浙江大學

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。

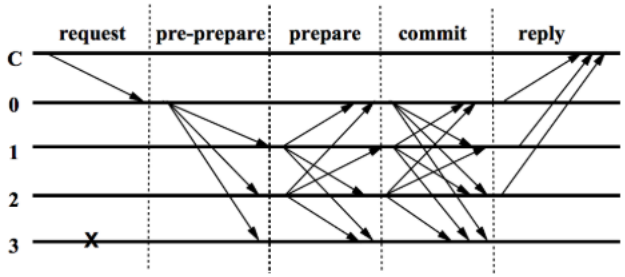

图 3.1 PBFT 三阶段提交

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

$$2\,(2f+1) - \,(3f+1) = f+1$$

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 P_m 组成的集合，m 表示消息，P_m 表示序号为 m 的消息达成 prepared 使用的消息集合，P_m 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 P_m 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 $O(n^2)$
- 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.