Paxos算法

1990年的论文中，Lamport描述了一个名为Paxos的希腊城邦，这个城邦是按照民主的议会制度来进行选举的，所有的居民进行提议和投票来选出决议。但是居民们不想花时间一直在选举上，大家都不定时地来提议、了解提议、投票、看进展等等，而Paxos算法的目标就是通过少数服从多数的方式来达成最终的一致意见，解决多个节点冲突，产生优先次序，达成这一意见有两个阶段，如下。

**一、Prepare准备阶段**：在该阶段，Proposer会尝试告诉所有的其他机器，我现在有一个提案（操作），请告诉我你们是否支持（是否能接受）。其他机器会看看自己是否已经支持其他提案了（是否接受过其他操作请求），并回复给Proposer（如果曾经接受过其他值，就告诉Proposer接受过什么值/操作）。

* Acceptor如果已经支持了编号N的提案，那么不会再支持编号小于N的提案，但可以支持编号更大的提案；
* Acceptor如果生效了编号为N的提案，那么不会再接受编号小于N的提案，且会在回复时告知当前已生效的提案编号与内容。

**二、Accept提交阶段**：在该阶段，Proposer根据上一阶段接收到的回复，来决定行为：

如果上一阶段超过半数的机器回复说接受提案，那么Proposer就正式通知所有机器去生效这个操作；

* 如果上一阶段超过半数的机器回复说他们已经先接受了其他编号更大的提案，那么Proposer会更新一个更大的编号去重试（随机延时）；
* 如果上一阶段的机器回复说他们已经生效了其他编号的提案，那么Proposer就也只能接受这个其他人的提案，并告知所有机器直接接受这个新的提案；
* 如果上一阶段都没收到半数的机器回复，那么提案取消。

PS：接受其他提案，以及提案取消的情况下，Proposer就要直接告诉客户端该次请求失败了，等待客户端重试即可。

这里可以看到，超过半数以上的机器是个很重要的结果它是走向结果的条件。

至此，已经描述完了针对一次达成共识的过程，这就是paxos算法的两个阶段，最终达成一致意见。