

# paxos algorithm

## 问题描述

假设有一组可以提出提案的进程集合，那么对于一个一致性算法来说需要保证以下几点：

在这些被提出的提案中，只有一个会被选定。

如果没有提案被提出，那么就不会有被选定的提案。

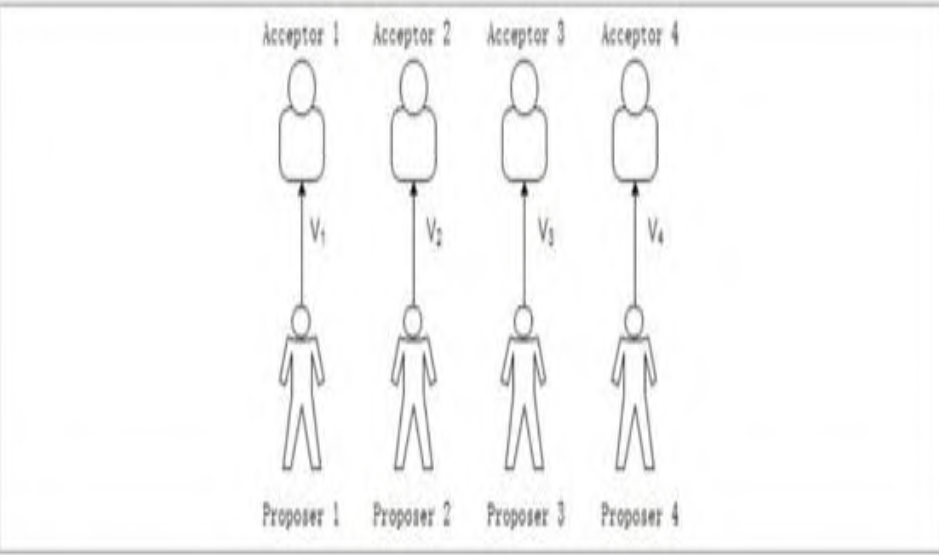
当一个提案被选定后，进程应该可以获取被选定的提案信息。

## paxos basis

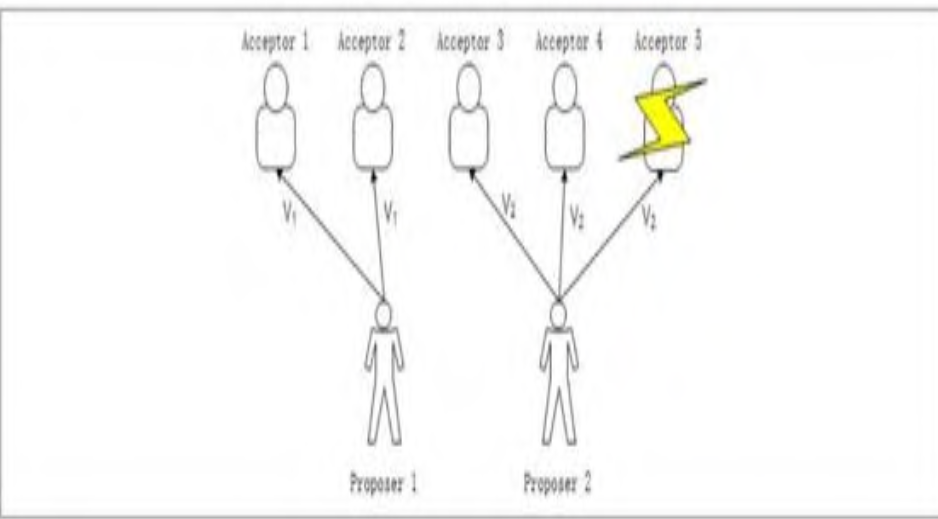
paxos定义了三个角色，分别为 proposer（提议者），acceptor（接受者），learner（学习者）。

## 提案的选定

* 考虑一个acceptor的情况，由于只有一个，Proposer只能发送提案给这个Accpetor，Acceptor会选择它接收到的第一个提案作为被选定的提案。虽然实现很简单，但是一旦acceptor出现故障整个系统就无法工作了。
* 所以需要考虑多个acceptor，但是如图，下面的提案都无法被选定，因为提案没有被半数以上acceptor批准，因此可以使用一个全局的编号，以“[编号，Value]”来表示一个提案。



图一



图二

根据编号我们引入P2：如果编号为 M0、Value值为 V0的提案（即[M0，V0]）被选定了，那么所有比编号M0更高的，且被选定的提案，其Value值必须也是V0。

P2a： [M0，V0]被选定了，那么所有比编号M0更高的，且被Acceptor批准的提案，其Value值必须也是V0。[[1]](#footnote-1)

P2b：[M0，V0]被选定后，那么之后任何Proposer产生的编号更高的提案，其Value值都为V0。

P2c：如果提案[Mn，Vn]被提出，那么肯定存在一个由半数以上的Acceptor组成的集合S，满足以下两个条件中的任意一个。

* S中不存在任何批准过编号小于Mn的提案的Acceptor。
* 选取S中所有Acceptor批准的编号小于Mn的提案，其中编号最大的那个提案其Value值是Vn。

P2c=＞P2b=＞P2a=＞P

## Proposer生成提案

Proposer 选择一个新的提案编号 Mn，然后向某个 Acceptor 集合的成员发送请求，要求该集合中的Acceptor做出如下回应。

* 向Proposer承诺，保证不再批准任何编号小于Mn的提案.
* 如果 Acceptor 已经批准过任何提案，那么其就向 Proposer反馈当前该Acceptor已经批准的编号小于Mn但为最大编号的那个提案的值。

我们将该请求称为编号为Mn的提案的Prepare请求。

在确定提案之后，Proposer就会将该提案再次发送给某个Acceptor集合，并期望获得它们的批准，我们称此请求为Accept请求。

1. 文本

   中度可信度描述已自动生成 [↑](#footnote-ref-1)