# Paxos

1. 简介：paxos算法是为了实现容错分布式系统而提出的，它的核心是为了实现一致性
2. 问题

假设一些进程都可以扮演提议者（proposer）、接收者（acceptor）、学习者（learner）其中的一个或多个，提议者可以提出value，接收者可以选择value，学习者可以学习value，一致性算法就是为了保证在所有的values中只有唯一的value会被选择，并且保证只有被提出的value 才可以被选择，只有被选择的value才能被学习，即以下三点安全性要求（safety requirements）:

1. 只有被提出(proposal)的value 才可以被选择(chosen)
2. 在所有的values中只有唯一的value会被选择(chosen)
3. 只有被选择(chosen)的value才能被学习(learn)

同时，该算法还要实现以下活性要求（liveness requirements）:

1. 最终会有value被选择
2. 选择的value最终会被学习
3. 怎么选择一个value

先假设一个最简单的模型，只有一个接收者和一个提议者，则提议者提出value，接收者选择最先接收的value。但是当这个接收者失败的时候，就不能保证程序有进展。

则将一个接收者修改为多个接收者，proposer向多个接收者发送提出的value,接收者可以选择接收某个value，当一个value被大部分的acceptor（majority）接收的时候，则这个value被选择。（majority是为了保证选择的value必须唯一，因为每个acceptor都只能接收一个value，而每两个majority都肯定存在交集，所以不可能有一个以上的m ajority选择不同的value）

因为可能存在消息丢失和进程失败的情况，即使只有一个提议者提出了一个proposal也要能够被接收者接受，则需要保证以下要求：

[1]、一个接收者必须接收它收到的第一个proposal

但在实际场景中，经常会有多个提议者同时提出不同的value，则很有可能会出现一种情况，即所有的接收者都接收到了value，但是却没有一个majority接收到了相同的value，此时就需要提出新的要求：

接收者可以接受多余一个value，为此我们设置一个数值（number）来记录接收者接收提议的时间顺序，提议者提出的内容也从单一的value变成包含数值（number）和value的提议（proposal），并且不同的proposal的number值必须不同，随着被提出时间的后移，number值越来来越大。则我们说一个value被选择就是包含这个value的proposal被选择。但是因为每个接收者都可以接受不止一个proposal，则可能会有多个proposal被majority接收，即被选择。

为了保证安全性要求中的第二点（在所有的values中只有唯一的value会被选择(chosen)），所以我们需要保证所有被选择的proposal都包含相同的value，即：

[2]、只要一个proposal假设为（n1,v1）被选择，其他任何被选择的number大于n1的value都必须与已选中的v1相同。

[3]、为了保证以上的要求，我们只需要保证当value被选择之后，所有大于n1的被接收的value 都与v1相同。

但可能会存在一种情况，即新出现了一个提议者，它向一个还没有接收任何proposal的接收者发送了一个proposal,而该proposal的value并不与其他接收者接收的proposal相同，这样就违反了第[3]条要求，因此就需要保证新的要求：

[4]、当value被选择之后，所有大于n1的被提出的proposal中的value 都与v1相同。

但是怎么去保证之后所有被提出的proposal中的value都与v1相同呢，我们只需要保证以下的要求即可：

[5]、如果一个proposal（n, v）被提出，那么在一个majority中的所有的acceptor中：a、没有任何一个acceptor接收过任何proposal，则v可以是任意值。b、已经有acceptor接收了proposal，则v必须是所有的proposal中number最大的那个proposal对应的value。

所以，为了满足以上的要求，当一个提议者想要提出proposal时，他就需要知道接收者是否已经接收了其他的proposal，如果已经接收了，则其中最大的number和对应的value分别是什么，这样他才能决定自己的proposal的内容。因此可以模拟以下过程：

1. 当提议者想要提出proposal时，它先选好该proposal的number值，设为n，并向一个majority的acceptor发送申请，请求这些acceptor向他返回以下内容：
2. 一个promise保证再也不会接收number值小于n2的proposal
3. 该acceptor当前已经接收的最大的number值的proposal（如果还没有接收到proposal，则不包含b内容）
4. 如果这个提议者收到来自一个majority的acceptor的返回信息promise，则它可以提出一个proposal（n, v）其中n为之前选好的number值，v则是所有promise中返回的number最大的value值，但如果所有的promise中都不包含value值的话，则v的值可以由该提议者自己决定

则提议者通过以上的方式确定好要提出的proposal之后，再向majority的acceptor发送这个proposal，并请求被接收。可以看出，提议者需要向acceptor发送两个申请才可能使proposal被接收，我们把这两个申请分别称之为预备申请（prepare request）和接收申请（accept request）。而当一个acceptor收到accept request之后，根据之前promise中包含的第一条信息，我们可以得出：

[6]、当一个acceptor收到一个accept request 的proposal [n, v]时，如果它没有回应过任何number值比n大的prepare request，则这个acceptor可以接收这个proposal。

于是我们可以整个算法的流程：

1、（a）一个提议者选择一个number：n,并向majority的acceptor发送prepare request

（b）一个acceptor收到一个 prepare request，如果这个prepare request中的n大于所有的该acceptor已经接收到的proposal的number的值，则acceptor向提议者返回一个promise，其中包括

* + 1. 保证不在接收任何小于n的prepare request
    2. 该acceptor当前已经接收的最大的number值的proposal（如果还没有接收到proposal，则不包含此内容）

2、（a）一个提议者如果收到了来自一个majority返回的promise，则该提议者产生一个proposal（n, v）并向majority的acceptor发送accept request，n为prepare阶段产生的n值，其中v的取值有两种可能：

1. 如果所有的promise中都不包含value值的话，则v的值可以由该提议者自己决定
2. v则是所有promise中返回的number最大的value值

（b）如果一个acceptor收到了一个accept request，设为proposal（n, v）那么如果这个acceptor还没有收到number值大于n的prepare request，那么它就接收这个proposal。

1. 学习一个已选择的value

经过以上过程，已经可以在保证安全性的前提下实现提议者提出proposal并且被接收者接收了。当有一个majority的acceptor都接收了相同value的proposal之后，我们就说这个value的proposal被选择（chosen）了。那么选择的value最终怎么被学习到就需要学习者发挥作用。

每当一个acceptor接收了一个proposal的时候，它就会向所有的learner发送包含这个proposal的通知，learner就会将这条消息记录下来，当learner收到的来自不同的acceptor的相同value的通知超过majority的时候，learner就学习到这个value被chosen。

但是因为learner数可能很多，每个acceptor都向每个learner发送通知可能会很费时费力，因此我们可以选出一个distinguished learner ，所有的acceptor都只想这个learner发送通知，当一个value被chosen之后，再由这个learner向其他的所有的learner发送通知。

但是这个distinguished learner可能会失败，所以为了保证程序进行我们可以选出一个多个distinguished learner集，每个acceptor都只需向这个集合中的learner发送通知，当一个value被chosen之后，再由这些learner通知其他的所有learner，由于这个集合中learner的数量还是远远小于所有的learner数，所有可以再保证程序安全性进行的前提下大大提升了效率。

1. 保证progress

以上程序虽然可以保证安全性要求，却不可以完全保证活性要求，既不能保证程序的进展，可能会产生活锁。

比如：当number为n1的proposal收到足够的promise后，向acceptor发送accept request时，该acceptor又收到number为n2的proposal的prepare request，而n2>n1,则acceptor不再接收n1的proposal;而当n2收集到足够的promise,向acceptor发送accept request时，又出现更大的number n3 的proposal的prepare request；以此往复，则一直不会有proposal被选中，但程序也不会block.）

则需要通过一个leader election算法来选举一个leader proposer，使得只有他能够发送proposal,则可以避免在value chosen 过程中活锁的产生，保证程序的进展。