

快速响应的高效多值拜占庭共识方案

周旺^{1,2}, 胡红钢^{1,2}, 俞能海^{1,2}

(1. 中国科学院电磁空间信息重点实验室, 安徽 合肥 230027;
2. 中国科学技术大学网络空间安全学院, 安徽 合肥 230027)

摘要: 由于网络设备的增多和传输环境的不确定性, 消息时延同样具有不确定性, 异步共识协议发挥出更多优势。Miller 等于 2016 年提出第一个异步共识协议 HoneyBadgerBFT, 但其在实现高吞吐量的同时传输效率依然可以再优化。针对 HoneyBadgerBFT 中的广播协议进行改进, 减少广播过程中的消息复杂度, 同时增加可选的消息请求过程, 以达到快速响应和高效传输的效果。

关键词: 快速响应; 高传输效率; 拜占庭协议; 共识方案

中图分类号: TP309.3

文献标识码: A

doi: 10.11959/j.issn.2096-109x.20201006

Rapid responsive and efficient multi-valued Byzantine consensus scheme

ZHOU Wang^{1,2}, HU Honggang^{1,2}, YU Nenghai^{1,2}

1. Key Laboratory of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei 230027, China
2. School of Cyber Science, University of Science and Technology of China, Hefei 230027, China

Abstract: Due to the increase of network equipments and the uncertainty of the transmission environment, the message delay is also uncertain, and the asynchronous consensus protocol possesses more advantages. Miller et al proposed the first asynchronous consensus protocol HoneyBadgerBFT in 2016, but its transmission efficiency can be optimized furthermore while achieving high throughput. The broadcast protocol in HoneyBadgerBFT was improved by reducing the message complexity in the broadcast process, and adding optional message request process to achieve rapid response and efficient transmission.

Keywords: rapid response, efficient transmission, Byzantine protocol, consensus scheme

收稿日期: 2020-01-14; 修回日期: 2020-03-02

通信作者: 胡红钢, hghu2005@ustc.edu.cn

基金项目: 国家自然科学基金 (61632013, 61972370)

Foundation Item: The National Natural Science Foundation of China (61632013, 61972370)

论文引用格式: 周旺, 胡红钢, 俞能海. 快速响应的高效多值拜占庭共识方案[J]. 网络与信息安全学报, 2021, 7(1): 57-64.
ZHOU W, HU H G, YU N H. Rapid responsive and efficient multi-valued Byzantine consensus scheme[J]. Chinese Journal of Network and Information Security, 2021, 7(1): 57-64.

1 引言

自2008年中本聪首次提出比特币^[1]的概念后,区块链作为其底层技术也被广泛关注。区块链本质上是一个分布式数据库,同时具有去中心化、开放性、自治性、信息不可篡改和匿名性这五大特性。区块链系统与传统数据库系统的区别在于其能够在不依赖第三方的情况下,在分布式环境中正确运行,其核心之一是共识机制,即如何在—组节点之间达成消息的一致。

对于共识机制的研究一直热度不减,从20世纪70年代末开始,研究人员便对分布式系统中的容错问题进行了深入研究。对于宕机容错(CFT)问题,Lamport于1989年做出了开拓性的工作,提出了一个新的状态机复制协议——Paxos^[2],并在2001年进行了解释^[3]。1982年,Lamport等提出了一个新的问题——拜占庭将军问题^[4],对于新的拜占庭容错(BFT)问题,Castro和Liskov于1999提出了著名的PBFT^[5]方案。2008年以后,随着比特币的关注度升高,同时由于其使用的工作量证明(PoW)机制的公平性,实现简单,PoW受到追捧,以太坊^[6]、莱特币等部分主流“数字货币”使用的均是PoW机制。但鉴于其消耗大量计算资源与电力资源,相关社区一直在寻找PoW的替代机制。权益证明(PoS)依据系统中的稀缺资源,如资金,选取出块者,抵抗女巫攻击的同时不用消耗大量资源,并且近些年来有研究人员提出了可证明安全的PoS协议^[7-9],所以PoS机制正逐渐被区块链系统设计者考虑使用。为了解决共识机制的可拓展性与低吞吐量问题,一系列新的方案与机制被提出^[10-13],所以对于共识的研究会根据应用场景、需求的变化一直持续下去。

上述协议为保证可用性大多需要弱同步甚至同步环境的假定,但当底层网络由于某些原因导致消息转发缓慢甚至停止时,同步协议的性能将明显下降,异步共识协议在这种环境下展示出较大优势。HoneybadgerBFT^[14]是由Miller等于2016年提出的第一个异步拜占庭协议,但协议过程中传输效率可进一步优化,文献[15]直接将HoneybadgerBFT中的异步公共子集(ACS)协议

作为一个单独模块,提出一种“先共识消息哈希,后请求缺失消息”的共识思路,单位传输代价优于HoneybadgerBFT。但文献[15]对消息先通过广播哈希值,接收多个签名的方式进行处理,增加了时间消耗,无法快速响应共识请求。本文则对ACS协议进行改进,替换其中的广播子模块,增加可选的消息请求模块,避免对消息预处理的同时减小带宽消耗,达到快速响应、高效传输的效果。

2 预备知识

2.1 时间假设

底层网络可能因为拥堵或攻击导致消息转发缓慢,甚至由于恶意节点的操作使消息停止转发,这使共识协议的设计、运行变得愈加困难。研究人员针对不同应用需求,根据消息延迟的界定义了不同的时间假设。

1) 同步:发送的每个消息最多在延迟一段时间后收到,即消息的延迟存在上界。

2) 部分同步^[16]:有两种不同的情形。第一种是网络存在最大延迟,但对于参与者是未知的;第二种是在一个未知时间(被称为全局标准时间,GTS)后,网络变为完全同步。

3) 弱同步:延迟的界随时间变化,但增长速度不会比时间的多项式函数级快。

4) 异步:消息的延迟没有上界,但诚实用户之间发送的消息最终会达到。

著名的PBFT算法在保证可用性和安全性的前提下提供了 $\frac{n-1}{3}$ 的容错性,其中,安全性不依赖于同步假设,但可用性依赖于同步假设。而HoneyBadgerBFT则是在异步环境满足可用性的情况下也可达到高吞吐量。

2.2 HoneyBagerBFT

HoneyBadgerBFT协议是Miller等在2016年提出的一种BFT协议。其作为第一个异步的BFT共识协议,完全不依赖于网络中对时间条件的假设。与传统的PBFT共识协议相比,当节点增加时,效率不会明显下降。

此协议假定每两个节点都有可靠的通信信道,诚实节点之间的消息一定会被投递。在整个

网络中，节点总数大于 $\frac{1}{3}$ 的敌手节点，即 $N \geq 3f+1$ ，其中 f 为敌手数， N 为总节点数。协议执行时，用户将交易发送给系统中的节点，节点将用户交易存储在自己的缓存队列中，在一轮开始时，系统中每个节点取缓存中交易的子集作为建议值输入共识协议中，协议的输出是这些建议值集合的子集。与之前的 BFT 协议相比，HoneyBadgerBFT 通信复杂度为 $O(N)$ 。

HoneyBadgerBFT 由两个模块组成：门限加密 (TPKE) 模块和异步公共子集 (ACS) 模块。ACS 模块负责最终交易集合的共识，TPKE 模块保证协议的公平性，防止有针对性的审查攻击。ACS 模块也由两个模块组成：可靠广播 (RBC^[17]) 模块和二元共识 (ABA) 模块。RBC 模块将每个节点提出的建议值广播给其他节点，ABA 模块对所有节点之间在 N 个比特上达成共识，第 i 个比特为 1，代表节点 i 提出的建议值被包含在最终交易集合中。本文关注 ACS 协议中的 RBC 模块，ACS 协议及 RBC 协议流程如下。

ACS 协议流程 (以 P_i 为例)

//系统维护 N 个 RBC 实例， N 个 ABA 实例

输入 每个节点的建议值

输出 所有建议值的一个子集

1) 选取建议值 v_i ，将其作为 RBC 的输入；

2) 当从 RBC 收到 v_j 时，若还未向 ABA _{j} 提供过输入，则将 1 作为 ABA _{j} 的输入；

3) 当已向 $N-f$ 个 ABA 实例提供过输入，则将 0 作为未提供输入的 ABA 实例的输入；

4) 当所有 ABA 实例均已完成，取输出为 1 的 ABA 实例对应 RBC 输出的并集作为最终共识的交易集合。

RBC 协议流程 (以 P_i 为例，发送者为 P_{sender})

1) 当 P_i 是 P_{sender} ，输入为 v 时，将 v 作为 $(N-2f, N)$ -纠删码的输入，输出为 $\{s_j\}_{j \in [N]}$ ，令 h 为以 $\{s_j\}$ 计算出的默克尔树根，向每个 P_j 发送 $\text{VAL}(h, b_j, s_j)$ ，其中 b_j 是第 j 个默克尔树枝。

2) 当从 P_{sender} 收到 $\text{VAL}(h, b_i, s_i)$ 时，广播 $\text{ECHO}(h, b_i, s_i)$ 。

3) 当从 P_j 收到 $\text{ECHO}(h, b_j, s_j)$ 时，检查 b_j 对于树根 h 以及 s_j 是否有效，如果无效，则抛弃

$\text{ECHO}(h, b_j, s_j)$ 。

4) 当从 $N-f$ 个不同的节点收到有效的 $\text{ECHO}(h, \bullet, \bullet)$ ，使用 $N-2f$ 个 $\{s_j\}$ 重建默克尔树根 h' ，若 $h' \neq h$ 则丢弃；否则，若 $\text{READY}(h)$ 未被发送过，广播 $\text{READY}(h)$ 。

5) 当收到 $f+1$ 个匹配的 $\text{READY}(h)$ ，若还未发送过 READY 消息，则广播 $\text{READY}(h)$ 。

6) 当收到 $2f+1$ 个匹配的 $\text{READY}(h)$ ，等待 $N-2f$ 个 ECHO 消息，解码恢复 v 。

由 RBC 协议流程可知，在每个 VAL 消息、ECHO 消息中都需要将纠删码编码块 s_j 加入，以重建默克尔树根，并以此验证消息正确性，但同时占用了更多的带宽，增加了消息复杂度。本文从此入手，使用带有指纹的异步可验证消息分发协议 (AVID-FP^[18]) 替换 ACS 协议中的 RBC 协议，使在广播过程中加入极少信息便能验证消息的正确性，同时可以完整地获得广播的消息。

2.3 AVID-FP

AVID-FP^[18] 是 Hendricks 于 2007 年提出的用于优化纠删码带宽的方案，其使用同态指纹作为每次广播的信息。同态指纹保持了纠删码的结构，允许每个编码块可以单独验证其是否对应特定的原信息，于是每个节点均可以利用首次收到的编码块以及对应的指纹信息验证收到的编码块的正确性，同时仅需在之后的广播消息中加入指纹信息即可确保所有诚实节点拥有的编码块对应于同一个原信息，由于其带宽消耗最优，文献[19]中使用 AVID-FP 作为线性存储的实现方案。

AVID-FP 协议流程如下。

AVID-FP 协议流程

//客户端分发消息

1) 对于要分发的消息 B ，将其作为 (m, N) -纠删码的输入，输出为 $d_1 \cdots d_n$ ，根据 $d_1 \cdots d_n$ 计算出 fpcc ，向服务器 S_i 发送 d_i 及 fpcc 。

2) 当第一次从 S_i 收到 (STORED)，计数器加 1，当计数器的值等于 $2f+1$ 时，返回成功。

//客户端恢复消息

1) 向所有服务器发送消息恢复请求 (RETRIEVED)；

2) 当收到服务器 S_i 发送的 $(\widehat{\text{fpcc}}, (\text{fpcc}', d))$ ，验证 d 的正确性，并等待 $f+1$ 个相同的 fpcc 。

3) 当收到的对应的编码块数量为 m 时, 恢复消息。

//服务器端

1) 当从客户端收到分发的编码块 d_i 及 fpcc , 验证 d_i 与 fpcc 是否匹配, 若匹配, 向其他服务器广播 (ECHO, fpcc), 并令 $\text{echoed} = \langle \text{fpcc}, d_i \rangle$ 。

2) 当从其他服务器收到 (ECHO, fpcc), 若 ECHO 消息数为 $m+f$, 并且 READY 消息数少于 $f+1$, 则向其他服务器广播 (READY, fpcc)。

3) 当从其他服务器收到 (READY, fpcc), 若 READY 消息数为 $f+1$, 并且 ECHO 消息数小于 $m+f$, 则向其他服务器广播 (READY, fpcc);

4) 当 READY 消息数为 $2f+1$ 时, 令 $\text{verified}=\text{fpcc}$, 向客户端发送 (STROED)。

5) 当从客户端收到消息恢复请求 (RETRIEVED) 时, 向客户端发送 (VERIFIED, echoed)。

不同于之前的 AVID^[20]协议, AVID-FP 在 ECHO 消息以及 READY 消息中仅附带指纹交叉验证和 fpcc , 而 fpcc 的大小相对于编码块可忽略不计, 假设系统中有 N 个节点, 原消息块 B 经过 (m, N) -纠删码编码, 任意 m 个编码块即可恢复原消息, 其中 $m \geq f+1$, $N=m+2f$, 于是广播原消息块 B 需要的带宽为 $O(N \frac{|B|}{m}) = O(\frac{m+2f}{m} |B|) = O(|B|)$ 。经过 AVID-FP 之后, 得到的仅是对于 fpcc 的共识, 每个节点仍然只有一份编码块, 但这对于 ABA 的执行是足够的, 即当节点 j 收到 $2f+1$ 个与节点 i 分发的编码块对应的 fpcc 时, 向 ABA _{i} 输入 1。所以, 若想获得完整的原消息, 需要添加一个消息恢复模块, 同时消息的恢复与否并不影响共识协议的进行, 于是节点可以根据自身的网络环境决定何时恢复原消息, 而且即使加入消息恢复模块, 带宽消耗仍然比 HoneyBadgerBFT 更优。

3 协议设计

在 RBC 广播协议中, 为了验证每个收到的广播消息的正确性, 即是否对应于特定原消息, 在广播消息中需加入编码块、对应的默克尔树根以及默克尔树路径信息, 所以, 对于大小不可忽略

的输入, RBC 协议执行时将占用大量带宽。文献[15]直接将 ACS 协议作为单独的模块, 并依据本地节点的消息和最终建议值集合中的消息有交集的假定, 对消息哈希先签名后广播, 在收到多个签名后组成建议值特定格式项, 然后使用 ACS 先共识消息哈希, 再请求缺失消息。其通过形成特殊格式的建议值, 将该值作为 ACS 协议的输入, 在 ACS 协议的广播模块执行时, 广播消息的大小大大降低。本文通过对 ACS 协议的广播子模块进行替换, 使用 AVID-FP 进行消息的广播, 在不需要对输入做特定预处理的情况下, 广播模块执行时, 其广播消息的大小与消息请求模块请求内容的大小相比可忽略, 而且由于使用了同态指纹技术, 对每个收到的广播消息均可验证其正确性, 于是在不需要更多预处理操作的同时达到高效传输的效果。

需要注意的是, 使用 AVID-FP 替换广播子模块后的 ACS 协议在执行完后, 共识的结果依旧是大小可忽略的 fpcc 集合, 为了获得最终的共识消息, 需要增加额外的消息请求模块。但在 AVID-FP 算法流程中有消息恢复的部分, 于是恢复消息模块可由各节点以客户端的角色运行消息恢复部分得到最终共识消息。

协议整体流程如下: 系统中维护 N 个 AVID-FP 实例, N 个 ABA 实例, 每个节点均选取一定交易消息作为建议值, 然后以客户端的角色运行 AVID-FP 协议, 分发选取的建议值, 并以服务器端的角色共识 fpcc ; 每当一个 fpcc 验证, 在向客户端发送 (STROED) 消息时, 向对应的 ABA 实例输入 1; 若该节点已经向 $N-f$ 个 ABA 实例提供过输入, 则向其他所有 ABA 实例输入 0; 等待所有 ABA 实例完成, 取输出为 1 的 ABA 实例对应的 AVID-FP 输出的 fpcc 的并集作为最终建议值指纹集合; 若此时节点网络环境不佳, 则可以先进行下一轮共识, 而不需要立即请求 fpcc 对应的建议值集合, 否则节点依次广播消息恢复请求 (RETRIEVED), 在最优情况下只需从其他 $f+1$ 个节点收到每个建议值的编码块即可恢复相应的建议值, 进而得出最终建议值集合。改进的 ACS 协议流程如下。

改进的 ACS 协议流程 (以 P_i 为例)

//系统维护 N 个 AVID-FP 实例, N 个 ABA 实例

输入 每个节点的建议值

输出 所有建议值指纹的一个子集或所有建议值的一个子集

1) 选取建议值 v_i , 将其作为 AVID-FP_i 客户端的输入。

2) 当从 AVID-FP_j 验证过 fpcc 即 verified = fpcc 时, 若还未向 ABA_j 提供过输入, 则将 1 作为 ABA_j 的输入。

3) 当已向 $N-f$ 个 ABA 实例提供过输入, 则将 0 作为未提供输入的 ABA 实例的输入。

4) 当所有 ABA 实例均已完成, 取输出为 1 的 ABA 实例对应的 AVID-FP 输出的 fpcc 并集作为最终共识的建议值指纹集合。

5) 由 fpcc 集合发送消息恢复请求, 并根据收到的编码块恢复消息集合。

在文献[15]中, 为了使本地节点的消息和最终建议值集合中的消息交集最大化, 在将消息 m 输入 ACS 协议之前, 需要对消息做处理, 形成如下格式的消息插入本地建议值集合中。

$$(\text{Hash}(m), \text{Set}_{\text{echo}}(m))$$

$\text{Set}_{\text{echo}}(m)$ 是包含了 $2f+1$ 个不同的签名项的集合, 其中每一项都具有如下格式。

$$(\text{PK}_j, \sigma_{\text{Hash}(m)})$$

两个域分别表示节点 P_j 的公钥以及 P_j 对 $\text{Hash}(m)$ 的签名。所以每个节点第一次收到某个消息 m 时, 都会对 $\text{Hash}(m)$ 签名, 并广播给其他节点, 并且需要等到对 $\text{Hash}(m)$ 的签名个数达到 $2f+1$ 个时才会加入建议队列, 等待成为建议值中的一项。虽然哈希值及签名的大小相比原消息的大小可忽略不计, 但花费更多的时间完成上述处理是毋庸置疑的。

在本文的协议中, 没有处理用户发来消息的时间消耗, 而且消息恢复模块不影响共识的进程, 若只考虑共识的 fpcc, 则可对参与的节点进行快速的响应; 即使考虑完全恢复最终建议值集合, 也会比文献[15]中的改进方案响应更快。改进前后流程如图 1 所示, 模块对比如表 1 所示。

4 传输效率分析

本文协议通信效率的计算包括从 AVID-FP 分

发编码块开始, 一直到所有节点恢复最终建议值集合, 其中, 由于 fpcc 的大小远小于编码块的大小, 所以计算通信代价时只关注编码块的传输。同时为了客观地评价和比较协议的通信效率, 对比时采用文献[15]中定义的“单位传输代价”, 即在一轮共识协议中, 协议的总传输代价与本轮中得到共识的用户请求数量的比值; 并且采用相同假设, 假定每个消息大小为 $|B|$, 经过 (m, N) -纠删码编码后每个编码块大小为 $\frac{|B|}{m}$,

任意 m 个编码块即可恢复原消息, 其中 $m \geq f+1$, $N=m+2f$ 。

4.1 本协议传输效率分析

在 AVID-FP 中, 只有第一次分发编码块 d_i 时占用大量带宽, 之后的广播消息中仅加入了大小可忽略的 fpcc, 所以 AVID-FP 传输代价为 $N \frac{|d_i|}{m} \approx 3|d_i|$, 本文协议中共有 N 个 AVID-FP 实例, 所有 AVID-FP 的传输代价为 $3N|d_i|$ 。假定每轮共识 t 个消息, 每个节点提供的建议值中有 $\frac{t}{N}$ 个消息, 则每个建议值编码块大小为 $\frac{t|B|}{Nm}$ 。

对于之后的消息恢复模块, 因为仅需 m 个编码块便可恢复出原消息, 对于某个建议值最优情况下仅需发送 m 个消息恢复请求, 收到 m 个诚实节点发送的建议值编码块即可恢复出此建议值。所以, 最优情况下的单位传输代价为

$$(3 \cdot \frac{t|B|}{Nm} \cdot N + N \cdot m \cdot \frac{t|B|}{Nm} \cdot N) / t \approx N|B|$$

最坏情况下系统中有 f 个敌手, 在消息恢复模块需要请求至少 $m+f$ 个编码块以恢复原建议值。于是, 最坏情况下单位传输代价为

$$(3 \cdot \frac{t|B|}{Nm} \cdot N + N \cdot (m+f) \cdot \frac{t|B|}{Nm} \cdot N) / t \approx 2N|B|$$

4.2 HoneyBadgerBFT 传输效率分析

HoneyBadgerBFT 中占有大量带宽的为 RBC 模块, 其采用 $(f+1, 3f+1)$ -纠删码编码交易消息。假设 HoneyBadgerBFT 每次选取 $\frac{t}{N}$ 个交易消息作为建议值, 建议值编码后的编码块大小为 $\frac{t|B|}{N(f+1)}$ 。在 RBC 协议中, 每个节点至少发送广

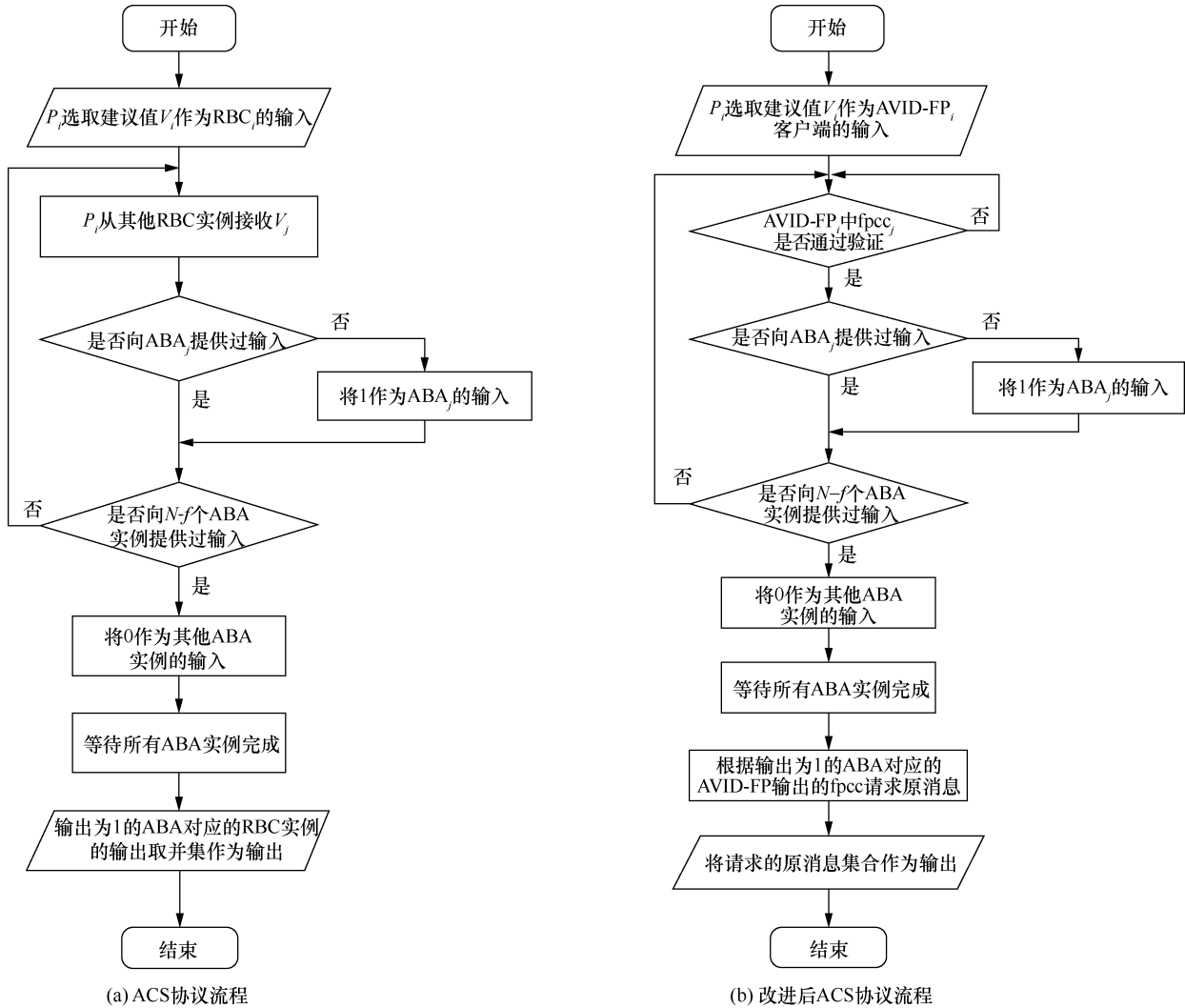


图1 改进前后 ACS 协议流程
Figure 1 ACS protocol flow charts before and after improvement

表1 改进前后模块对比
Table 1 Comparison of modules before and after improvement

协议	建议值预处理	广播模块	广播内容	共识结果	消息请求模块
本协议	无	AVID-FP	原消息的 fpcc	fpcc 集合	据 fpcc 请求
ACS 协议	无	RBC	原消息编码块密文	原消息密文	无
文献[15]	带签名建议值	RBC	原消息哈希值	哈希值集合	据哈希请求

播一个编码块，所以 RBC 传输代价为 $\frac{t|B|}{N(f+1)} \cdot N \cdot N \approx 3t|B|$ 。协议中共有 N 个 RBC 实例，则所有 RBC 的传输代价为 $3Nt|B|$ 。

在最优情况下，即

- 1) 系统中 N 个节点均为诚实节点；
- 2) 各个建议值之间没有重复消息；

3) 一轮中建议值均被共识。

HoneyBadgerBFT 的单位传输代价为

$$3Nt|B|/t = 3N|B|$$

在最坏情况下，即

- 1) 系统中有 f 个恶意节点；
- 2) 每轮只有 $2f+1$ 个建议值被共识，且其中的 f 个来自恶意节点， $f+1$ 个来自诚实节点。

所有恶意节点可以在各自建议值中提供大量重复内容, 此时最后共识的消息几乎来自诚实节点, 所以共识的消息数大约为 $(f+1) \cdot \frac{t}{N} + \frac{t}{N} \approx \frac{B}{3}$ 。于是最坏情况下 HoneyBadgerBFT 的单位传输代价为 $3Nt|B|/(\frac{t}{3}) = 9N|B|$ 。

4.3 文献[15]中共识方案传输效率分析

文献[15]直接将 ACS 协议作为单独模块, 使用了“共识消息哈希, 后请求缺失消息”的共识思路, 并预先对用户发来的消息进行了处理, 形成特定格式的建议值, 广播消息的大小与消息请求模块请求内容的大小相比可忽略不计, 于是在最优情况下, 即

- 1) 所有节点均是诚实节点;
- 2) 所有节点均有了建议值中的原消息。

所有节点不会对最终共识的哈希列表中的项发起原消息传输请求, 此时协议的单位传输代价为

$$(t \cdot |B| \cdot N) / t \approx N|B|$$

在最坏情况下, 即系统中有 f 个恶意节点, 恶意节点总可以发起 f 个原文请求消息, 每个原文请求消息收到 $2f+1$ 个回复, 每个回复中包含一个大小为 $\frac{|B|}{f}$ 的 RS code^[21]编码块, 此时的单位传输代价为

$$(t \cdot |B| \cdot N + t \cdot f \cdot (2f+1) \cdot \frac{|B|}{f}) / t \approx \frac{3}{5} N|B|$$

通过表 2 对比可知, 本文协议的传输效率在最优、最坏情况下均优于 HoneyBadgerBFT, 最优情况下与文献[15]相同, 最坏情况下比文献[15]略差, 响应速度由第 3 节分析可知, 本文协议比文献[15]响应速度更快。

表 2 单位传输代价比较
Table 2 Comparison of unit transmission cost

协议	单位传输代价 (最优情况)	单位传输代价 (最坏情况)
本文协议	$N B $	$2N B $
HoneyBadgerBFT	$3N B $	$9N B $
文献[15]	$N B $	$\frac{5N B }{3}$

5 结束语

在异步共识协议中不仅要考虑传输效率的高低, 同时需要关注对于请求的响应速度。本文在 HoneyBadgerBFT 的基础上, 使用修改的 AVID-FP 协议改进其 RBC 协议, 使在某些特定场景下不需要恢复最终建议值集合也能快速响应共识请求, 即使在请求完整建议值集合时也比文献[8]响应更快速, 同时传输代价比 HoneyBadgerBFT 在最优、最差情况下都有所改善。

值得关注的是, 在 ACS 与 ABA 协议中有多次广播操作, 消耗大部分带宽资源, 所以, 如何在减少广播操作的同时达到相同的功能是接下来的研究重点。

参考文献:

- [1] NAKAMOTO S. Bitcoin: a peer-to-peer electronic cash system[R]. 2008.
- [2] LAMPORT L. The part-time parliament[J]. ACM Transactions on Computer Systems (TOCS), 1998, 16(2): 133-169.
- [3] LAMPORT L. Paxos made simple[J]. ACM Sigact News, 2001, 32(4): 18-25.
- [4] LAMPORT L, SHOSTAK R, PEASE M. The Byzantine generals problem[J]. ACM Transactions on Programming Languages and Systems (TOPLAS), 1982, 4(3): 382-401.
- [5] CASTRO M, LISKOV B. Practical Byzantine fault tolerance[C]//OSDI. 1999: 173-186.
- [6] WOOD G. Ethereum: a secure decentralised generalised transaction ledger[J]. Ethereum Project Yellow Paper, 2014, 151: 1-32.
- [7] KIAYIAS A, RUSSELL A, DAVID B, et al. Ouroboros: a provably secure proof-of-stake blockchain protocol[C]//Annual International Cryptology Conference. 2017: 357-388.
- [8] DAVID B, GAŽI P, KIAYIAS A, et al. Ouroboros praos: an adaptively-secure, semi-synchronous proof-of-stake blockchain[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2018: 66-98.
- [9] BENTOV I, PASS R, SHI E. Snow white: provably secure proofs of stake[J]. IACR Cryptology ePrint Archive, 2016: 919.
- [10] EYAL I, GENCER A E, SIRER E G, et al. Bitcoin-ng: a scalable blockchain protocol[C]//13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16). 2016: 45-59.
- [11] KOKORIS-KOGIAS E, JOVANOVIĆ P, GASSER L, et al. Omniledger: a secure, scale-out, decentralized ledger via sharding[C]//2018 IEEE Symposium on Security and Privacy (SP). 2018: 583-598.
- [12] ZAMANI M, MOVAHEDI M, RAYKOVA M. Rapidchain: scaling blockchain via full sharding[C]//2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 931-948.
- [13] WANG J, WANG H. Monoxide: scale out blockchains with asynchronous consensus zones[C]//16th USENIX Symposium on Net-

worked Systems Design and Implementation (NSDI 19). 2019: 95-112.

- [14] MILLER A, XIA Y, CROMAN K, et al. The honey badger of BFT protocols[C]//2016 ACM SIGSAC Conference on Computer and Communications Security. 2016: 31-42
- [15] 郭兵勇, 李新宇. 一个高传输效率的多值拜占庭共识方案[J]. 密码学报, 2018, 5(5): 516-528.
GUO B Y, LI X Y. Multi-valued Byzantine consensus scheme with high transmission efficiency[J]. Journal of Cryptologic Research, 2018, 5(5): 516-528.
- [16] DWORK C, LYNCH N, STOCKMEYER L. Consensus in the presence of partial synchrony[J]. Journal of the ACM (JACM), 1988, 35(2): 288-323.
- [17] BRACHA G. Asynchronous Byzantine agreement protocols[J]. Information and Computation, 1987, 75(2): 130-143.
- [18] HENDRICKS J, GANGER G R, REITER M K. Verifying distributed erasure-coded data[C]//The 26th Annual ACM Symposium on Principles of Distributed Computing. 2007: 139-146.
- [19] DUAN S, REITER M K, ZHANG H. BEAT: asynchronous BFT made practical[C]//2018 ACM SIGSAC Conference on Computer and Communications Security. 2018: 2028-2041.
- [20] CACHIN C, TESSARO S. Asynchronous verifiable information dispersal[C]//24th IEEE Symposium on Reliable Distributed Systems (SRDS'05). 2005: 191-201.
- [21] REED I S, SOLOMON G. Polynomial codes over certain finite fields[J]. Journal of the Society for Industrial and Applied Mathematics, 1960, 8(2): 300-304.

[作者简介]



周旺(1995-), 男, 安徽淮南人, 中国科学技术大学硕士生, 主要研究方向为共识协议、区块链应用。



胡红钢(1978-), 男, 四川彭州人, 博士, 中国科学技术大学教授、博士生导师, 主要研究方向为密码学、网络安全。



俞能海(1964-), 男, 安徽无为, 博士, 中国科学技术大学教授、博士生导师, 主要研究方向为多媒体数据处理分析与检索、互联网信息检索、数字内容安全。