结合 BLS 聚合签名改进实用拜占庭容错共识算法*

陈佳伟，冼祥斌，杨振国，刘文印†
（广东工业大学 计算机学院，广州 510006）

摘 要： 针对实用拜占庭容错（practical Byzantine fault tolerance，PBFT）共识算法运用在联盟链中达到 $O(n^2)$ 的通信复杂度难以支持大规模网络的问题，提出一种聚合签名的拜占庭容错算法（aggregate-signature byzantine fault tolerance，ABFT）。首先，改进 PBFT 共识算法中节点的信息交互方式，在 prepare 阶段各个副本节点单点发送信息及签名给主节点验证，在 commit 阶段由主节点收集签名并验证，结合 BLS（boneh-lynn-shacham）签名将验证通过的多个签名聚合成一个聚合签名，将该聚合签名以及其他必要信息广播给其他所有副本节点验证；此外增加了 finish 阶段，用于防止大部分的副本节点超时而导致视图变更。ABFT 算法将网络通信的复杂度降低为 $O(n)$，通过实验表明，在多个节点的情况下，ABFT 算法有效地降低了共识的时延且提高了交易吞吐量，可扩展性更优，使联盟链可容纳大量节点。

关键词： 区块链；PBFT 共识算法；聚合签名；吞吐量；时延
中图分类号： TP301.6 **文献标志码：** A **文章编号：** 1001-3695(2021)07-006-1952-04
doi:10.19734/j.issn.1001-3695.2020.12.0403

Improved practical Byzantine fault tolerant consensus algorithm combined with BLS aggregating signature

Chen Jiawei, Xian Xiangbin, Yang Zhenguo, Liu Wenyin†
(*School of Computers, Guangdong University of Technology, Guangzhou* 510006, *China*)

Abstract: To solve the problem that the PBFT consensus algorithm can't support large-scale network with the quadratic communication complexity of $O(n^2)$ in consortium blockchain, this paper proposed an ABFT algorithm. Firstly, it improved the information interaction mode of nodes in PBFT consensus algorithm on multiple phases. On the prepare phase, each replica node sent information and signatured to the master node for verification. On the commit phase, the master node collected and verified the signatures, and combined with BLS signature, it aggregated the multiple verified signatures into an aggregating signature, which would be broadcast to all other replica nodes with other necessary information for verification. Moreover, ABFT algorithm the added the finish phase to prevent most replica nodes from timing out and causing view change. The ABFT algorithm reduced the complexity of network communication to $O(n)$. Experiments show that in the case of multiple nodes, the ABFT algorithm can effectively reduce the consensus delay and improve the transaction throughput, which makes the consortium blockchain being able to accommodate a large number of nodes.

Key words: blockchain; PBFT consensus algorithm; aggregating signature; throughput; time delay

0 引言

区块链[1]技术近年来由于区块链底层技术的创新及一些业务场景的落地得到了全世界范围的关注。区块链作为一种新型技术被发现及研究的起源是中本聪在 2008 年底发表的白皮书《比特币：一个点对点的电子货币体系》[2]，从那之后比特币逐渐走入人们的视野，而区块链技术作为比特币的底层技术，也逐渐被一些学者和爱好者了解及研究。中本聪结合点对点分布式技术、加密技术、分布式数据存储、哈希算法、共识机制等多种技术创建了一个完全去中心化[3]的电子现金系统，由于采用去中心化的区块链设计，节点各处分散、难以统一所有节点的意见和数据，所以必须设计一套制度来维护系统的运作顺序和公平性、维护系统各节点的数据一致性。区块链就是将一系列包含信息的区块以链的形式链接在一起，这些包含信息的区块存在于整个分布式网络的所有节点中，区块的生产和确认是通过共识机制[4]来实现的。随着加密货币的兴起，不断推动新的共识机制被提出，如 PoW[5]、PoS[6]、DPoS[7]、DAG[8]等。

按照区块链开放程度和应用场景进行划分主要分为私有链、公有链和联盟链[9]三大类。私有链是指权限仅在一个组织或机构范围内的区块链，一般用于某个中心化机构。私链的共识算法运用的是传统分布式系统里的共识算法，主要代表的共识算法有 Paxos[10]、Raft[11]等，这类算法不会考虑拜占庭容错问题[12,13]，一般只考虑因为节点自身以及网络原因导致的故障（如节点宕机、网络故障等因素）而不考虑集群中会有恶意节点的情况；公有链在这三种类别的链中属于去中心化程度最高的链，耳熟能详的公有链包括比特币、以太坊[14]等，公有链允许每个参与者查看链上的信息，公有链最著名的共识算法为工作量证明（proof of work，PoW），而 PoW 这种共识算法有浪费能源、性能低下的缺点；联盟链指的是由一定数量的组织和机构通过联盟的方式构建的一条链，仅对特定的组织和机构开放，最著名的项目由多家国际银行和金融机构组成的区块链联盟 R3 和 IBM 的超级账本（HyperLedger[15]），联盟链最常用的

收稿日期：2020-12-07；修回日期：2021-01-11 基金项目：国家自然科学基金资助项目（62076073，91748107）；广东省基础与应用基础研究基金资助项目（2020A1515010616）；广东省引进创新科研团队计划资助项目（2014ZT05G157）

作者简介：陈佳伟（1996-），男，广东汕头人，硕士研究生，主要研究方向为区块链技术；冼祥斌（1996-），男，广东湛江人，硕士研究生，主要研究方向为区块链技术；杨振国（1988-），男，山东潍坊人，副教授，博士，主要研究方向为跨模态检索、区块链技术；刘文印（1966-），男（通信作者），吉林长春人，教授，博导，博士，主要研究方向为区块链技术、网络与信息安全（liuwy@gdut.edu.cn）。

共识算法为 PBFT 共识算法[16]，不同于 Paxos、Raft 算法，PBFT 算法会考虑分布式系统中有恶意节点的情况，即使某条联盟链由多个信用和名气极高的组织或机构组成，也不能排除节点作恶的情况。PBFT 算法主要为了解决拜占庭将军问题，假设网络中作恶节点数量最多为 f，通过数学理论证明了网络中节点数量 $n>3f$ 的情况下，共识能在分布式网络中达成，即 $n\geqslant 3f+1$ 的情况下达成共识，PBFT 算法能容忍不超过 1/3 的拜占庭节点。PBFT 算法为了达成数据的一致性采用了多阶段交互的方式，而在一轮共识的过程整个网络达到 $O(n^2)$ 的通信复杂度，此处 n 为参与共识的节点数量。因此实用拜占庭容错协议的缺点在于不适用于大规模的节点共识，因为随着节点规模的增大，达成共识需要的时间大大增加，不符合效率需求，整个网络的通信复杂度会随着节点的增加而呈平方级增大，可扩展性较差。针对该问题，SBFT[17] 利用收集器来实现线性通信模型，SBFT 将收集器分为提交收集器和执行收集器，用两种不同类型的收集器在达成共识的不同阶段中收集各个副本节点的消息及签名；方维维等人[18] 就这个问题也提出了优化一致性协议，通过主节点收集和验证各个副本节点的反馈信息，并比对各个信息的核心内容是否一致，在不存在拜占庭节点的情况下能够实现高效并快速地达成共识。但该方法也有一定的缺点，在存在拜占庭节点的情况下，该共识流程将会被切换，需要执行原 PBFT 算法来达成共识以保证系统各节点的数据一致性，极大地增加了达成共识的时延。

本文基于 PBFT 共识算法，提出了一种改进方案 ABFT 算法，其优势在于：a) 结合 BLS 签名可聚合多个对同一消息的签名为一个签名的优点，在 prepare 阶段将原本各节点的广播更换为由各节点单点发送 prepare 消息给该轮的主节点，在 commit 阶段由主节点收集各个节点签名，收到 $2f+1$ 个对区块的签名后将验证过的这批签名聚合成一个聚合签名，再发送该聚合签名以及参与签名的节点列表给各个参与共识的节点，节点会根据发送过来的参与签名的节点列表获得参与签名节点的公钥组对该聚合签名进行验证，通过改进这两个阶段将共识过程整个网络 $O(n^2)$ 的通信复杂度降低到 $O(n)$；b) 增加了 finish 阶段，finish 阶段的目的是若各节点对最终的区块及数据验证无误则通知下一轮出块的主节点，防止下一个出块的主节点提前进入 pre-prepare 阶段而其他节点由于网络延迟等问题没有进入下一轮共识过程，导致最终视图切换；c) 由于改进之后的机制比较依赖主节点的响应，改进了视图变更协议及流程，防止主节点作恶的情况。

1　预备知识

1.1　PBFT 算法

PBFT 算法也称为实用拜占庭容错算法，PBFT 算法在 1999 年提出，用于解决拜占庭将军问题[13,19]。拜占庭将军问题主要描述的是如何在有叛军的情况下传递消息使军队达成一致的行为。要解决这个问题的前提是通信必须是可靠的，如果通信不可靠则问题无解。而拜占庭将军问题中通过通信不可靠而试图达成一致的结果几乎是不可能的或者非常困难，所以要在通信可靠的前提下来解决此问题，也就是在系统上有一些恶意组件不断发送错误信息的情况下让系统依旧正常运行的能力。一致性问题在分布式领域既是最基础又是最重要的问题，共识算法解决的是如何在分布式系统中达成一致性的问题[20]。

PBFT 算法将参与共识的节点分为主节点和副本节点。每一轮共识有且只有一个主节点，主要负责打包消息、发起共识流程，为了保证公平性，每轮共识后均会切换 leader；副本节点负责参与共识，副本节点不唯一，每轮共识过程中有多个副本节点。

PBFT 算法主要分为以下三个阶段：

a) pre-prepare 阶段。主节点（节点 1）收到客户端发来的请求，验证请求的信息，封装 pre-prepare 消息〈〈pre-prepare，v，n，d〉，s，m〉广播给其余的副本节点，其中，v 为视图编号，n 为序列号，m 为消息，d 为 m 的摘要，s 是主节点对 m 的签名。

b) prepare 阶段。当其他副本节点收到主节点发送的 pre-prepare 消息后，会根据主节点的公钥去验证签名的有效性以及摘要的正确性，并对其他的一些信息进行验证，如果消息验证都通过，会构造 prepare 消息〈prepare，v，n，d，i，s〉，其中，i 为节点编号，s 为该副本节点对区块摘要的签名信息。在 prepare 阶段，节点还会收集其他节点发送过来的 prepare 消息，通过比对信息的异同验证收到的 prepare 消息的准确性。如果收到超过 $2f+1$ 个（包括自己）prepare 消息并验证通过，就代表 pre-pare 阶段已经完成，进入 commit 阶段。

c) commit 阶段。网络中所有节点生成 commit 消息〈commit，v，n，d，i〉并广播，同理，如果收到超过 $2f+1$ 个（包括自己）commit 消息并验证通过，则代表绝大多数节点已经进入 commit 阶段，这一阶段已达成共识。

PBFT 算法共识过程如图 1 所示。

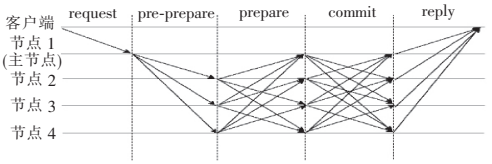

图 1　PBFT 算法共识过程

Fig. 1　Consensus process of PBFT algorithm

1.2　BLS 签名算法

BLS 签名方案最初在 2011 年提出，是一种可以实现集合多个签名的算法，可以将区块中的所有签名聚合成一个，容易实现 m-n 多重签名，也可以避免签名者之间的多余通信。通过聚合签名，主节点可以聚合其他节点的签名，而副本节点收到主节点包含聚合签名的信息后，可以一次性地对合并后的签名再次进行验证。运用 BLS 签名算法可以很好地解决 PBFT 算法通信复杂度太高的问题。BLS 签名算法主要依赖于双线性映射函数。定义配对函数 $e(P,Q)$，P 和 Q 为一条曲线（或两条不同的曲线）的两个点，配对函数对曲线上的运算满足分配律、交换律和结合律。假设要签署的消息为 m，G 为曲线上的一个生成点，私钥为 pk，则公钥为 $P=pk*G$。在 BLS 签名算法中，需要用到新型散列函数，不同于一般散列函数得到的结果是数值，新型散列函数的结果是椭圆曲线上的一个点。为了得到签名，首先将消息 m 映射为曲线上的一个点 H，签名 $S=pk*H$。验证签名可以通过验证 $e(P,H)$ 是否等于 $e(G,S)$，由于配对函数满足分配律、交换律和结合律，所以证明如下：

$$e(G,S)=e(G,pk*H)=e(pk*G,H)=e(P,H) \tag{1}$$

本文的 ABFT 算法采用主节点收集各个副本节点发送的签名，再对验证通过的签名进行聚合，聚合后的签名大小与单个签名大小一致，仅需 33 Byte，非常节约空间。对于签名的聚合操作，假设要聚合 100 个节点的签名，以 P_i 代表第 i 个节点的公钥，以 S_i 代表第 i 个节点的签名，每个签名都是不同节点的密钥对对同一个消息的签名。根据运算，聚合签名只是所有签名的总和，用 S 代表聚合后的聚合签名 $S=S_1+S_2+\cdots+S_{100}$，根据上面的证明，可以通过验证 $e(G,S)$ 是否等于 $e(P_1,H)*e(P_2,H)*\cdots*e(P_{100},H)$ 对聚合签名进行验证，根据配对函数的特征，则对聚合签名的验证如下：

$$\begin{aligned}e(G,S)&=e(G,S_1+S_2+\cdots+S_{100})=e(G,S_1)*e(G,S_2)*\cdots*e(G,S_{100})=\\&e(G,pk_1*H)*e(G,pk_2*H)*\cdots*e(G,pk_{100}*H)=\\&e(P_1,H)*e(P_2,H)*\cdots*e(P_{100},H)\end{aligned} \tag{2}$$

2　ABFT 算法方案

2.1　ABFT 算法共识流程

当 PBFT 算法运用到区块链中，由于区块链点对点的网络拓

扑结构的特点，无须再使用 C/S 响应模型。针对 PBFT 算法通信复杂度达到 $O(n^2)$ 的问题，本文通过改进共识阶段的 prepare 和 commit 阶段，使其通信复杂度减少到 $O(n)$，使其更加适合于多节点参与的联盟链中。ABFT 算法的核心流程如图 2 所示。

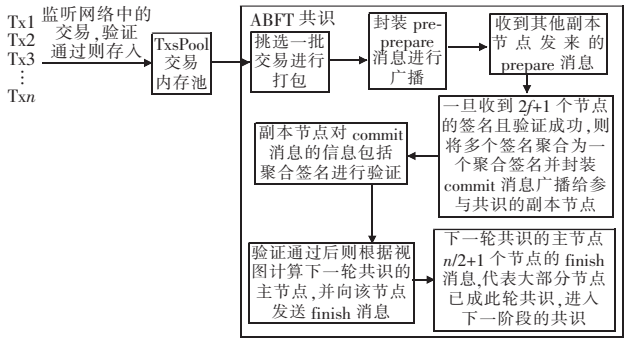

图 2 ABFT 算法流程

Fig. 2 Flow chart of ABFT algorithm

a) pre-prepare 阶段。主节点的主要职责是收集交易内存池中的交易，将交易打包成区块并广播给各个副本节点共识。主节点广播一条 $\langle pre\text{-}prepare, h, v, d, block, s\rangle$ 消息给其他节点，其中：*pre-prepare* 代表这是一条 *pre-prepare* 消息；h 代表区块高度；v 代表视图编号；d 代表 *block* 的摘要，即区块哈希值，*block* 则为整个区块的内容，包括了所收集的交易及所有交易的签名。

b) prepare 阶段。所有的副本节点收到主节点发送过来的 pre-prepare 消息后，首先会对该消息进行验证，检查摘要、区块高度、视图以及签名的合法性，验证完毕后会对其中的所有交易进行验证，验证无误后在准备阶段发送一条 $\langle prepare, h, v, d, i\rangle$ 给主节点。与原 PBFT 算法略有不同的是，这里节点的发送方式不再是广播，而是只将签名后的信息发送给主节点，通过这种方式减少通信的消息量。

c) commit 阶段。主节点在此阶段一直会监听各个副本节点发送过来的 prepare 信息，主节点会对每个 prepare 信息进行验证，验证通过则收集起来。一旦主节点收集到 $2f+1$ 个（包括自己）签名，验证这批签名并将验证通过的签名通过 BLS 签名聚合成一个签名，广播一条 $\langle commit, h, v, d, aggrsignature, node\rangle$ 消息，这里的 *aggrsignature* 就是聚合之后合成的签名，*node* 指的是参与该聚合签名的所有节点的 ID 列表，方便后面收到该消息的节点能利用参与签名的节点的公钥验证该签名是否正确，此时各节点收到主节点的聚合签名后，验证无误则将该区块链接到区块链的链尾完成同步。commit 阶段的主要作用是由主节点收集签名并验证，结合 BLS 签名将验证通过的多个签名聚合成一个聚合签名，并将该聚合签名以及其他必要信息广播给其他所有副本节点验证，实现在不需要全广播通信的情况下各个副本节点能够根据参与签名的节点的公钥验证该聚合签名的真实性。

d) finish 阶段。在以上三阶段的基础上加入了 finish 阶段，该阶段的主要目的是进行最终确认，也是为了将信息发送给下一轮的主节点，防止下一轮的主节点预先进入下一轮的 pre-prepare 阶段而其他的副本节点却未就绪的情况。在此阶段会有一个超时时间 t，如果在 t 之前下一轮主节点已经收到多于一半的 finish 消息，证明大多数节点已经就绪，可以提前进入下一个区块的共识过程。这一阶段 t 统一设置为 200 ms，因为这一阶段只是对各个副本节点的共识完成情况进行确认的一个过程，所以 t 也不必过大。而在超时时间 t 内没有收到多于一半节点的消息，则会重发已完成的共识结果给其余的副本节点，再进行下一轮的共识流程。

在这四阶段，通信的消息总量为 $n-1+n-1+n-1+n-2=4n-5$ 个，而原先三阶段的消息总量为 $n-1+(n-1)^2+n(n-1)=2n^2-2n$ 个，使网络的通信量从节点数的平方级 $O(n^2)$ 低到线性 $O(n)$，大大减少了网络的通信量，使得拜占庭容错算能够应用于实际应用中。ABFT 算法共识过程如图 3 所示。

2.2 ABFT 算法视图切换

由于 ABFT 算法比较依赖主节点，为了防止主节点权力太大导致作恶、故意不应答的情况，必须作出一些改进。本文改进了视图切换协议，视图切换协议主要用于两种情况：a) 主节点发来的区块信息有误（包括伪造交易或者签名），诚实的节点会根据账号的公钥去验证消息的正确性，根据本文所改进的四阶段共识协议，若要对恶意消息达成一致性，至少需要超过 $n/3$ 的恶意节点，所以归根到底更重要的是考虑主节点故意不应答的情况；b) 主节点由于宕机或作恶，没有在规定时间内发起本轮共识操作或者在 commit 阶段超时没有回应，使本轮无法完成共识。

针对情况 b)，若主节点在规定时间内没有作出指定的操作和回应，则需要更换主节点使整个网络对区块的共识流程正常进行，此时主节点有可能使恶意的节点，故意破坏整个共识网络，也有可能仅仅是节点宕机了。主节点异常时视图切换流程如图 4 所示。

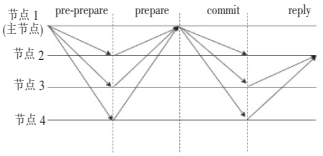

图 3 ABFT 算法共识过程

Fig. 3 Consensus process of ABFT algorithm

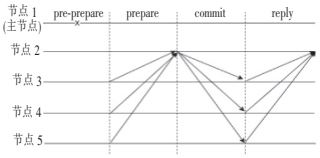

图 4 主节点异常时视图切换流程

Fig. 4 View switching process when the master node is abnormal

a) view-change 阶段。在视图 v 下，各个副本节点在指定时间 t 内没有收到主节点的响应，则认为主节点发生故障，发起视图变换请求，将旧视图的编号自增 $v=v+1$，根据视图编号选择下一轮共识新的主节点，封装 view-change 消息以及节点编号、消息签名 $\langle\langle view\text{-}change, v, h\rangle, i, s\rangle$ 发送给下一轮共识的主节点，h 为当前进行的区块高度，i 为节点编号，s 为该副本节点对 *view-change* 消息的签名。

b) new-view 阶段。此时下一轮共识的主节点会不断监听各个副本节点发来的 view-change 消息对 view-change 消息进行验证，验证各个副本节点发来的消息中的视图以及高度是否一致。收集到 $2f+1$ 个（包括自己）消息后，验证这批消息的签名和信息的正确性，统一高度和视图，通过 BLS 聚合签名将验证通过后的签名聚合成一个聚合签名，封装 new-view 消息 $\langle new\text{-}view, v, h, aggrsignature, node\rangle$ 广播给各个副本节点，这里的 *aggrsignature* 同样是聚合后合成的签名，*node* 指的是参与该聚合签名的所有节点的 ID 列表，方便各个副本节点验证。

c) new-view-ack 阶段。各个副本节点收到主节点发来的 new-view 消息后对聚合签名进行验证，验证完毕后对下一轮的主节点发送一个 new-view-ack 消息表示验证通过，而下一轮的主节点开始打包区块，开始新一轮的区块共识。

3 ABFT 算法实验方案

为了验证本文改进后的 ABFT 算法的效率，基于 Golang 语言实现了 ABFT 算法，采用多机器多节点模拟共识进行。本次实验在一台 Intel Xeon（Cascade Lake） Platinum 8269CY @ 2.5 GHz/3.2 GHz、32vCPU、32 GB 内存、操作系统为 CentOS 7.6 64 位的云服务器上进行。账号体系用与比特币一致的 secp256k1 签名方案，而区块链节点间的公私钥和签名采用的是 BLS 签名方案。实验中每个区块设置 1 000 笔交易，在该系统中对 PBFT 和 SBFT 以及本文提出的 ABFT 算法进行了验证，从共识时延、吞吐量等方面对 PBFT、SBFT 和 ABFT 算法进行分析；另外还从聚合签名时延和验证聚合签名时延方面进行

实验，证实结合 BLS 签名并不会带来额外的开销，且有利于维持较高的吞吐量以及从一定程度上减少共识时延。

3.1　聚合签名时延和验证聚合签名时延

由于改进后的 ABFT 算法利用了 BLS 聚合签名，需要详细分析聚合签名所花费的时间以及验证聚合签名所花费的时间，了解其是否会影响整个共识过程的性能。这里的聚合签名时延和验证聚合签名时延分别指的是从聚合开始到聚合成一个聚合签名所花费的时间、从验证聚合签名开始到验证完毕所花费的时间。实验使用 Golang 的 Benchmark 对聚合签名函数和验证聚合签名函数进行性能基准测试和压力测试，发现聚合签名和验证聚合签名花费的时间并不多，且耗时随着节点的增多而线性增加，并不会指数级增加，非常适用于联盟链多节点的共识中。

对于聚合签名，模拟了 n 个签名进行聚合，对于每一个签名使用的是不同的公私钥进行签署。分别模拟了 50、100、150、200、250、300、350、400、450、500 个签名的聚合，对聚合签名函数进行了性能测试，结果如图 5 所示。可以从图 5 发现，聚合签名的时延非常短，仅随着节点的增大而线性缓慢增加，聚合 500 个签名仅需花费 0.000 1 s，所以这一部分基本不会耗费太多的时间和性能。

对于验证聚合签名，本文同样通过 Golang 的 Benchmark 进行压力测试，通过先聚合 50、100、150、200、250、300、350、400、450、500 个签名，再用这 50、100、150、200、250、300、350、400、450、500 个签名所对应的公钥组对聚合过后的签名进行验证，从而得知验证聚合签名的开销，结果如图 6 所示。

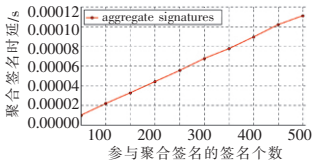

图 5　聚合 n 个签名所花费的时间

Fig. 5　Spent time of aggregating n signatures

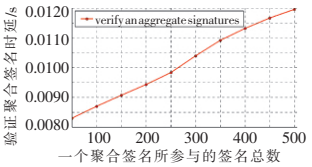

图 6　验证由 n 个签名聚合而成的聚合签名所花费的时间

Fig. 6　Spent time of verifying an aggregate signature aggregated by n signatures

从图 6 可以发现验证由 n 个签名聚合而成的聚合签名所花的时间比聚合 n 个签名所花的时间要多，但仍不足以影响到整个网络的性能，验证一个由 500 个签名聚合而成的聚合签名仅需要 0.012 s，故聚合签名非常适用于多节点的共识中。通过实验可以了解到结合聚合签名去改进 PBFT 共识算法不仅不会造成整个网络耗时的大幅增加，而且有效地改进了 PBFT 算法全广播通信开销大的问题，减少了通信的次数以及开销，有效提高了共识的效率，解决了 PBFT 算法在节点增加的情况下性能快速下降的问题，有利于提升整个网络的可扩展性。

3.2　共识时延

共识时延是指主节点打包完区块并发起网络共识到确认区块完成共识的时间间隔。实验的设计是每个区块打包 1 000 笔交易，分别测试在 5、10、15、20、25、30、35、40 个节点参与共识的情况下每个区块平均完成的时间，通过多次测试求平均值，结果如图 7 所示。根据图 7 可以发现，随着参与网络共识节点的增多，ABFT 算法明显优于 PBFT 和 SBFT 算法，完成一个区块的共识所需的时间并不会随着节点的增多而急剧增大，相对来说 ABFT 算法共识时延比较稳定，而 PBFT 算法共识时延随着节点的增多而急剧增大，SBFT 算法效率跟 ABFT 算法相比还有差距。PBFT 算法通过多阶段的全广播方式达成共识，完成最终一致性确认过程，在参与共识的节点数量不断增多的情况下，通信开销会不断增大，因此共识时延较高。SBFT 算法改进了 PBFT 算法全广播的方式，利用收集器实现线性通信模型，将收集器分为提交收集器和执行收集器，用两种不同类型的收集器在达成共识的不同阶段中收集各个副本节点的消息及签名，一定程度上降低了共识的时延，但仍可改进；ABFT 算法采用主节点单点收集和验证各个副本节点的签名，利用 BLS 聚合签名对各个副本节点的签名进行聚合，并将该聚合签名以及其他必要信息广播给其他所有副本节点验证，优化了达成共识的过程，减少了通信的次数以及开销，共识时延大大降低。因此 ABFT 算法的可扩展性明显优于 PBFT 以及 SBFT 算法，更适用于多节点的联盟链中。

3.3　吞吐量

交易吞吐量是指一个单位时间内系统处理交易的数量，用于衡量系统处理交易的能力。区块链常用每秒交易数(TPS)来表示，该指标也经常用于检测系统的性能。

由图 8 可以看出，ABFT、SBFT 和 PBFT 算法的吞吐量都是随着参与网络共识节点的增多而下降，而 ABFT 算法下降趋势变缓，曲线会逐渐走平，而且交易吞吐量依旧维持在较高的水平，参与网络共识节点数量的增加对其影响不会很大。PBFT 算法的吞吐量随着参与网络共识节点数量的增加快速下滑，而 SBFT 算法在吞吐量上跟 ABFT 算法相比还是有所差距，从图中可以看出 ABFT 算法的性能明显优于 SBFT 和 PBFT 算法。

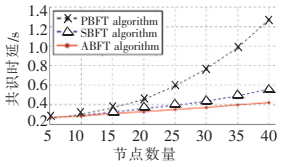

图 7　ABFT、SBFT 和 PBFT 算法共识时延对比

Fig. 7　Comparison of consensus delay between ABFT, SBFT and PBFT algorithms

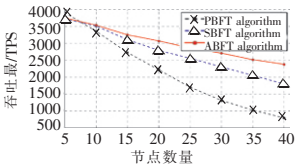

图 8　ABFT 和 SBFT、PBFT 算法交易吞吐量对比

Fig. 8　Comparison of transaction throughput between ABFT, SBFT and PBFT algorithms

4　结束语

近年来区块链得到了迅猛发展，展示了其潜力以及未来广泛的应用前景，若想支持大规模的区块链应用离不开联盟链的发展，而其发展的核心在于共识机制的发展与改进。改进后的 ABFT 算法将原本 PBFT 算法的通信复杂度从 $O(n^2)$ 降至 $O(n)$，极大地减少了网络通信量，提升了通信效率，不再仅仅支持少数节点的共识，也可以适用于更多节点的共识场景中，有利于提高联盟链的可扩展性。通过结合聚合签名，使原本需要多阶段广播的共识改进为通过主节点聚合各个节点的签名并能将此签名给各个节点验证。经过实验验证，该改进大大减少了共识时延，随着参与共识节点的增多，ABFT 算法明显优于 PBFT 以及 SBFT 算法。而且通过压力测试，证明聚合签名和验证聚合签名几乎不会造成大的性能影响，耗费的时间很少，而且耗费的时间随着节点的增多仅仅是线性增加，可扩展性也比较强。在未来的工作中，增加动态加入和退出节点的功能对 ABFT 算法进一步的优化，使其更能适用于联盟链。

参考文献：

[1] Di Pierro M. What is the blockchain? [J]. Computing in Science & Engineering, 2017, 19(5): 92-95.

[2] Nakamoto S. Bitcoin: a peer-to-peer electronic cash system [EB/OL]. (2009-03-24). https://bitcoin.org/bitcoin.pdf.

[3] 袁勇，王飞跃. 区块链技术发展现状与展望 [J]. 自动化学报，2016，42(4)：481-494. (Yuan Yong, Wang Feiyue. Blockchain: the state of the art and future trends [J]. Acta Automatica Sinica, 2016, 42(4): 481-494.)

[4] 刘懿中，刘建伟，喻辉. 区块链共识机制研究：典型方案对比 [J]. 中兴通讯技术，2018，24(6)：2-7. (Liu Yizhong, Liu Jianwei, Yu Hui. Research on blockchain consensus: comparison of typical schemes [J]. ZTE Technology Journal, 2018, 24(6): 2-7.)

(下转第 1962 页)

[16] 李敏,孟祥茂. 动态蛋白质网络的构建、分析及应用研究进展[J]. 计算机研究与发展,2017,54(6):1281-1299. (Li Min,Meng Xiangmao. The construction,analysis,and applications of dynamic protein-protein interaction networks [J]. Journal of Computer Research and Development,2017,54(6):1281-1299.)

[17] 王昊奋,丁军,胡芳槐,等. 大规模企业级知识图谱实践综述[J]. 计算机工程,2020,46(7):1-13. (Wang Haofen,Ding Jun,Hu Fanghuai,*et al*. Survey on large scale enterprise-level knowledge graph practices [J]. Computer Engineering,2020,46(7):1-13.)

[18] Khalil E,Dai Hanjun,Zhang Yuyu,*et al*. Learning combinatorial optimization algorithms over graphs [C] //Advances in Neural Information Processing Systems. 2017:6348-6358.

[19] Bruna J,Zaremba W,Szlam A,*et al*. Spectral networks and locally connected networks on graphs [EB/OL]. (2014-05-21). https://arxiv.org/abs/1312.6203.

[20] Defferrard M,Bresson X,Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering [C] //Advances in Neural Information Processing Systems. 2016:3844-3852.

[21] Duvenaud D K,Maclaurin D,Iparraguirre J,*et al*. Convolutional networks on graphs for learning molecular fingerprints [C] //Advances in Neural Information Processing Systems. 2015:2224-2232.

[22] Li Yujia,Tarlow D,Brockschmidt M,*et al*. Gated graph sequence neural networks [EB/OL]. (2017-09-22). https://arxiv.org/abs/1511.05493.

[23] Veličković P,Cucurull G,Casanova A,*et al*. Graph attention networks [EB/OL]. (2017-10-30). https://arxiv.org/abs/1710.10903.

[24] Gilmer J,Schoenholz S S,Riley P F,*et al*. Neural message passing for quantum chemistry [EB/OL]. (2017-06-12). https://arxiv.org/abs/1704.01212.

[25] Tang Yuhang,Zhang Dongkun,Karniadakis G E. An atomistic fingerprint algorithm for learning ab initio molecular force fields [J]. The Journal of Chemical Physics,2018,148(3):034101.

[26] Klicpera J,Weiβenberger S,Günnemann S. Diffusion improves graph learning [C] //Advances in Neural Information Processing Systems. 2019:13354-13366.

[27] Klicpera J,Bojchevski A,Günnemann S. Predict then propagate:graph neural networks meet personalized PageRank [EB/OL]. (2019-02-27). https://arxiv.org/abs/1810.05997.

[28] Kondor R I,Lafferty J. Diffusion kernels on graphs and other discrete structures [C] //Proc of the 19th International Conference on Machine Learning. 2002:315-322.

[29] Wu Shu,Tang Yuyuan,Zhu Yanqiao,*et al*. Session-based recommendation with graph neural networks [J]. Proceedings of the AAAI Conference on Artificial Intelligence,2019,33:346-353.

[30] Tan Y K,Xu Xingxin,Liu Yong. Improved recurrent neural networks for session-based recommendations [C] //Proc of the 1st Workshop on Deep Learning for Recommender Systems. 2016:17-22.

[31] Sarwar B,Karypis G,Konstan J,*et al*. Item-based collaborative filtering recommendation algorithms [C] //Proc of the 10th International Conference on World Wide Web. 2001:285-295.

[32] Xu Chengfeng,Zhao Pengpeng,Liu Yanchi,*et al*. Graph contextualized self-attention network for session-based recommendation [C] //Proc of the 28th International Joint Conference on Artificial Intelligence. 2019:3940-3946.

[33] Chen Tianwen,Wong R C W. Handling information loss of graph neural networks for session-based recommendation [C] //Proc of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020:1172-1180.

[34] Paszke A,Gross S,Chintala S,*et al*. Automatic differentiation in PyTorch [EB/OL]. (2017-10-28). https://openreview.net/forum?id=BJJsrmfCZ.

[35] Fey M,Lenssen J E. Fast graph representation learning with PyTorch geometric [EB/OL]. (2019-04-25). https://arxiv.org/abs/1903.02428.

[36] Abu-El-Haija S,Perozzi B,Al-Rfou R,*et al*. Watch your step:learning node embeddings via graph attention [C] //Advances in Neural Information Processing Systems. 2018:9180-9190.

[37] Grover A,Leskovec J. node2vec:scalable feature learning for networks [C] //Proc of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016:855-864.

(上接第 1955 页)

[5] Gervais A,Karame G O,Wüst K,*et al*. On the security and performance of proof of work blockchains [C] //Proc of ACM SIGSAC Conference on Computer & Communications Security. New York:ACM Press,2016:3-16.

[6] Saleh F. Blockchain without waste:proof-of-stake [J]. Review of Financial Studies,2021,34(3):1156-1190.

[7] 武岳,李军祥. 区块链共识算法演进过程[J]. 计算机应用研究,2020,37(7):2097-2103. (Wu Yue,Li Junxiang. Evolution process of blockchain consensus algorithm [J]. Application Research of Computers,2020,37(7):2097-2103.)

[8] 高政风,郑继来,汤舒扬,等. 基于 DAG 的分布式账本共识机制研究[J]. 软件学报,2020,31(4):1124-1142. (Gao Zhengfeng,Zheng Jilai,Tang Shuyang,*et al*. State-of-the-art survey of consensus mechanisms on DAG-based distributed ledger [J]. Journal of Software,2020,31(4):1124-1142.)

[9] 黄步添,蔡亮. 区块链解密:构建基于信用的下一代互联网[M]. 北京:清华大学出版社,2016:43. (Huang Butian,Cai Liang. Blockchain decryption:building the next generation of Internet based on credit [M]. Beijing:Tsinghua University Press,2016:43.)

[10] Lamport L,Massa M. Cheap paxos [C] //Proc of International Conference on Dependable Systems & Networks. Washington DC:IEEE Computer Society,2004.

[11] Ongaro D,Ousterhout J. In search of an understandable consensus algorithm [C] //Proc of USENIX Annual Technical Conference. Berkeley,CA:USENIX Association,2014:305-319.

[12] Lamport L,Shostak R,Pease M. The Byzantine generals problem [J]. ACM Trans on Programming Languages and Systems,1982,4(3):382-401.

[13] Pease M,Shostak R,Lamport L. Reaching agreement in the presence of faults [J]. Journal of the ACM,1980,2(4):228-234.

[14] Wood G. Ethereum:a secure decentralised generalised transaction ledger [J]. Ethereum Project Yellow Paper,2014,151:1-32.

[15] Androulaki E,Barger A,Bortnikov V,*et al*. HyperLedger Fabric:a distributed operating system for permissioned blockchains [C] //Proc of the 13th EuroSys Conference. New York:ACM Press,2018:1-15.

[16] Lamport L,Shostak R,Pease M. Practical Byzantine fault tolerance and proactive recovery [J]. ACM Trans on Computer Systems,2002,20(4):398-461.

[17] Gueta G G,Abraham I,Grossman S,*et al*. SBFT:a scalable decentralized trust infrastructure for blockchains [EB/OL]. (2018-04-04). https://arxiv.org/pdf/1804.01626v1.pdf.

[18] 方维维,王子岳,宋慧丽,等. 一种面向区块链的优化 PBFT 共识算法[J]. 北京交通大学学报,2019,43(5):58-64. (Fang Weiwei,Wang Ziyue,Song Huili,*et al*. An optimized PBFT consensus algorithm for blockchain [J]. Journal of Beijing Jiaotong University,2019,43(5):58-64.).

[19] Lamport L,Shostak R,Pease M. The Byzantine generals problem [J]. ACM Trans on Programming Languages and Systems,1982,4(3):382-401.

[20] Fischer M J. The consensus problem in unreliable distributed systems (a brief survey) [C] //Proc of International Conference on Fundamentals of Computation Theory. Berlin:Springer,1983:127-140.