谈一谈对paxos 算法的理解与认识

通过这学期接触到hadoop这门课程，我了解到了很多不同的算法。其中，我对paxos算法印象深刻。首先，我先来简单介绍一下paxos 算法是什么。Paxos 算法是基于消息传递且具有高度容错特性的一致性算法，是目前公认的解决分布式一致性问题最有效的算法之一。

令我最为敬佩的是这种基于消息传递的分布式一致性算法获得了2013年的图灵奖，并且这一算法自从问世以来持续垄断了分布式一致性算法，同时，Google的很多大型分布式系统都采用了Paxos算法来解决分布式一致性问题，比如chubby, megastore 以及spanner等。开源的zookeeper, 以及mysql 5.7推出的用来取代传统的主从复制的Mysql Group Replication 等纷纷采用paxos 算法解决分布式一致性问题。

由此可见，paxos 算法的重要性，先进性与实用性。更加表明了此算法对我们大数据专业的学生学习专业知识的重要性。但是经过一段时间的简单了解之后，我发现这个算法存在两个明显的缺点，第一点就难以理解，难以操作。第二点是它在工程上的实现较为复杂。因为在生活中常见的分布式系统中，总会发生诸如机器待机或者网络异常等问题（消息的延迟，信息的丢失，重复，乱序，网络分区）。Paxos 算法需要解决的问题就是如何在一个可能发生上述此类的系统中，在集群内部对某个数据的值达成一致。最重要的是此算法必须保证不论发生以上任何异常，都不会破化整个系统的一致性。

在paxos 算法中，存在四个角色proposer, acceptor, learner, client 。其中proposer角色可以由多个，并在算法开始时提出提案，acceptor角色在集群中有n个，彼此完全对等且独立。Learner 角色不参与选举过程，而是学习被批准的value 。client 角色作为议题的产生者，并不直接参与选举过程。

通过一系列的推导，我们总结了paxos算法的流程。首先proposer选择一个提案编号N,然后向半数以上的acceptor发送编号为n 的prepare 请求。如果一个acceptor收到一个编号为n的prepare 请求，且N大于该acceptor已经响应过的所有Prepare 请求的编号，那么它就会将它已经接受过的编号最大的提案作为响应反馈给proposer, 同时该accepter 承诺不再接受任何编号小于n 的提案。

综上所述，是我对paxos 算法的理解与认识。