doi:10.11959/j.issn.1000-436x.2017261

基于树形拓扑结构的拜占庭容错系统设计

吕伟栋，周学广，袁志民
（海军工程大学信息安全系，湖北 武汉 430033）

摘 要：提出了一种能够应用于树形结构的拜占庭容错系统，并给出了基于深度与广度的拜占庭节点上限算法。通过对树进行群组划分，设计了一致性协议，保障了系统的安全性；通过联合签名机制决定节点的权限；通过视图转换协议将高层拜占庭节点置换到叶子节点，保证了系统的活性。在传统协议的基础上，该系统支持多任务同时执行，减少了节点间通信量，较大地降低了系统的空间复杂度，提高了系统的可行性，进一步缩小了与非拜占庭系统的差距。

关键词：拜占庭容错；树形拓扑结构；一致性协议；视图转换协议；检查点协议
中图分类号：TP302 **文献标识码**：A

Design of tree topology based Byzantine fault tolerance system

LYU Wei-dong, ZHOU Xue-guang, YUAN Zhi-min
(Department of Information Security, Naval University of Engineering, Wuhan 430033, China)

Abstract: A tree topology based Byzantine fault tolerance system was designed and the calculation of the Byzantine node bound based on depth and width was given. The consistency protocol was designed through divide the tree into groups by communication and work, which guaranteed the safety of system. The access of a node was determined with joint signature mechanism. The Byzantine nodes were placed to the leaves by view change protocol, which guaranteed the liveness of the system. On the foundation of traditional protocol, the new protocol support multiple tasks operating at the same time, and communication between nodes is largely decreased, as well as the space complexity, hence the feasibility being promoted, and the gap between Byzantine fault tolerance system and the non-Byzantine systems being narrowed.
Key words: Byzantine fault tolerance, tree topology, consistency protocol, view change protocol, checkpoint protocol

1 引言

自从第一台计算机于 1946 年问世，计算机系统已经在短短几十年内经历了单片机、独立 CPU、网络直到现在的超大规模网络的发展阶段。其应用也从最初的科学研究迅速渗透到了人类生活的方方面面。随着人们对于计算能力要求的不断提高，分布式系统也逐渐成为当前计算机技术的主流之一，如亚马逊 ASW、Drop Box 等。由于人们对于分布式服务的依赖性越来越强，分布式系统错误所造成的损失也不断增大。因此，保证系统提供不间断的正确服务成为当前分布式系统亟待解决的问题。分布式系统规模正在不断增大，系统所面临的风险也越来越大。安全漏洞、软件错误、不当操作等情况都对系统安全性、可靠性构成了严重威胁[1]。针对这些问题逐个进行保护成本过高，而且对于传统安全技术的简单叠加无法处理所有可能发生的意外，解决上述问题正在成为分布式系统的巨大挑战[2]。

目前正在获得广泛关注的区块链技术也面临着相似的问题[3]。中本聪提出了一种 PoW 共识机制，该技术的本质是通过引入分布式节点的算力竞

收稿日期：2017-10-07
基金项目：国家自然科学基金资助项目（No. 61672531）；海军工程大学科研自主立项课题基金资助项目（No. 20161606）
Foundation Items: The National Natural Science Foundation of China (No. 61672531), The Scientific Research Project of Naval University of Engineering (No. 20161606)

争来保证数据一致性和共识的安全性，在一定程度上解决了拜占庭将军问题，但是由于确认时间过长、资源浪费等问题广受诟病。在 PoW 机制的基础上，学者们又针对区块链技术设计了 PoS 机制、DPoS 机制等[4]。

随着数据重要性不断增强与硬件成本的不断降低，有学者提出了使用复制技术来解决可用性问题[5,6]，即通过算法协调一组复制服务器共同对应用提供服务。复制技术所提供给客户的是一个单一抽象的逻辑接口，而算法则保证了即使一部分服务器由于系统错误或网络攻击等原因停止工作时，客户仍然能够得到正确的服务。

拜占庭容错技术是应用复制技术解决分布式系统容错问题的通用方案，由 Lamport 在 1982 年提出[7]，并引起了学术界广泛关注与研究[8~12]，目前已经应用到 HDFS、Zookeeper 等广泛使用的分布式系统中[13]。其基本思想是运用一组服务器的互相通信来消除错误服务器对系统的影响。拜占庭系统的设计主要分为状态机拜占庭协议与 Quorum 拜占庭协议，前者需要给所有的请求分配序列号且请求必须按顺序执行，主要用于系统状态敏感的分布式计算系统中，后者不需要分配序列号且多个请求可以同时执行，主要用于分布式存储系统中。

从当前的研究现状看，拜占庭容错技术的主要研究方向是降低系统开销，缩小其与目前实际应用的系统之间的差距。传统的拜占庭算法不适用于树形拓扑结构[14]。

本文通过研究拜占庭协议的经典算法如 Zyzzyva、PBFT 等，分析树形网络对经典算法提出的挑战，基于状态机复制机制设计了一种能够应用于树形分布式网络的拜占庭算法，并给出了基于树的深度与广度产生的拜占庭服务器数量上限，经过证明，能够满足系统的安全性与活性。相比于应用于非树形网络的拜占庭协议，在网络规模较大时，本文算法在通信次数与时间上都有较大的优势，大大节省了通信时间、空间资源。

2 拜占庭系统模型

本文研究的应用对象是异步分布式系统，其中，节点通过广度大于或等于 3 的树形结构进行连接，应用本协议于系统中则称为拜占庭系统。网络可能无法通信、延迟通信、复制消息或乱序发送消息等。该系统的服务器可以视为集合 $N=\{N_d\}_{d=1}^{D}$，$D>1$，N_d 是深度为 d 的服务器的数量，D 是网络中叶子节点的深度，则网络中的服务器总量为 $n=\sum_{d=1}^{D}N_d$。用 $l_{d,a}$ 表示树中的任意一个节点，其中，d 为该节点的深度，a 为该节点在深度为 d 的节点集合中的序号，则 $l_{1,1}$ 是整个系统初始状态下的主节点。用 $G_{d,a}(d\leqslant D)$ 为节点 $l_{d,a}$ 与其所有的孩子所构成的群组，那么除了叶子节点与根节点外，任意其他节点都处在 2 个群组中，即系统中的每一个节点与其兄弟和双亲、与其孩子分别构成由双亲和自己作为局部主节点的群组中，前者称为高层群组，后者称为低层群组。

需要强调的是，本文假设网络中节点将转发与协商 2 个任务进行划分：在执行前者的过程中节点必须严格遵循树形拓扑，按照连接关系进行发送；而后者则是在群组内自由通信。本系统的节点群组内的协商机制与 PBFT 算法采用了同一种拜占庭节点数目上限，即在群组内节点的总数量 n' 与拜占庭节点的数量 f' 满足 $n'\geqslant 3f'+1$ 时，该群组始终可以达到一致性。

考虑到视图转换协议会使非拜占庭节点不断向高层调动，所以本文的网络要求树中的连接是可变的，即转发关系是可变的，实践证明这是可行的。

本文采用拜占庭错误模型，即假设错误节点可能的表现是任意的，只服从下文提出的限制：所有节点的错误都是独立、不相关的，这一点可以通过运行不同版本的应用程序、操作系统或由不同的管理员或密码来实现。

系统中还会应用密码机制来避免对手伪造、重放等攻击。各个节点之间流通的消息包括公钥签名、消息认证码以及消息摘要。本文中使用的全局变量如表 1 所示。这里遵循惯例，对消息摘要进行签名并将其附加到消息文本上。所有的节点都由系统中其他节点的公钥来认证签名。

对手可以组织各个错误节点进行诸如延迟、漏发等扰乱系统的行为，但是对手不能长期延误正确节点的工作。本文假设对手以及其控制的错误节点在计算能力上是有限的，无法破解上述密码体制，如对手无法伪造上文中提出的某正确节点的签名或某群组的集体签名，无法通过对摘要的计算得到消息，也无法得到 2 个具有相同摘要的消息。

表 1　　协议中的全局变量

变量	含义
D	叶子节点的深度
w	节点的度
N_d	深度为 d 的服务器的数量
N	树中节点的总数量
f	树中拜占庭节点的总数量
$l_{d,a}$	树中 $G_{d,a}(d \leqslant D)$ 的任意一个节点，d 表示该节点的深度，a 表示该节点同层节点集合中的序号
$l_{1,1}$	系统初始状态下的主节点
$G_{d,a}(d \leqslant D)$	其 $G_{d,a}(d \leqslant D)$ 所有的孩子所构成的群组
n'	群组中节点的数量
f'	群组中拜占庭节点的数量
$\langle M \rangle_{\sigma_{l_{d,a}}}$	节点 $l_{d,a}$ 发出的带有签名的消息 M
$\langle M \rangle_{\sigma_{G_{d,a}}}$	由节点 $l_{d,a}$ 发出的带有群组 $G_{d,a}$ 中 $f'+1$ 个节点联合签名的消息 M
$D(m)$	消息 m 的摘要

此外，这里需要假设所有的复制都满足确定性，且运行时处于相同状态。

3　协议的目标与面临的挑战

3.1　协议的目标

拜占庭算法的总体目标是保证系统在面临一部分节点出错的情况下能够保持一致性并正常工作。本协议在第2节中的假设被满足且系统中拜占庭服务器数目f小于上限的基础上，可以满足 2 个关键指标：安全性与活性。Lamport[15]在 1977 第一次描述了安全性与活性，安全性指系统在有限的时间内不会出现问题，活性指系统在有限的时间内会达到一致。本文中系统的安全性是指系统一直保持一致性并不断针对任务进行原子操作，但是，仅依靠安全性不能彻底消除拜占庭节点的损害，因为不加限制的拜占庭攻击会对任务本身进行影响，如任务是计算后对文件进行写入，而拜占庭节点可以通过向文件写入无意义的内容导致任务失败，针对这种特定情况，协议将读写等敏感操作的权限交给根节点。

本文的活性则是指任务在限定的拜占庭节点数目的前提下一定会在有限的时间内得出一致性结果，即算法会在规定时间内返回有效值。

3.2　协议面临的挑战

显然，拜占庭服务器所处的层次越高，对系统的威胁越大。本协议的基本思想是通过各个群组的内部通信限制高层节点的危险行为并通过各群组不断激活视图转换协议使高层次的拜占庭节点逐步调换到底层；在执行一定次数后，协议的安全性是一定可以达到的，并且拜占庭节点一定会处于最底层。但是如此逐层进行的视图转换带来了巨大的时间成本，所以协议选择在任务进行到低层时加入新任务来保证系统的效率性。

与传统的拜占庭协议的去中心化思想不同，树形结构本身就带有中心，即叶子节点一定会受到根节点的影响，节点的层级越高，对系统的影响越大。这就成为协议面临的 2 个挑战。

1) 多 $G_{d,a}(d \leqslant D)$ 项任务同时进行所带来的复杂任务调度，如何制定任务完成的条件且制定相应的满足多任务的一致性、视图转换协议以及低成本的检查点协议。

2) 在 $G_{1,1}$ 群组成员全部出错的情况下如何激活系统，在这种情况下，低层节点接收不到任何消息或请求。

4　拜占庭节点上限算法

不同于其他拓扑结构的分布式系统，树形网络的节点总数与深度、广度成函数关系，且通信线路只能沿着树进行，这使拜占庭节点上限与传统协议的 $f < \dfrac{N-1}{3}$ 完全不同，其中，N 为系统中的节点总数，f 为系统的拜占庭节点数目。本协议的拜占庭节点上限与节点总数以及树的度有关。

为了方便进行阐述，本文采用所有非叶子节点度都为 w 且叶子深度为 D 的树进行分析，其中 $w \geqslant 3$ 且 $d > 1$，那么系统内节点共有 $N = \dfrac{w^d - 1}{w - 1}$。

本文中节点在群组的协商机制与 PBFT 算法采用同一拜占庭节点上限，即 $n' \geqslant 3f' + 1$。显然，如果一个群组无法达成一致，该群组的局部主节点即使正常也无法工作，系统就可以视为增加了拜占庭节点。

本系统中拜占庭节点上限f与节点总数 N 的关系，与每个节点的度、拜占庭所处位置有关，传统的 $N \geqslant 3f + 1$ 的关系显然是不成立的。在下文中会进行介绍，如果高层存在拜占庭节点，那么系统会依照协议通过视图转换协议将拜占庭节点全部调整到叶子节点，所以这里不妨假设所有拜占庭节点都是叶子节点并处在 D 层，第 D 层 w^{D-2} 个群组，在不影响系统一致性的情况下，每个群组内最多只能有 $\left[\dfrac{w}{3}\right]$ 个拜占庭节点，$\left[\dfrac{w}{3}\right]$ 为 $\dfrac{w}{3}$ 向下取整的结

果。所以 $f \leqslant w^{D-2} \times \left\lceil \frac{w}{3} \right\rceil$，最终可得 $f \leqslant \frac{(w-1)N+1}{3w}$，具体推导过程如下所示。

$$f \leqslant d^{w-2}\left\lceil \frac{w}{3} \right\rceil \leqslant w^{D-2}\frac{w}{3} = \frac{w^{D-1}}{3}$$

$$f \leqslant \frac{w^{D-1}}{3} = \frac{(w^d - 1 + 1)(w-1)}{3w(w-1)} = \frac{(w-1)N+1}{3w}$$

即 $f \leqslant \frac{(w-1)N+1}{3w}$，其中，$w$ 为树的度，N 为树的总节点数。

该条件仍适用非对称的树结构，但是需要满足 w 是树中节点的最小度且 $w \geqslant 3$，$d > 1$。

5 协议算法设计

本文针对树形拓扑结构的分布式系统提出了一种状态机复制协议，与传统的拜占庭协议相似，本协议也由 3 个子协议组成：一致性协议、视图转换协议以及检查点协议。一致性协议组织各个群组执行请求，视图转换协议在当前的局部主节点发生错误时协调新的局部主节点，检查点协议使每个节点保存足够的状态并尽可能降低执行视图转换协议的成本。

下面，对 3 个协议进行分别介绍。

5.1 一致性协议

由于树形结构的层次性，系统无法使每个正确节点的结果都能反馈到客户，协议的安全性关键在于保证一个非拜占庭节点的根节点能够产生满足条件的 $\langle REPLY, v, t, c, r \rangle_{\sigma_{G_{1,1}}}$，即获得足够的联合签名来保证自己的结果被接受。图 1 和图 2 分别为度为 3 的二层和三层树的一致性协议的基本流动。

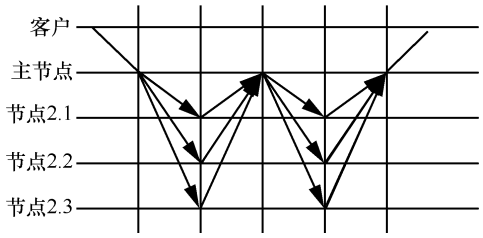

图 1 度为 3 的二层树协议执行过程

由图 1 和图 2 可以得出树形结构的深度 D 与系统执行每次任务的通信轮数 r 满足 $r = 4D - 2$，其中，$D \geqslant 2$，显然 D 与 r 满足线性关系。但是与传统的 PBFT 算法不同，系统能同时执行最多 D 个任务，所以系统的整体效率是基本不变的。

上文中已经提到，树形结构被分为若干个群组，所以任务的完成实际上是协议在各个群组内部的协调，群组的工作最终保证了目标的实现。为了详细描述协议内容，下面将对一条请求的执行过程进行解释。

1) 向主节点发送请求

客户端 c 请求系统执行操作 o，向根节点发送消息 $m = \langle REQUEST, o, t, c \rangle_{\sigma_c}$，其中，$t$ 为该请求的时间戳，保证该请求语义的时效性。

此时客户针对操作 o 设置 2 个计时器 $Timer_s$，$Timer_c$ 并开始计时，前者用于收集主节点自证消息的返回时间，后者用于收集主节点操作结果的返回时间。

2) 主节点收到请求，向客户发送自证消息

当主节点接收到请求 $\langle REQUEST, o, t, c \rangle_{\sigma_c}$ 后，会

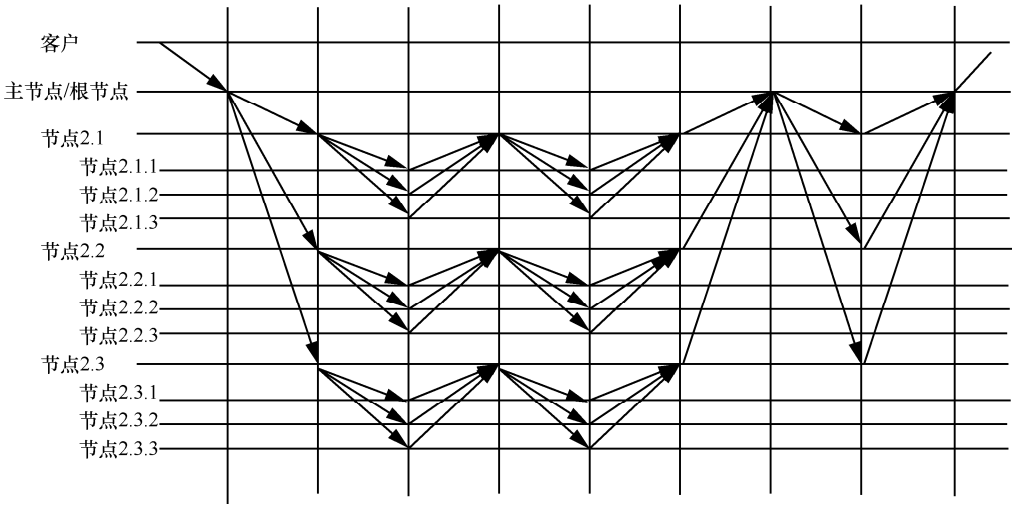

图 2 度为 3 的三层树协议执行过程

立刻向客户发送自证消息$\langle PRE-REPLY,v,t,c,\rangle_{\sigma_{l_{1,1}}}$以证明自己是非拜占庭节点。

①客户在$Timer_s$的可允许范围内收到自证消息。

客户收到自证消息后，对其签名进行验证。停止时钟$Timer_s$，每隔 τ 向根节点发送一个新的请求（ τ 为非拜占庭节点从开始收集联合签名到开始为双亲签名的平均时间间隔）。

②客户在$Timer_s$可允许范围内未收到自证消息。

向主节点的序列号最低的孩子节点发起视图转换协议，等待$Timer_s$计时 *Tworst* 后再次发送请求(*Tworst* 为系统在最差情况下整体执行视图转换协议所需时间)。

3) 主节点为请求分配序列号并群发给孩子，然后执行操作并开始请求联合签名

主节点在视图 v 下为请求分配序列号 n 并群发$\left\langle\langle ORDER-REQ,v,n,d,var\rangle_{\sigma_{l_{1,1}}},m\right\rangle$到所有的孩子，其中，$v$ 为发送时群组$G_{1,1}$的视图号，n 为请求的序列号，$d=H(m)$为消息 m 的摘要，*var* 为执行操作 o 的参数集。

然后执行操作得出结果 r，并开始向孩子发送联合签名请求，内容为

$$\left\langle\langle JOINT-SIG,v,n,d,r\rangle_{\sigma_{l_{1,1}}},m\right\rangle$$

只有在$t>t_c$时主节点才会接受请求并将请求加入其日志中，其中，t_c为主节点之前所接收过的最大时间戳。

①节点$l_{i,j}$接收到请求后，群发给孩子并执行操作，然后开始请求联合签名。

节点$l_{i,j}$接收到请求$\left\langle\langle ORDER-REQ,v,n,d,var\rangle_{\sigma_f},m\right\rangle$后（$\sigma_f$为节点$l_{i,j}$的双亲的签名，$v$为节点$l_{i,j}$所在的高层群组视图号），首先验证 m 是否为一个完整的请求消息，d 是否为 m 的正确摘要，$n=n_{\max}+1$是否成立（$n_{\max}$为节点日志中最大的序列号），v 是否与自己所在的高层群组视图号相同。如果上述条件满足，则接受该请求并将其加入日志中。

如果节点$l_{i,j}$不是叶子，则向孩子发送$\left\langle\langle ORDER-REQ,v,n,d,var\rangle_{\sigma_{l_{i,j}}},m\right\rangle$，然后执行操作得出结果 r，开始不断向完成联合签名收集任务的孩子群发联合签名请求$\left\langle\langle JOINT-SIG-REQ,v,n,d,r\rangle_{\sigma_{l_{i,j}}},m\right\rangle$，直到收集到$f'+1$个节点的联合签名，其中，$v$为$l_{i,j}$的低层群组视图号，收集到$f'+1$个节点的联合签名后告知双亲，消息形式为$\langle JOINT-SIG-COM,v,n,d,r\rangle_{\sigma_{G_{i,j}}}$。

如果节点$l_{i,j}$是叶子，则直接执行操作得出结果 r 并进入下一步。

②节点$l_{i,j}$等待了时间T_d未接收到请求。

如果节点等待了时间T_d一直未收到请求（T_d为节点所在层的等待时间上限），那么向双亲发送测试消息$\langle TEST-MES,v\rangle_{\sigma_{l_{i,j}}}$，其中，$v$ 为节点所在的高层群组视图号；如果双亲答复了$\langle TEST-MES,v\rangle_{\sigma_f}$，那么再次重置计时器；否则激活视图转换协议。

4) 节点$l_{i,j}$接收到联合签名请求

节点$l_{i,j}$收到联合签名请求$\left\langle\langle JOINT-SIG-REQ,v,n,d,r\rangle_{\sigma_f},m\right\rangle$后（$\sigma_f$为节点$l_{i,j}$的双亲的签名），首先验证 m 是否是一个完整的请求消息，d 是否是 m 的正确摘要，$n=n_{\max}+1$是否成立（$n_{\max}$为节点日志的最大的序列号），v 是否与自己保存的高层群组的视图号相同。如果上述条件满足，则开始下一步判定。

节点$l_{i,j}$比对 r 是否与自己所得结果相同，如果相同则执行签名并将该请求加入日志，如果不同则拒绝签名，如果一个非拜占庭叶子节点连续拒绝其双亲 2 次，则启用视图转换协议。

5) 主节点向客户返回操作结果

主节点在收集到$f'+1$个孩子的联合签名后，将其加入日志，然后向客户发送回复信息$\langle REPLY,v,t,c,r\rangle_{\sigma_{G_{1,1}}}$客户对签名进行认证正确后，接受该结果 r。

5.2 检查点协议

在 PBFT 协议中，节点需要在收集到至少 f+1 个非拜占庭节点的结果之后才能将存储的运算结果删除。在本协议中，一旦节点执行了双亲发出的对 n 号请求的联合签名，那么它就会将自己的 n 号请求运算结果删除，主节点在客户接受了自己的回复后也会将自己的 n 号请求运算结果删除。

如果拜占庭节点是局部主节点（根节点也可以视作$G_{1,1}$的局部主节点），需要通过视图转换协议选出新的局部主节点，如何提供证明一个节点为非拜

占庭节点的证据成为检查点协议需要解决的首要问题。此外，如果节点丢失了部分消息，也需要通过向其发送全部或部分服务状态来为其更新状态，这些状态由检查点提供。

在传统 PBFT、zyzzyva 等算法中，节点会收集 *2f*+1 个检查点消息作为证据，由于收集证据的计算成本太高，所以会在每执行一定数量（如 100 次）的请求后进行证据计算，由证据计算产生的系统状态称作检查点。

考虑到系统中流转的请求一般不止一个，对不同层次的节点要求统一请求序列号会导致系统混乱，但是同一层次的节点的执行序列号是相同的，所以本算法只要求节点与其兄弟进行统一。

检查点生成的具体方法如下：当一个节点 $l_{i,j}$ 在任务 n 期间收集到 $f'+1$ 个节点的联合签名后，向兄弟节点发送 $\left\langle \langle CHECKPOINT, n', d, i, j\rangle_{\sigma_{l_{i,j}}}, \langle JOINT-SIG, v, n\rangle_{\sigma_{G_{i,j}}} \right\rangle$ 其中，n' 为上一个任务的序列号，d 为当前状态的摘要，v 为当前的视图号。每个收到消息的节点都将其记入日志，$\langle JOINT-SIG, v, n\rangle_{\sigma_{G_{i,j}}}$ 就是节点正确的证据，协议通过孩子的联合签名证明双亲的正确性。生成检查点后，节点将日志中所有序列号小于 n 的检查点全部清除。

检查点协议主要用于视图转换时为选取局部主节点提供依据，并且保持节点日志中的数据有序、有界。

5.3 视图转换协议

视图转换协议保证了系统在主节点出错的情况下仍具有活性。当节点等待时间超过预设值时，协议将触发以避免节点陷入无限等待。

所有的视图转换协议都是在群组内进行的，所以本文中讨论在群组 $G_{i,j}$ 内的执行过程。

如果一个节点 l_{i+1,k_1}（$G_{i,j}$ 为节点 l_{i+1,k_1} 的高层群组，即局部主节点为 l_{i+1,k_1} 的双亲 $l_{i,j}$）拒绝双亲 $l_{i,j}$ 的联合签名请求 2 次或在视图 v 中的等待时间超过 T_d 且未回复节点的测试消息，那么开始请求向视图 v+1 变换。节点开始只接收视图变换、视图选举、新视图、认同和检查点消息，并向其兄弟群发

$$\langle\langle VIEW-CHANGE, v+1, n, C, P\rangle, \langle VIEW-ELET, v+1, n, C\rangle\rangle_{\sigma_{l_{i+1,k_1}}}$$

其中，n 为节点 $l_{i,j}$ 的检查点号，C 为节点 $l_{i,j}$ 的证据，P 为节点 $l_{i,j}$ 中序列号大于 n 的请求集合。

如果收到消息的节点 l_{i+1,k_2} 验证签名正确，则接受消息并向兄弟群发消息 $\langle VIEW-ELECT, v+1, n, C\rangle_{\sigma_{l_{i+1,k_2}}}$ 群组内成员在收到 $f'+1$ 条后，根据收到的选举消息，认同 C 的序号最大且 j（节点在层次内的序号）最小的节点为视图 v+1 的局部主节点。

选举完成后，新的局部主节点 $l_{i+1,a}$ 向兄弟群发消息 $\langle NEW-VIEW, v+1, (i+1, a)\rangle_{\sigma_{l_{i+1,a}}}$，向前任局部主节点所在的高层群组成员发送消息 $\langle NEW-VIEW-UP, (i+1, a)\rangle_{\sigma_{l_{i+1,a}}}$，向孩子发送消息 $\langle NEW-VIEW-DOWN, (i, j)\rangle_{\sigma_{l_{i+1,a}}}$。

所有接收到消息的节点根据消息中的坐标位置建立新的通信关系。

经过视图转换协议 $l_{i+1,a}$ 与 $l_{i,j}$ 变换了位置和关系，收到 *NEW-VIEW-UP* 的节点接纳新的 $l_{i,j}$ 节点，且其在群组中的地位不变，然后向新 $l_{i,j}$ 节点发送消息 $\langle ACCEPT, v, p\rangle_{\sigma_l}$，其中，$v$ 为该群组的当前视图号，p 为节点中序列号大于 n 的请求集合，σ_l 为各节点的签名。收到 *NEW-VIEW-DOWN* 消息的节点接纳新的 $l_{i+1,a}$ 节点且其在群的地位不变，然后向新 $l_{i+1,a}$ 节点发送消息 $\langle ACCEPT, v\rangle_{\sigma_l}$，其中，$v$ 为该群组的当前视图号，σ_l 为各节点的签名。由于在系统中拜占庭节点会被降到低层群组中，正确节点会被升至高层群组，所以在 *NEW-VIEW-UP* 中会给新节点更新任务序列，而 *NEW-VIEW-DOWN* 中没有该消息。

通过视图转换协议系统不断将拜占庭节点向底层调动以防止其影响系统运行。

6 性能分析

本节针对协议的性能进行分析，证明其具有安全性、活性，并对其时间、空间成本进行分析比较。

6.1 安全性证明

本系统的安全性是指系统一直保持一致性并不断针对某任务进行原子操作。

1) 多任务下的一致性挑战

为了提高吞吐量，协议允许多任务同时在系统中运行。此时的一致性体现在运行中的群组中，即同一群组中的节点有相同的视图号与任务序列号，

为保证这一点，视图转换协议在执行时会为正确节点更新任务序列以保证群组内的一致性。除了根节点与叶子节点外，每个节点都处在 2 个群组中，当其工作于高层群组时主要考虑任务序列的下发，对应于未执行的任务序列，工作于低层群组时主要考虑当前任务与上个任务的序列号，对应于已执行的检查点。通过群组的一致性实现多任务条件下的一致性。

2) 多种拜占庭节点位置下的一致性挑战

假设一个非拜占庭节点的双亲为拜占庭节点，那么它无法得到来自该节点的联合签名，根据系统整体假设，树中拜占庭节点的数目有数量限制，所以即使拜占庭节点处在高层，也无法将错误信息继续向高层扩散。

假设一个非拜占庭节点的孩子为拜占庭节点，那么由于孩子无法收集足够数目节点的联合签名，所以无法将错误向高层扩散。

如果一个群组内的拜占庭节点数目大于 f' 个，由于对手相互独立，所以这个群组会停止工作，无法对一致性造成影响。而长时间停止工作则会触发视角转换协议，最终将低层的正确节点加入本群组并将错误节点换入低层。

综合上述讨论，协议保证了系统的安全性。

6.2 活性证明

为了提供活性，节点需要在一段时间内没有请求的情况下触发视图转换协议。

在最坏情况下，所有的拜占庭节点都集中在 $G_{1,1}$ 群组内使客户的请求无法下达或系统中有足够的拜占庭节点使系统无法工作，系统会在有限的等待时间内触发视图转换协议，并通过有限次的迭代将所有的拜占庭节点调动到叶子的位置。

视图转换协议实际上可以视为一个可以在各群组内执行的可迭代的函数，具体操作是由可自证的子节点发起，将不作为的双亲节点替换并向兄弟节点发出声明，结果是原低层群组内正常的子节点成为双亲节点并成为高层群组的子节点。其实质是将低层的正常节点转换到高层，所以在最不理想的情况下，每个拜占庭节点都经历 $D-1$ 次视图转换协议并进入最底层，很明显这是个有界值，所以活性是一定可以保证的。

6.3 可用性分析

拜占庭算法的可用性主要取决于算法能否将开销降低到现实可接受范围内。下面就其可用性方面与经典算法如 PBFT、Zyzzyva 等算法进行比较。

1) 拜占庭节点上限与系统规模对比

假设拜占庭节点数目为 f，经典算法如 PBFT、Zyzzyva、HQ 等算法对系统规模的要求最小为 $3f+1$，最大为 $5f+1$，本算法要求为

$$\frac{3wf-1}{w-1}$$

其中，w 为树中节点最小度。显然在 w 较大时系统规模可视为 $3f$，与经典算法中的最小要求相符。

2) 空间复杂度对比

由于树形结构中节点间无法进行大规模通信，所以本协议将协商范围缩小并按层次执行，其实质是牺牲时间成本来交换空间优势。

本文提出的算法空间复杂度由节点所构成的树的结构决定，以 D 层 w 度的树为例，为方便演示，假设每个非叶子节点都有 w 个孩子，则该树共有 $N=\frac{w^D-1}{w-1}$ 个节点，对于同等规模的 PBFT 拜占庭系统，执行一次任务需要进行 $\mathrm{C}_N^2\cdot T$ 次交互，T 为单个节点执行一次任务需要交互的轮数，本系统则需要执行 $4(N-1)(w-1)(2D-1)$ 次计算，在节点数量足够的情况下，本算法比传统算法在空间复杂度上降低了一个数量级。

本系统在单任务的时间复杂度上不具备优势，但是与传统算法不同的是系统可以同时执行 d 个任务，所以在吞吐量上仍有较好的表现，直接与传统算法比较瓶颈服务器计算量显然是无意义的。表 2 比较了不同拜占庭协议的参数。

表 2　本系统与不同拜占庭系统各参数比较

算法	服务器数量	单一请求所需阶段数	单一请求所需计算数
本文算法	$\frac{3wf-1}{w-1}$	$4d-2$	$4(N-1)(d-1)(2w-1)$
PBFT	$3f+1$	4	$2N(N-1)$
HQ	$3f+1$	4	$2N(N-1)$
Zyzzyva	$3f+1$	3	$\frac{3N(N-1)}{2}$

7 结束语

已有的拜占庭系统都无法应用于树形结构的分布式系统，并且占用大量计算与存储资源。本文提出了一种适用于树形结构的拜占庭系统，对其协议进行了详细介绍与分析，并在理论上对其进行了

分析与比较，证明了本系统满足安全性与活性，并分析了其可用性。

经过比较，本系统在空间复杂度上具有较大优势，但是与 Zyzzyva 协议相比，本协议在 MAC 计算量上较为复杂，影响了系统的整体表现。下一步工作将针对降低 MAC 计算量进行展开，以提高算法的效率。

参考文献：

[1] SCHROEDER B, GIBSON G A. A large-scale study of failures in high-performance computing systems[C]//IEEE Trans on Dependable and Secure Computing. 2010: 337-350.

[2] 范捷, 易乐天, 舒继武. 拜占庭系统技术研究综述[J]. 软件学报, 2013,24(6):1346-1360.
FAN J, YI L T, SHU J W. Research on the technology of Byzantine system[J]. Journal of Software, 2013, 24(60):1346-1360.

[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):481-493.
YUAN Y, WANG F Y. Blockchain: the state of the art and future trends[J]. Acta Automatic Sinica, 2016,42(4):481-494.

[4] 夏清, 张凤军, 左春. 加密数字货币系统共识机制综述[J]. 计算机系统应用, 2017,26(4):1-8.
XIA Q, ZHANG F J, ZUO C. Review for consensus mechanism of cryptocurrency system[J]. Computer Science & Application. 2017, 26(4):1-8.

[5] LISKOV B, GHEMAWAT S, GRUBER R, et al. Replication in the harp file system[C]//13th ACM Symp on Operating System Principles(SOSP). 1991:226-238.

[6] OKI B, LISKOV B. Viewstamped replication: a new primary copy method to support highly-available distributed systems[C]//ACM Symposium on Principles of Distributed Computing. 1988: 8-17.

[7] LAMPORT L, SHOSTAK R, PEASE M. The Byzantine generals problem[C]//ACM Trans. on Programming Languages and Systems, 1982, 4(3):382-401.

[8] CASTRO M, LISKOV B. Practical Byzantine fault tolerance and proactive recovery[C]//ACM Trans on Computer Systems. 2002: 398-461

[9] WOOD T, SINGH R, VENKATARAMANI A, et al. ZZ and the art of practical BFT execution[C]//6th Conf on Computer Systems. 2011:123-138.

[10] HENDRICKS J, SINNAMOHIDEEN S, GANGER G R, et al. Zzyzx: scalable fault tolerance through Byzantine locking[C]//2010 IEEE/IFIP Int'1 Conf on Dependable Systems and Networks. 2010: 363-372.

[11] SERAFINI M, NOKOR P, DOBRE D, et al. Scrooge: Reducing the costs of fast Byzantine replication in presence of unresponsive replicas[C]//2010 IEEE/IFIP Int'1 Conf on Dependable Systems and Networks. 2010: 353-362.

[12] RAMAKRISHNA K, LORENZO A, MIKE D, et al. Zyzzyva: speculative byzantine fault tolerance[C]//ACM SIGOPS Symposium on Operating Systems Principles. 2007:45-48.

[13] CLEMENT A, KAPRITSOS M, LEE S, et al. Upright cluster services[C]//ACM SIGOPS 22nd Symp on Operating Systems Principles. 2009:277-290.

[14] DANNY D. The Byzantine generals strike again[J]. Journal of Algorithms, 1982, 3(1): 14-30.

[15] LAMPORT L. Proving the correctness of multiprocess programs[C]// IEEE Transactions on Software Engineering, 1977, SE-3(2):125-143.

作者简介：

吕伟栋（1993-），男，湖北襄阳人，海军工程大学硕士生，主要研究方向为信息系统与决策支持。

周学广（1966-），男，江苏高邮人，博士，海军工程大学教授、博士生导师，主要研究方向为信息安全与密码学。

袁志民（1964-），男，河南淮阳人，博士，海军工程大学讲师，主要研究方向为通信安全。