Paxos

一、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

二、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

三、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 majority 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：
   a、 一个 promise 保证再也不会接收 number 值小于 n2 的 proposal
   b、 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）
2、 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。

于是我们可以得出整个算法的流程：

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

i. 保证不在接收任何小于 n 的 prepare request
ii. 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：

i. 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
ii. 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。

四、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。

五、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。