

硕 士 研 究 生 读 书 报 告



题目 PBFT读书报告

作者姓名 张浩

作者学号 22051152

指导教师 李启雷

学科专业 电子信息

所在学院 软件学院

**摘 要**

拜占庭将军问题（Byzantine Generals Problem）是由Leslie Lamport在同名论文[1]中提出的分布式对等网络中的通信容错问题，拜占庭将军问题被认为是容错性问题中最难的问题类型之一。

在PBFT（Practical Byzantine Fault Tolerance）之前，大部分BFT算法缺乏实践可行性。PBFT则提出了一个实用的算法，将算法复杂度从指数级降低到多项式级，使得拜占庭容错在实际系统中使用称为可能。

**关键词**：拜占庭问题，分布式一致性，PBFT

**第一章 引言**

随着软件行业的不断发展，恶意攻击和软件错误会持续增多，所以拜占庭问题的解决算法会愈发重要。PBFT（Practical Byzantine Fault Tolerance[2]）论文描述了一个新的replication算法来解决拜占庭问题，作者认为之前的BFT算法过于沉迷理论证明而忽视了算法可行性，所以提出了实践可用的拜占庭容错算法PBFT：它可以在异步环境下正常工作，且把算法复杂度从指数级降低到了多项式级。

本文是对PBFT的读书报告。第二章简要介绍PBFT的算法特点；第三章着眼于PBFT的算法模型，介绍算法细节；第四章进行回顾总结。

**第二章 PBFT算法特点**

论文第一章直接表明，PBFT虽然是一个可以实践的算法，但是需要一定的前提条件作为约束：

* 节点总数为3f+1的情况下，最大容许f个恶意节点
* 恶意节点只能以自身名义发消息，无法伪造其他节点的消息（可以通过签名算法来保证，每个节点都知道其他节点的公钥，用来验证发送者身份）
* 部分逻辑采用类似同步的通信机制保证算法正确性：比如超时的情况下，client通过重发机制来保证系统的可用性。

在此前提下，实用PBFT算法可以保证整个分布式系统的安全性（safety）和可用性（liveness）。

其中，安全性（safety）意味着整个系统满足线性一致性（linearizability），也可以理解为：即使在存在作恶节点的情况下，其他节点访问系统的state得到的依然是统一一致的结果，作恶节点的行为不会导致整个系统的state错乱。可用性（liveness）意味着client最终都能收到它们发出去的requests的replies，也就是上面提到的同步通信，不过这个基于一个前提：即网络延迟不会无限期增长，否则就无法模拟同步通信。

**第三章 PBFT算法模型**

假设一个分布式网络中存在着标号为{0,…,|R|-1}的共R个节点，且|R| = 3f + 1，其中f为作恶节点的个数。网络中的节点一种有两种角色：primary和replica，同一时间只会存在一个primary，其余节点则为replica。

此外，算法还定义了一个属性view，view可以看作是一个连续增长的数字，每条消息都会携带当前的view值，当primary超时或者被其余节点发现为作恶节点时，会发生view change，切换成功后view值加一。而且算法规定，当前的primary由view的值来确定，规则是p = v mod |R|，即当前的view值对R取模，结果对应的节点就是当前的primary。

**3.1 算法的简要流程**

首先描述一下算法的一个粗浅流程：

1. client向primary发送一个requesst
2. primary将这个request广播给其余的replicas
3. replicas执行这个request并且将reply直接回复给client
4. client等待replies，如果收到f + 1个结果一致的reply的话，那么client就会认可这个结果作为本次request的结果。

request的数据格式为：<request, o, t, c>，其中request表示消息类型，o表示这个request的具体操作，t是该条消息的时间戳，c表示client的身份id。

reply的数据格式为：<reply, v, t, c, i, r>，其中reply和request作用相同，v表示当前的view number，t和c与request中的含义一致，i表示replica的身份id，r就是回复的结果。

**3.2 f+1**

下面解释一下为什么client收到f+1个结果一致的reply就可以确认结果。

PBFT算法整体看起来是一个三阶段提交的过程，primary收到reuqest之后就开始了三阶段提交的流程，如图3.1所示。

primary在收到request之后，讲一个pre-prepare消息广播给所有replicas，pre-prepare的消息格式为：<<pre-prepare, v, n, d>p’, m>，其中n是给消息分配的一个唯一编号sequence number，d是m的摘要。值得注意的是，pre-prepare消息并没有携带request的具体内容，通过减少三阶段提交中消息的大小来提高了通信效率。

replica收到pre-prepare消息后，必须满足以下条件才会接受该消息：

1. 验证消息是否正确：pre-prepare格式是否正确，m的摘要是否正确
2. 该消息的view值是本节点当前的v
3. 节点目前还未收到view值是v且sequence number也是n但是d不一样的pre-prepare
4. sequence number在最大值H和最小值h之间（防止primary作恶耗尽sequence number）

如果replica接受了该pre-prepare后进入prepare阶段，它就会对外广播消息prepare(m, v, n, i)。同时当一个replica收到了2f+1个与自己pre-prepare相匹配的prepare消息时，该节点认为消息达成共识，进入commit阶段。

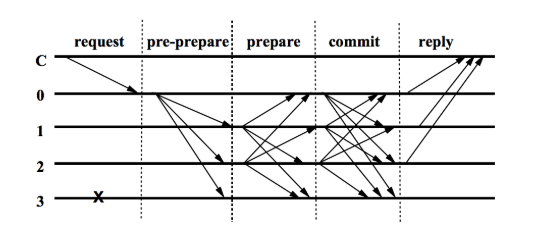


图3.1 PBFT三阶段提交

为什么2f+1个prepare就可以进入commit阶段？这里用反证法证明，在一个系统中，同时最多只能有一个消息收到2f+1个一致的prepare。假设存在两个消息m和m’，都在prepare阶段达到了2f+1个一致响应，那么

即，最少存在着f+1个节点是作恶节点，这违背了系统中至多只有f个作恶节点的假设，所以同时只能有一个正确的消息收到2f+1个prepare确认。

节点进入commit阶段后会广播commit(m, v, n, i)消息，当某个节点收到了2f+1个匹配的commit消息后，意味着除了自己，已经至少有2f个节点认可了这个request，可以进行commit，返回reply给client。

为什么client收到f+1个一致的reply就可以确认结果？client收到了f+1个reply，意味着其中至少有一个是正常节点返回的，而经过prepare和commit阶段，系统可以保证一个正常节点的reply是系统中大多数节点的共识，所以，client只要拿到f+1个一致的reply就能保证拿到了大多数正常节点的意见，从而可以确认结果。

**3.3 checkpoint**

PBFT为了保证可靠性，三阶段中所有认可的消息都需要持久化到log中，显然log会越来越大，通过checkpoint对log进行了裁剪。按照指定规则（如没处理100个request），每个节点都会定时生产checkpoint(n, d, i)。

节点每生成好一个checkpoint都会把其广播出去。和上文类似，当一个节点收到2f+1个一致的checkpoint，说明这个checkpoint在系统内达成了共识，即标记该checkpoint为一个stable checkpoint。然后将之前的所有checkpoint和对应的log删除。

**3.4 view change**

view change是PBFT中的关键设计，保证了整个系统的safety和liveness。

当primary节点失效或者作恶时，对于一个request，系统显然是无法达成共识的。当client检测到超时，它会把这个request直接广播给所有的replicas，如果仍然无法共识，检测到超时的节点就会发送view change消息，进入view change流程。

view change的数据格式为：<view-change, v+1, n, C, P, i>，其中v是当前的view值；n是replica i当前stable checkpoint的sequence number；C是验证当前stable checkpoint正确性的2f+1个消息集合；P是一个Pm组成的集合，m表示消息，Pm表示序号为m的消息达成prepared使用的消息集合，Pm内包含关于消息m的1个pre-prepare消息和2f个prepare消息。

当新的primary收到了2f+1（包含自己的）个v+1的view change消息时，它会广播一个消息<new-view, v+1, V, O>。其中V表示验证view change生效的2f+1条view change消息集合；O是pre-prepare消息的集合，具体规则如下：新的primary对于所有的checkpoint进行扫描，取出最小的sequence number min-s和最大的sequence number max-s；对介于min-s和max-s之间的每一个sequence number n，创建一个pre-prepare消息加入O集合；如果P集合中存在一个Pm的sequence number为n，就创建<pre-prepare, v+1, n, d>；如果不存在的话，就创建<pre-prepare, v+1, n, d\_null>。

replica收到primary的new view消息后会用同样的方法验证这个消息的正确性，更新view到v+1，然后为min-s到max-s之前的request广播prepare消息。这样是为了确保，在发生view change前，如果已经有消息m被分配了序号n且达到了prepared状态，那么在view change后，要保持这个状态。

**第四章 总结**

**4.1 优点**

* 通信复杂度O(n2)
* 首次提出在异步网络环境下使用状态机副本复制协议，该算法可以工作在异步环境中，并且通过优化在早期算法的基础上把响应性能提升了一个数量级以上
* 使用了加密技术来防止欺骗攻击和重播攻击，以及检测被破坏的消息。消息包含了公钥签名、消息验证编码（MAC）和无碰撞哈希函数生成的消息摘要（message digest）
* 适用于permissioned systems (联盟链/私有链)，能容纳故障节点，也能容纳作恶节点。要求所有节点数量至少为3f+1（f为作恶/故障不回应节点的数量），这样才能保证在异步系统中提供安全性和活性
* 解决了原始拜占庭容错算法效率不高的问题，将算法复杂度由指数级降低到多项式级，使得拜占庭容错算法在实际系统应用中变得可行

**4.2 缺点**

* 仅仅适用于permissioned systems (联盟链/私有链)
* 通信复杂度过高，可拓展性比较低，一般的系统在达到100左右的节点个数时，性能下降非常快
* PBFT在网络不稳定的情况下延迟很高

**参考文献**

1. Lamport L, Shostak R, Pease M. The Byzantine generals problem[M]
2. Castro M, Liskov B. Practical byzantine fault tolerance[C]//OSDI. 1999, 99(1999): 173-186.