续进行共识。为保持算法的活性，我们使用超时机制保证视图的更换。

SBFT 把 PBFT 的两阶段确认扩展成了三阶段确认，即四阶段共识过程。其流程如图 2 所示，具体共识过程如下：

(1)请求阶段：客户机 c 发起请求，并将请求消息〈*REQUEST*, o, t, c〉发送到共识网络。其中，o 表示请求的具体操作，t 表示请求的时间戳。

(2)预准备阶段：主节点收到验证通过的服务请求后，在当前视图 v 中为这个请求分配请求标记序号 *seq*，并向所有共识节点广播预准备消息〈〈*PRE-PREPARE*, v, *seq*, d, s〉, m〉。其中 m 为请求消息，d 为消息 m 的摘要，s 为抽签数据，包括 *vrf_hash*, *proof* 和 j。

(3)准备阶段：共识节点 i 收到验证通过的预准备消息后，向主节点发送准备消息〈*PREPARE*, v, *seq*, d, i〉。

(4)预提交阶段：主节点将接收到的验证通过的准备消息都写入日志，当接收到 $2f+1$ 个来自不同共识节点(包括其自身)验证通过的准备消息时，向所有共识节点广播预提交消息〈*PRE-COMMIT*, v, *seq*, d, i〉。

(5)提交阶段：共识节点 i 在收到验证通过的提交消息后执行请求的具体操作，并向主节点发送确认消息〈*COMMIT*, v, *seq*, d, i〉。

(6)答复阶段：主节点接收到 $2f+1$ 个确认消息后执行请求的具体操作，并向客户端和其他共识节点发送答复消息〈*REPLY*, v, t, c, i, r〉，r 为执行请求操作的结果。客户端若收到主节点的正确答复，则认为请求已经得到执行。

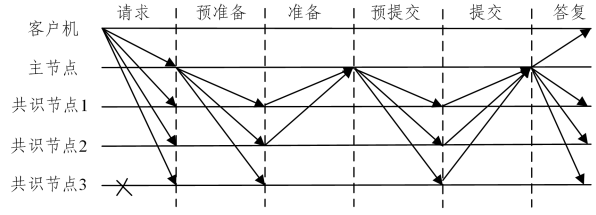

图 2 SBFT 算法共识算法执行过程

Fig. 2 Execution flow of SBFT consensus algorithm

4 实验及结果分析

本文实验在配置为 Intel i7-9700 3.00GHz 处理器、16GB 内存的 Windows 10 系统下进行，通过 Go 语言实现 SBFT 和 PBFT 算法，使用线程监听不同的端口代替节点，开启多个线程模拟共识节点的通信过程。通过分析与 PBFT 算法的通信量、吞吐量、交易时延和容错性对比来验证 SBFT 算法的有效性。实验中的参数设置如表 1 所列。

表 1 验参数表

Table 1 Experimental parameters

参数	设置
节点贡献值初始值 CV_{init}	50
节点贡献值阈值 CV_{low}, CV_{high}	20,100
标志共识是否成功 $Succ_v$	1,−1
标志是否作为主节点参与共识 Pri_v^i	1,2
节点贡献值计算的基础增量 δ	1.0
一轮共识过程包含的交易数 *Transactions*	2000
超时阈值/s	10

4.1 通信量分析

(1)通信量分析

假设系统中总节点个数为 N。在 PBFT 的共识过程中，预准备阶段消息交互量为($N-1$)，准备阶段和提交阶段的消息交互量都为 $N*(N-1)$。而发生一次视图更换所需通信次数为 $N*(N-1)+(N-1)$，假设 g 为发生视图变更的

概率，则 PBFT 共识过程的通信总次数为$(2+g)N^2-N-g-1$。

设在 SBFT 的共识过程中，选取参与共识的节点为 n，令 $n=hN$，$0<h\leqslant 1$，其 4 个共识阶段的消息交互量都为 $hN-1$。在每次共识结束后 SBFT 都会进行一次多对多通信进入新视图，故 SBFT 共识过程总交互信息量为 $h^2N^2+3hN-4$。

(2)通信开销比较

令 PBFT 算法与 SBFT 算法的通信开销之比为 Y，则有：

$$Y=((2+g)N^2-N-g-1)/(h^2N^2+3hN-4) \quad (6)$$

分别取 $g=0$，$h=2/3$；$g=0$，$h=1$；$g=1/2$，$h=2/3$；$g=1/2$，$h=1$ 进行实验，对比图如图 3 所示。可以看出，即使是在 $g=0$，$h=1$，即 PBFT 中不发生视图变更且 SBFT 中全部节点都参与共识的情况下，PBFT 的通信量约为 SBFT 的 1.5 倍。

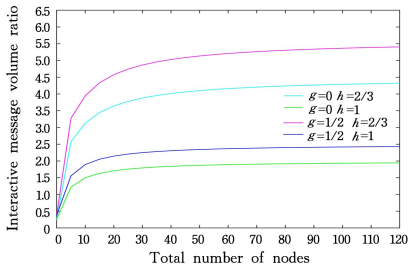

图 3　共识交互消息量对比图

Fig. 3　Diagram of consensus interaction message volume comparison

4.2　吞吐量分析

为测试对比 PBFT 算法和 SBFT 算法的吞吐量，实验在存在拜占庭节点和不存在拜占庭节点两种情况下，取节点个数为 4，7，10，13，16，19，22，相对应地，在存在拜占庭节点的情况下取拜占庭节点个数为 1，2，3，4，5，6，7 进行了多次实验，统计了节点每秒达成共识的交易数的平均值，对比结果如图 4 和图 5 所示。可以看出，在不同节点数量的情况下，不管拜占庭节点存在与否，SBFT 算法的吞吐量都大于 PBFT。这是由于 SBFT 共识规模小，参与共识的节点更可信、更可靠，具有能快速达成共识的优势。

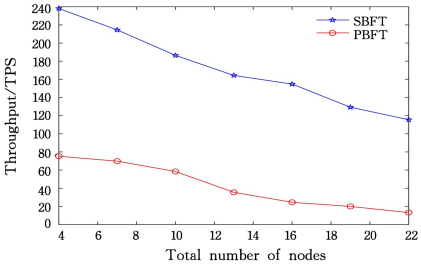

图 4　不存在拜占庭节点时吞吐量对比图

Fig. 4　Throughput comparison chart without Byzantine nodes

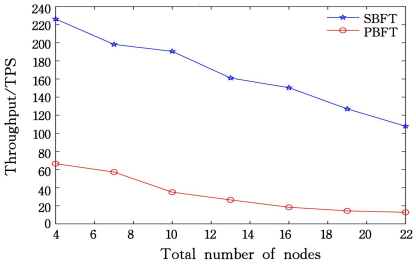

图 5　存在拜占庭节点时吞吐量对比图

Fig. 5　Throughput comparison chart with Byzantine nodes

4.3　交易时延分析

实验取 4，7，10，13，16，19，22 个节点的数量值，在存在拜占庭节点和不存在拜占庭节点的两种情况下对交易从产生到被执行所需时延进行了测试，在存在拜占庭节点的情况下取最大能容忍的拜占庭数作为实验数值，根据多次试验的平均值得到两种算法的比较结果。对比图如图 6 和图 7 所示。在节点个数增多时，两种算法的时延都随之增大。但是 SBFT 算法的时延始终小于 PBFT 算法，且其时延增长较慢。对比图 6 和图 7 可以发现，相较于不存在拜占庭节点的情况，SPFT 在存在拜占庭节点的情况下时延的增幅并不大，而 PBFT 在存在拜占庭节点的情况下时延增加较快，这是由于 SPFT 中拜占庭节点参与共识的可能性小，而 PBFT 的拜占庭节点一直存在，会导致较多的视图更换。

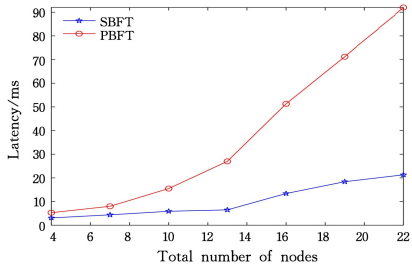

图 6　不存在拜占庭节点时时延对比图

Fig. 6　Latency comparison chart without Byzantine nodes

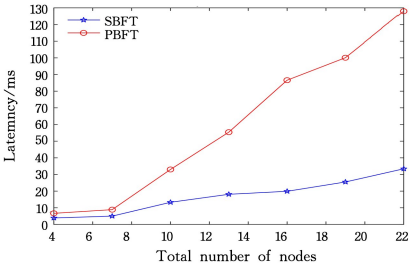

图 7　存在拜占庭节点时时延对比图

Fig. 7　Latency comparison chart with Byzantine nodes

4.4　容错性分析

在总节点数为 N 时，PBFT 算法最大可容忍拜占庭节点数如式(1)所示。而 SBFT 优化了 PBFT 的拜占庭容错率。令 $n=hN$，$0<h\leqslant 1$，当系统选择贡献值排名前 n 的节点作为共识节点参与共识时，在共识过程中，为保证共识的正确性，主节点需至少收到 $hN-f$ 个来自不同节点的消息，其中，接收的验证通过的正确消息应多于 f。因此需要使 $hN-f-f\geqslant f$，即 $f\leqslant(hN-1)/3$。在剩下的$(1-h)N$ 个不参与共识的节点中，在极端情况下，都可为拜占庭节点。此时，SBFT 可容忍的最大拜占庭数 $f'=f+(1-h)N$，只在 $h=1$ 时，即 SBFT 中所有节点都参与共识时，其容错率才会降到和 PBFT 一样。

结束语　本文基于 PBFT 算法，提出了一种优化的共识算法 SBFT。通过计算节点贡献值选取共识节点，使用可验证随机函数随机选取主节点，使主节点和共识节点更加可靠，既能有效地抵抗恶意攻击，也能降低共识失败的可能性。并在选取更可信可靠的主节点的基础上改进了 PBFT 的一致性协议，减少了共识所需的时间和通信开销。实验结果表明，在相同的条件下与 PBFT 算法对比，本文提出的 SBFT 算法有效降低了网络中的通信开销，提高了吞吐量并减少了交易

时延。在未来的工作中,可以对算法中节点贡献值的计算进行改进,增加算法的动态性和适应性,并考虑结合门限签名加快共识的达成。

参 考 文 献

[1] PEASE M, SHOSTAK R, LAMPORT L. Reaching agreement in the presence of faults[J]. Journal of the ACM(JACM), 1980, 27(2): 228-234.

[2] LAMPORT L, SHOSTAK R, PEASE M. The Byzantine generals problem[J]. ACM Transactions on Prograomming and System, 1982, 4(3): 382-401.

[3] LAMPORT L. The Part-Time Parliament [J]. ACM Transactions on Computer Systems, 1998, 16(2): 133-169.

[4] DIEGO O, JOHN O. In search of an understandable consensus algorithm[C]// Proceedings of the 2014 USENIX Annual Technical Conference(USENIX ATC 2014). 2014: 305-319.

[5] CASTRO M, LISKOV B. Practical Byzantine Fault Tolerance [C]// Proceedings of the Third Symposium on Operating Systems Design and Implementation. 1999: 1-14.

[6] NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system [EB/OL]. [2020-06-10]. https://bitcoin.org/bitcoin.pdf.

[7] KING S, NADAL S. PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake[EB/OL]. [2020-04-10]. https://www.researchgate.net/publication/265116876_PPCoin_Peer-to-Peer_Crypto-Currency_with_Proof-of-Stake.

[8] SCHUH F, LARIMER D. Bitshares 2.0: Financial smart contract platform[EB/OL]. [2020-04-10]. https://www.weusecoins.com/assets/pdf/library/Bitshares% 20Financial% 20Platform.pdf.

[9] 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.

[10] LI Y, QIAO L, LV Z. An optimized byzantine fault tolerance algorithm for consortium blockchain[J]. Peer-to-Peer Networking and Applications, 2021, 14(5): 2826-2839.

[11] CHEN Z H, LI Q. Improved PBFT Consensus Mechanism based on K-medoids[J]. Computer Science, 2019, 46(12): 101-107.

[12] ABRAHAM I, GUETA G, MALKHI D. Hot-stuff the linear, optimal-resilience, one-message BFT devil [J]. arXiv: 1803.05069, 2018.

[13] GUETA G G, ABRAHAM I, GROSSMAN S, et al. SBFT: A Scalable and Decentralized Trust Infrastructure[C]// 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks(DSN). 2019: 568-580.

[14] SCHWARTZ D, YOUNGS N, BRITTO A. The Ripple Protocol Consensus Algorithm[EB/OL]. [2020-06-10]. https://ripple.com/files/ripple_consensus_whitepaper.pdf.

[15] SHENG GAO T Y. T-PBFT An EigenTrust-based practical Byzantine fault tolerance consensus algorithm[J]. China Communications, 2019, 16(12): 111-123.

[16] TONG W, DONG X, ZHENG J. Trust-PBFT: A PeerTrust-Based Practical Byzantine Consensus Algorithm[C]// 2019 International Conference on Networking and Network Applications(NaNA). 2019: 344-349.

[17] MICALI S, RABIN M, VADHAN S. Verifiable random functions[C]// Proceedings of the 40th Annual Symposium on Foundations of Computer Science. 1999: 120-130.

[18] LEUNG D, SUHL A, GILAD Y, et al. Vault: Fast Bootstrapping for the Algorand Cryptocurrency[C]// Proceedings of the Network and Distributed Systems Security(NDSS) Symposium. 2019: 1-15.

[19] GILAD Y, HEMO R, MICALI S, et al. Algorand: Scaling Byzantine Agreements for Cryptocurrencies[C]// Proceedings of the Twenty-Sixth ACM Symposium on Operating Systems Principles. 2017: 51-68.

[20] Ontology Project. Consenses mechanism [EB/OL]. [2020-06-14]. https://docs.ont.io/ontology-elements/consensus-mechanism.

[21] WU Y, SONG P, WANG F. Hybrid Consensus Algorithm Optimization: A Mathematical Method Based on POS and PBFT and Its Application in Blockchain[J]. Mathematical Problems in Engineering, 2020, 2020: 7270624.

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

[23] HANKE T, MOVAHEDI M, WILLIAMS D. DFINITY Technology Overview Series, Consensus System[J]. arXiv: 1805.04548, 2018.

[24] KWON J. Tendermint: Consensus without mining [EB/OL]. [2020-06-14]. https://tendermint.com/static/docs/tendermint.pdf.

HUANG Baohua, born in 1973, Ph.D, associate professor, postgraduate supervisor, is a senior member of China Computer Federation. His main research interests include database security, vehicular ad hoc network security, blockchain, etc.