

基于时间排序的监督共识算法

牟平, 梁鉴如

(上海工程技术大学, 上海 201620)

摘要:区块链作为一种新型的去中心化数据处理协议,其信息可追溯,不可篡改,灾备性能优异,不依赖特定机构背书的特点引起了国内外研究者的广泛关注。受制于网络的物理限制以及软件支持,为了获得较高的性能,传统的区块链算法往往依靠主节点对客户端的消息进行排序、打包和传播,如果主节点被恶意控制则会对系统造成极大危害。针对系统过度依赖主节点的问题,对主流的区块链主节点选举方式以及在系统中的作用进行了深入分析和讨论。从安全性、稳定性、吞吐率等方面详细阐述了不同算法的优点和弊端。在此基础上提出一种利用监督节点进行比较验证主节点是否诚实工作的方法。分析证明,优化后的共识能以较低的代价显著增加主节点作弊的成本,有效提升了系统的安全性。

关键词:区块链;监督;时间戳;主节点;排序

中图分类号:TP301

文献标识码:A

文章编号:1673-629X(2021)08-0087-05

doi:10.3969/j.issn.1673-629X.2021.08.015

Supervised Consensus Algorithm Based on Time Sorting

MU Ping, LIANG Jian-ru

(Shanghai University of Engineering Science, Shanghai 201620, China)

Abstract:Blockchain, as a new type of decentralized data processing protocol, has attracted extensive attention from researchers at home and abroad for its characteristics of traceability of information, non-tamper-proof, excellent disaster recovery performance and independent endorsement of specific institutions. Constrained by the physical limitations of the network and software support, in order to obtain higher performance, traditional blockchain algorithms often rely on the master node to sort, package and spread the messages of the client. If the master node is maliciously controlled, it will cause extreme damage to the system great harm. Aiming at the problem of the system's excessive dependence on the master node, the mainstream blockchain master node election method and its role in the system are analyzed and discussed in depth. The advantages and disadvantages of different algorithms are explained in detail from the aspects of security, stability, throughput rate, etc. On this basis, a method of comparing and verifying whether the master node is working honestly by using the supervision node is proposed. The analysis proves that the optimized consensus can significantly increase the cost of cheating by the master node at a lower cost, and effectively improve the security of the system.

Key words: blockchain; supervision; timestamp; master node; sort

0 引言

Satoshi Nakamoto 在 2008 年提出了 bitcoin^[1], 描述了一种全新的电子货币及其算法。它的不可篡改、多方维护的特性,引起了各个领域的广泛关注,其重要底层技术之一的区块链也吸引了诸多研究者关注。

在传统的数据系统中,账本上的一系列信息往往由一个中心服务器进行记录,这种设计特性,节省了使用成本,保证了数据的强一致性、规避了分区容错性但却使得可用性风险极高^[2]。在数据储存阶段,采用多备份容灾,足以抵御自然条件下的偶发灾害。相较而

言,恶意的节点对传统分布式数据系统造成的问题要严重得多,典型的有 DDOS 攻击,冒充客户端提交虚假信息,攻击中心服务器篡改用户提交的信息,篡改历史信息等^[3-4]。这对新型的数据系统分布式系统带来了以下挑战:在一个无中心,弱信任的分布式系统中,如何让各个节点在指定时间内达成共识^[5]。

所谓共识,简单来说就是分布式系统中节点对某个数值或状态的承认^[6]。Lynch 基于如今的计算机技术和网络技术的结构提出了 CAP 理论^[7],认为分布式系统不可能同时满足一致性(consistency)、可用性(a-

收稿日期:2020-10-10

修回日期:2021-02-11

基金项目:国家自然科学基金(61705127)

作者简介:牟平(1995-),男,硕士研究生,研究方向为区块链、分布式数据库;梁鉴如,高级实验师,硕士,全国研究生电子设计竞赛上海赛区专家组专家,研究方向为信号检测及控制。

availability) 和分区容错性(partition tolerance)这三个基本需求,最多只能同时满足其中两项。业界在分布式的实际使用中,对 CAP 特性进行进一步权衡,由 Dan Pritchett 总结为主要特征为基本可用(basically available)、软状态(soft state)和最终一致性(eventually consistent)的 BASE 原理。Gray 在 1978 提出了 2PC (concurrency control and recovery in database systems)通过两阶段进行事务提交,但如果从节点失效则会发生事务阻塞,使系统崩溃。为增强系统健壮性,Skeen 在 1981 年提出(3PC)通过三阶段进行事务提交,降低了数据不一致的概率。Lamport 在 1998 年提出的 Paxos^[8]是第一个获得广泛认可的基于消息传递的一致性算法。Diego Ongaro 和 John Ousterhout 通过问题拆分、状态空间降维等方式对 Paxos 进行简化,提出了 Raft 算法。Barbara Liskov 等在 1999 年提出了实用拜占庭容错算法^[9],解决了原始拜占庭不能容错的问题,并且将算法的复杂度由指数级降低到多项式级。扩展了系统的可用性。Ramakrishna Kotla 在 2007 年提出了基于 zyzzyva 协议^[10]的 SBFT 算法^[11],使得消息事件复杂度由 PBFT 的 $O(n^2)$ 降为 $O(n)$,改良后效果非常显著,极大地提高了共识效率。但这些算法为了保证数据的一致性,一般选择对客户端发送的消息由一个主节点进行排序、打包,再分发到各个副本节点进行校验,但是主节点的更换方式采取的是轮巡式,敌手非常容易猜到预备的主节点,这可能导致敌手有针对性地对攻击预备节点,或者故意瘫痪当前节点,迫使切换到已控制的节点中完成攻击。因此,应当避免系统通过单一主节点接受输入消息,防止主节点被劫持给系统带来风险。同时应当减少同步数据中的计算开支。文中在 SBFT 共识算法的基础上引入了计入时间戳排序的监督共识算法:

(1)引入多节点对客户端消息排序打包,防止单一主节点篡改、瞒报客户端的消息。

(2)引入消息的时间水位和延时排队概念,将时间戳作为消息排序的依据。设置消息缓冲,解决网络延迟带来的各节点消息不同步。

(3)增加节点间消息对比方式,对于不同的业务逻辑使用不同的对比逻辑,增加使用弹性。

1 主节点的选择方法

1.1 PoW 中主节点的选择方法

比特币的 PoW 共识算法^[1]中每个节点会将收到的交易数据(记为 m)、前一个区块的 merkle 值、时间戳(记为 t)以及随机数(nonce)等结合进行一次哈希运算,需要满足得到的哈希结果小于某一目标数值(target),因此每个节点会不断更改随机数进行尝试,

直到得到满足条件的随机数。然后该节点会将结果向整个区块链网络广播,其他节点验证属实后就会接纳此节点生成的区块,放到自己的链条上,得到一条“最长链”。

$$\text{Hash}(\text{block}, \text{nonce}) < \text{target}$$

PoW 机制是最早和理论上最安全的公有链算法,参与竞争消息排序的节点越多,系统会自动调整 target 的大小,控制难度,使得出块的速度大体稳定在一定区间,这使得篡改区块链的内容越困难,而且难以预测下轮的主节点避免了数据打包时可能受到的干扰。但是 PoW 的缺陷也很明显,巨量算力的消耗造成了资源和设备的浪费,而且根据调查显示 90% 的算力掌握在 5 家“矿场”中,这无疑违背了分布的原则。而且长达 10 分钟到 1 小时的确认时间以及 7tx/s 的交易速度使得应用场景非常受限^[12]。

1.2 PoS 中主节点的选择方法

2011 年 7 月 Quantum Mechanic 提出了权益证明 PoS 共识算法,提出将币龄引入以减少节点寻找随机数的难度。币龄指的是节点拥有的币乘以币剩余的时间。PoS 算法原理如下:

$$\text{Hash}(\text{block}, \text{nonce}) < \text{target} * \text{coinage}$$

$$\text{Coinage} = \text{币的个数} * \text{币的未使用时间}$$

其中, target 功能与 PoW 中的目标数一致,用来控制出块速度,而拥有更多,时间更长的余额的人则可以以较低的代价计算出 nonce 从而获得记账权。但是这个算法对于币龄没有限制,于是有用户开始囤币升值,以期获得在系统内的更多投票权。因此 Peercoin 修改了币龄的计算方法由线性增长改为了衰减增长和控制 coinage 的最大值,使得囤币者的效用增长不再明显。

1.3 PBFT 中主节点的选择方法

2016 年,IBM 提出了基于 PBFT^[9]中的共识协议 hyperledger,采用了授权式共识,在一个已经被验证过的节点群中,节点间链接配置被称为视图(view),在一个视图中,一个节点是主节点,其他节点是备份,当需要视图切换时(view-change)通过计算 $P = V \bmod |R|$ 选择下一个主节点,其中 V 是视图编号, R 是参与共识的节点数。主节点的更换通常由以下情况触发:主节点宕机超时;主节点是恶意节点,传播恶意消息,被过半节点发现异常;设置的阶段定时器超时。对于主节点 P ,由于在更替中视图遵循是 $V, V+1, V+2$ 原则使得攻击者容易推测下一个可能的主节点,进而攻击控制,而且如果主节点擅自增删,修改客户端的消息,其他节点未能发现异常,这对用户的安全性无疑带来极大危险。PBFT 共识算法流程如图 1 所示。

1.4 DPoS 中主节点的选择方法

2013 年 8 月,比特股(bitshare)则采用授权股份证

明算法 (delegated proof-of-stake, DPoS) 即系统中每个节点可以将其持有的股份权益作为选票授予一个证人, 获得票数最多且愿意成为证人的前 N 个节点将进入“证人名单”, 按照既定的时间表轮流对交易进行打包结算并且签署 (即生产) 新区块。每经过一个维护间隔 (maintenance interval) 时间 (1 天), 选票就会被统计一次, 届时活跃证人的名单也会更新一次。然后将证人名单洗牌 (shuffle), 并且每个证人节点会轮流地在固定的预先计划好的 2 秒内生产一个区块。当所有证人轮完之后, 他们又将被洗牌。

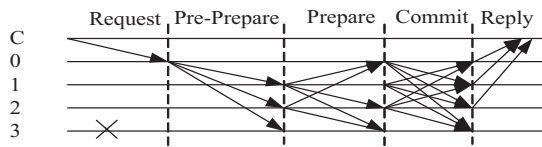


图 1 PBFT 共识算法流程

1.5 总结

综上所述, 主节点的选择一般出自以下几种方式: 算力竞争 (PoW); 资产所有权竞争 (PoS); 轮巡 (PBFT); 投票+轮巡 (DPoS)。在众多改进算法中, 除了 PoW 算法, 一般是通过增加随机选择的特性防止主节点, 如果主节点被腐蚀, 或者被恶意控制, 在算法中一般只考虑对节点的不统一欺骗即主节点对 P_1 发出 $A \rightarrow B$ 却对 P_2, P_3 发出 $A \rightarrow C$ 的交易命令。但是对于主节点篡改交易信息或凭空杜撰统一的消息记录, 即对原本 $A \rightarrow B$ 的交易信息被篡改为 $A \rightarrow C$, 统一发布给 P_1, P_2, P_3 。在这种情况下, 现有的共识机制无法识别拦截。Hyperledger 等联盟链一般依托线下的强身份认证使得主节点一旦作恶将被追溯, Carboni 引入了追溯机制使得作恶节点会被惩罚^[13], 但考虑到一般攻击方并不会是主节点的现实所有者, 更可能是被植入后门的节点, 造成的损失将很难挽回。因此一种对性能要求不大并能监视异常主节点行为的算法, 对提高有明确主节点主导的区块链算法安全性的提升很有必要。

2 改进方案

2.1 SBFT 共识流程结构

SBFT 算法分为 6 个阶段, 分别是请求、预准备、投票签名、完全共识证明、签名状态、完全执行证明, 流程如图 2 所示。

(1) 请求阶段: 客户端向主节点发送消息 m 。

(2) 预准备阶段: 主节点收到客户端发来的消息 m 后, 为其分配编号, 并将收到的消息按编号打包成块, 发送给其他节点。

(3) 投票签名阶段: 各节点对收到消息进行核验, 确认后使用门限签名 (BLS), 将签名消息发给收集节点 C。

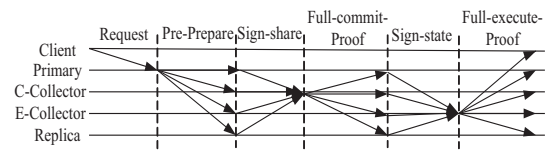


图 2 SBFT 共识算法流程

(4) 完全共识证明阶段: 每一个收集节点会收集签名, 然后为该块创建一个完全共识证明, 将其发送给所有节点, 一旦节点接收到完全共识消息, 就会对这个块进行共识同步, 然后执行块内记录的交易信息。

(5) 签名状态阶段: 当从节点执行完块中所有交易后, 开始创建一个完全执行证明, 并进行门限签名, 然后发送消息给收集节点 E。

(6) 完全执行证明阶段: 收集节点 E 收集签名, 达到门限要求后, 为该块创建一个完全执行完成证明, 并告诉所有节点和客户端当前状态是持久的, 并且交易操作已经被执行。

2.2 共识流程优化

为解决共识算法过度依赖主节点的问题, 引入监督节点 (supervisor) 对主节点的排序结果进行监督校准。流程如图 3 所示。如果主节点篡改客户端发来的消息, 除非能同时控制监督节点否则必然会造成消息摘要的不一致, 从而发现主节点异常, 没有必要通过等待收集 $f+1$ 个异常证明来提出 view-change。对于某些安全要求不高的场合, 还可以采用采纳多数的方式, 即在一个主节点两个监督节点的情况下, 当主节点和一个监督节点的摘要不同, 但与另一个监督节点相同时, 可以认为是系统误差造成, 继续执行后续步骤。

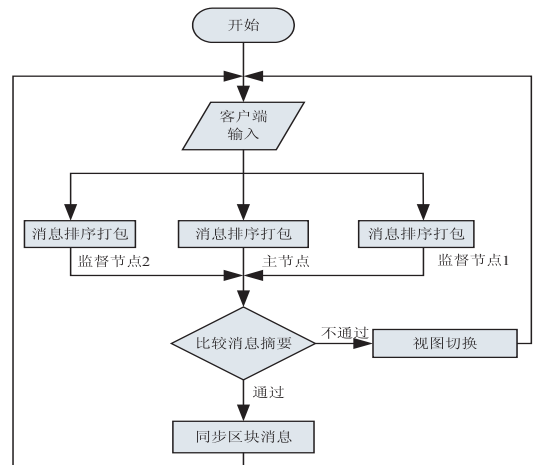


图 3 优化后的共识流程

优化后的流程具体如图 4 所示。

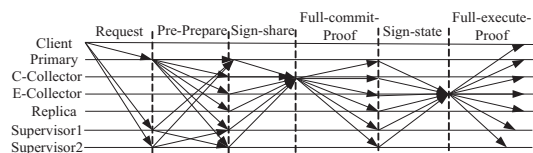


图 4 优化后的 SBFT 共识算法

(1) 请求阶段: 客户端提交消息 $\langle "request", m, c, t \rangle_{sig}$ 给主节点和监督节点, m 表示客户端向区块链请求写入的交易数据, c 表示客户端编号, t 表示时间戳, 一般来说由本地时钟控制, 其精度应当满足系统要求的 $\Delta\lambda$ 。客户端发出的时间戳是严格单调递增的。 sig 是客户端对这个消息的签名, 用以证明消息确实是此客户端发出的。消息发送成功后, 客户端启动计时器, 计算消息是否发送超时。

(2) 预准备阶段: 主节点接受到客户端发来的请求后, 通过时间排序为消息 m 分配编号, 满足打包条件后, 将消息按序号打包成块 ($block_p$), 并生成区块的摘要 $hash(block_p)$ 以及交易列表 $\{c_p, hash(m)_p, t_p\}$, 交易列表中存放了区块中所有提交消息的客户端编号和对应的提交时间戳。将 $\langle block_p, hash(block_p), \{c_p, hash(m)_p, t_p\} \rangle_{sig}$ 发送给其他节点; 监督节点接受到客户端发来的请求后, 按时间戳递增顺序为消息 m 分配编号, 满足打包条件后, 将消息按序号打包成块 ($block_s$) 并生成区块的摘要 $hash(m_s)$, 以及交易列表 $\{c_s, t_s\}$, 将 $\langle hash(block_s), \{c_s, t_s\} \rangle_{sig}$ 发送给其他节点。

(3) 投票签名阶段: 一般节点对收到消息进行核实验, 确认后使用门限签名 (BLS), 将签名消息发给收集节点 C。主节点和监督节点将收到的其他节点的摘要与自己生成的摘要比较, 生成比较消息。将比较消息发送给收集节点。

(4) 完全共识证明阶段: 收集节点 C 对比较消息验证, 不通过则进入视图转换, 通过则继续收集签名, 然后为该块创建一个 full commit proof 消息, 发送给所有节点, 一旦从节点接收到 full commit, 就会对这个块进行共识同步, 然后执行块内记录的交易信息。

(5) 与原方案相同。

(6) 与原方案相同。

3 特性分析

3.1 基于时间排序

主节点和监督节点每 Δt 进行一次打包操作, 抽取 $\langle c, hash(m), t \rangle$, 构造形如以 t 为序号的消息矩阵。

$$\begin{bmatrix} c_1 & hash(m_1) & t_1 \\ c_2 & hash(m_2) & t_2 \\ c_3 & hash(m_3) & t_3 \end{bmatrix}$$

受制于现实条件制约, 节点广播的信息不会同时抵达其他节点。可能在某些节点出现如下情况。

$$\begin{bmatrix} c_1 & hash(m_1) & t_1 \\ c_3 & hash(m_3) & t_3 \\ c_2 & hash(m_2) & t_2 \end{bmatrix}$$

这时需要通过重新排序实现顺序, 排序后的消息矩阵成为 $block$, 如果某条消息因为延时, 一部分节点将此消息打包入 $block_i$, 但另一部分节点中将其打包入 $block_{i+1}$, 将使得消息无法排序。因此, 引入了时间水位 $[h, H]$ 和延时排队的概念。低水位 h 等于上一个打包完成区块的最大时间戳编号, 高水位为 $t_H = t_h + \Delta t$, 其中 Δt 是指定的数值, 等于区块打包消息的时间间隔, 其中 $t_H + \Delta\lambda < t_{now}$, $\Delta\lambda$ 为系统造成的各个节点的延迟误差, t_{now} 为执行对 Δt 内消息进行打包动作的时刻。当系统的消息经过充分的传递和排序后, 对其进行排序打包, 避免出现乱序, 消息分在不同包内的情况下 (见图 5)。

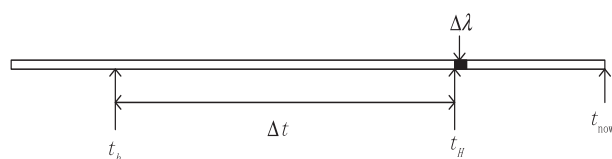


图 5 消息队列

3.2 消息比较

在投票阶段各节点将自己的区块摘要同其他节点的摘要进行比较^[14], 相同为 1, 不同为 0。若摘要全部相同则生成一条 (1, 1, 1) 消息, 若主节点摘要为 A, 监督节点 1 摘要为 B, 监督节点 2 摘要为 A, 则主节点生成 (1, 0, 1), 监督节点 1 生成 (0, 1, 0), 监督节点 2 生成 (1, 0, 1), 若主节点摘要为 B, 监督节点 1 摘要为 A, 监督节点 2 摘要为 A, 则主节点生成 (1, 0, 0), 监督节点 1 生成 (0, 1, 1), 监督节点 2 生成 (0, 1, 1)。对不同的任务场景可以使用不同的错误容忍策略, 对高要求场景可以使用严格模式必须 (1, 1, 1) 才能通过审查, 对于宽容度较高的任务类型可以接受 (1, 0, 1) 或者 (1, 1, 0) 但是拒绝 (1, 0, 0)。

4 仿真结果分析

改进方案通过引入了时间水位 $[h, H]$ 和延时排队以及消息比较的概念, 实现了对主节点的监督。通过引入不同的监督节点数, 增加攻击主节点操纵数据的成本。如果采用严格模式, 则使得攻击成功的难度指数上升。如果采用宽容模式, 敌手设定为 $n = 2f + 1$, 有总量为 n 节点, M_p 个敌手节点, 当选 $x - 1$ 个节点作为监督节点, 则被敌手攻击成功的概率 Pr 为:

$$Pr(n, M_p, x) = \begin{cases} \frac{\sum_{i=\frac{x}{2}}^{M_p} (C_{n-M_p}^{M_p-i} C_{M_p}^i)}{C_n^x} & (x \geq M_p) \\ \frac{\sum_{i=\frac{x}{2}}^x (C_{n-x}^{x-i} C_x^i)}{C_n^x} & (x < M_p) \end{cases}$$

在 $n=100$, $M_p=6$, $x=9$ 的条件下,敌手攻击成功的概率为 $2.97e-10$,如果系统打包出块的速度是每分钟一块,每天运行 20 小时,则平均 2 805 年才会出现一次攻击成功。图 6 是在 $n=100$ 的情况下以 \log_{10} 计的攻击成功概率图。以攻击成功的概率小于 $e-8$ 为通过标准(千万分之一),可以看到敌手攻击成功的概率随着敌手控制点的增加而增加,而且当敌手控制节点数目不变时增加背书节点数目可以使攻击成功概率呈指数下降,即使敌手控制了相对较多的节点 $M_p=10$;当 $x=13$ 时攻击成功概率仍然仅为 $2.001e-09$,充分体现了安全性。在实际使用中根据需求设置 n 和 x ,以满足不同的场景需求和不同强度的安全需要。

表 1 敌手攻击成功概率

监督节点数	敌手数目				
	1	2	3	4	5
0(原始方案)	0.01	0.02	0.03	0.04	0.05
2	0	$6.184e-06$	$1.80e-03$	$3.6e-03$	$5.9e-03$
4	0	0	$1.33e-08$	$5.11e-06$	$5.99e-04$
6	0	0	0	$6.25e-11$	$2.97e-08$

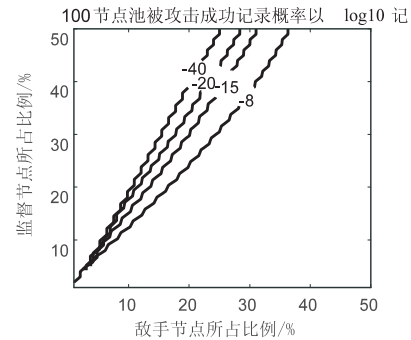
通过比较可以看到,引入监督节点后攻击成功的概率大大降低,效果明显。

5 结束语

主节点的极高权限使得恶意攻击者一旦得手收益极大,这也是各个算法极力避免主节点提前计算出的原因,但是系统为了维持数据一致又必须由单节点进行记账操作,因此引入一种带有主节点监督的共识算法被视为效率与安全的一中折中方案。文中针对共识算法中存在的对主节点消息过度依赖的问题,提出一种基于时间排序的监督共识方案。通过引入监督节点,在系统内部利用现有资源对主节点的消息进行校准,以较小代价显著提高主节点传播消息的造假成本。通过消息的排队解决多节点记录时的异步网络带来的消息传递延迟。仿真结果表明,基于时间排序的监督共识可以很好地在有部分恶意节点的网络中以较低的代价显著增加主节点作弊的成本,有效提升了系统的安全性,并且可以通过设置不同的共识模式、引入不同的监督节点调节系统的抗攻击能力。对有弹性安全需求的系统有较大应用潜力。

参考文献:

- [1] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system[EB/OL]. (2008-04-10). <http://bitcoins.info/bitcoin.pdf>.
- [2] 陈春玲,沈阳,余瀚. 去中心化的征信系统模型研究[J]. 计算机技术与应用,2019,29(3):122-126.
- [3] 韩璇,袁勇,王飞跃. 区块链安全问题:研究现状与展

图 6 敌手攻击成功概率图(以 \log_{10} 记)

在有 100 个节点的典型环境中,敌手数目为 1~5 时,不同监督节点数下攻击成功的概率细节如表 1 所示。

望[J]. 自动化学报,2019,45(1):206-225.

- [4] 孙国梓,王纪涛,谷宇. 区块链技术安全威胁分析[J]. 南京邮电大学学报:自然科学版,2019,39(5):48-62.
- [5] 毛志来,刘亚楠,孙惠平,等. 区块链性能扩展与安全研究[J]. 信息安全,2020,20(3):56-64.
- [6] 翟社平,李兆兆,段宏宇,等. 区块链关键技术中的数据一致性研究[J]. 计算机技术与应用,2018,28(9):94-100.
- [7] LYNCH N, GILBERT S. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services[J]. SIGACT News,2002,33(2):51-59.
- [8] LAMPORT L. The part-time parliament[J]. ACM Transactions on Computer Systems,1998,16(2):133-169.
- [9] CASTRO M, LISKOV B. Practical byzantine fault tolerance[J]. Operating Systems Review,1998(Special):173-186.
- [10] KOTLA R, CLEMENT A, WONG E, et al. Zyzzyva: speculative byzantine fault tolerance[J]. Communications of the ACM,2008,51(11):86-95.
- [11] GUETA G G. SBFT: a scalable decentralized trust infrastructure for blockchains[C]//2019 49th annual IEEE/IFIP international conference on dependable systems and networks (DSN). Oregon, USA: IEEE,2019.
- [12] 王雷,任南,李保珍. 区块链 51% 双花攻击的进化博弈及防控策略研究[J]. 计算机工程与应用,2020,56(3):28-34.
- [13] CARBONI D. Feedback based reputation on top of the bitcoin blockchain[EB/OL]. (2015-02-10). <https://arxiv.org/ftp/arxiv/papers/1502/1502.01504.pdf>.
- [14] 张华伟,熊璋,欧阳元新. 分布式系统中异地数据库的数据一致性维护[J]. 计算机工程与应用,2004,40(23):172-175.