# 2P/3P

为了保证事务的ACID（原子性、一致性、隔离性、持久性）

2P= Two Phase commit 二段提交（RDBMS经常就这种机制，保证强一致性）

3P= Three Phase commit 三段提交

2P- 阶段1：提交事务请求（投票阶段）

事

务

管

理

器

资源1

资源2

prepare

ok

prepare

ok

写入undo和redo事务日志

写入undo和redo事务日志

2P- 阶段2：执行事务提交（commit、rollback）

事

务

管

理

器

资源1

资源2

rollback

ok

rollback

failed

事

务

管

理

器

资源1

资源2

commit

ok

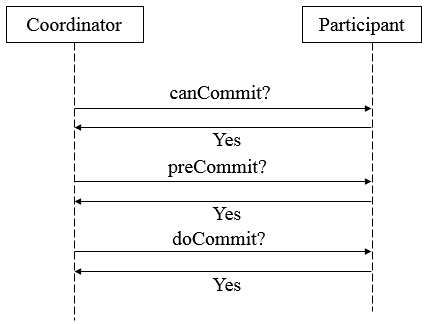
commit

ok

3P- 阶段1：是否提交

3P- 阶段2：预先提交

3P- 阶段3：提交（commit、rollback）



# CAP理论（分布式系统遵循的理论）

一致性（Consistency）、可用性（Availability）、分区容错性（Partition tolerance）



CA(放弃P)：讲所有的数据放在一个节点。满足一致性、可用性。

AP(放弃C)：放弃强一致性，用最终一致性来保证。

CP(放弃A)：一旦系统遇见故障，受到影响的服务器需要等待一段时间，在恢复期间无法对外提供服务。

# BASE理论

基本可用（bascially available）、软状态（soft state）、最终一致性（Eventually consistent）

基本可用：在分布式系统出现，允许损失部分可用性（服务降级、页面降级）

软状态：允许分布式系统出现中间状态。而且中间状态不影响系统的可用性。

最终一致性：data replications经过一段时间达到一致性。

# Paxos算法

**拜占庭将军问题**

拜占庭帝国就是5~15世纪的东罗马帝国，拜占庭即现在土耳其的伊斯坦布尔。我们可以想象，拜占庭军队有许多分支，驻扎在敌人城外，每一分支由各自的将军指挥。将军们智能靠通讯员进行通讯。在观察敌人以后，忠诚的将军们必须制订一个统一的行动计划——进攻或者撤退。然而，这些将军冢有叛徒，他们不希望忠诚的将军们能达成一致，因而影响统一行动计划的制订与传播。

问题是：将军们必须有一个协议，使所有忠诚的将军们能够达成一致，而且少数几个叛徒不能使忠诚的将军们作出错误的计划——使有些将军进攻而另一些将军撤退。

如果这11位将军中有间谍呢? 假设有9位忠诚的将军，5位判断进攻，4位判断撤退，还有2个间谍恶意判断撤退，虽然结果是错误的撤退，但这种情况完全是允许的。因为这11位将军依然保持着状态一致性。

1.11位将军进攻城池

2.同时进攻（议案、决议）、同时撤退（议案、决议）

3.不管撤退还是进攻，必须半数的将军统一意见才可以执行

4.将军里面有叛徒，会干扰决议生成

在Paxos算法面前，其他（包括2P、3P）都是渣渣，都是残次品。Google Chubby的作者Mike Burrows说过这个世界上只有一种一致性算法，那就是Paxos，其它的算法都是残次品。

Paxos：多数派决议（最终解决一致性问题）

三种角色：

Proposer：提交者（议案提交者）

提交议案(判断是否过半)，提交批准议案(判断是否过半)

Acceptor：接收者（议案接收者）

接受议案或者驳回议案，给proposer回应(promise)

Learner：学习者（打酱油的）

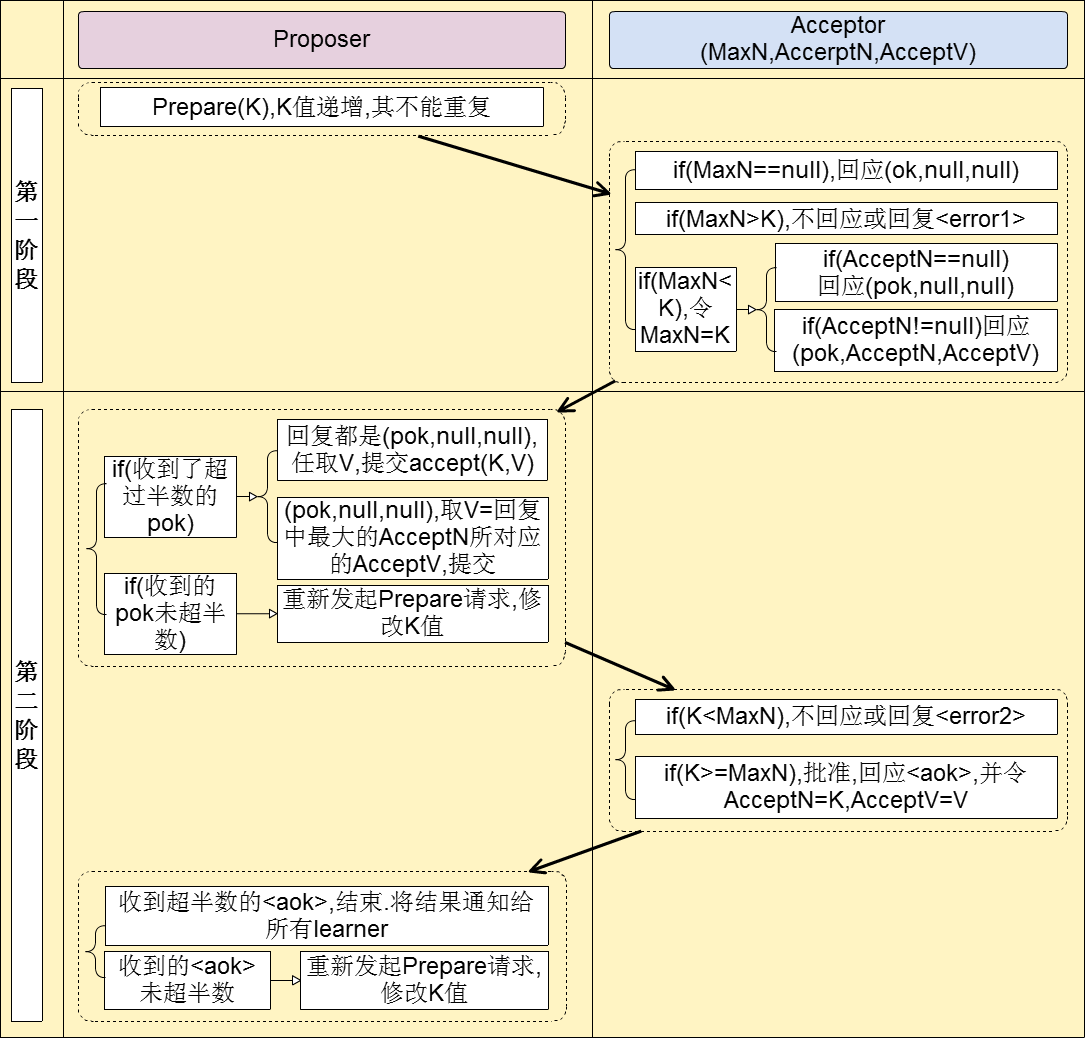
如果议案产生，学习议案。

**设定1：**如果Acceptor没有接受议案，那么他必须接受第一个议案

**设定2：**每个议案比必须有一个编号，并且编号只能增长，不能重复。越往后越大。

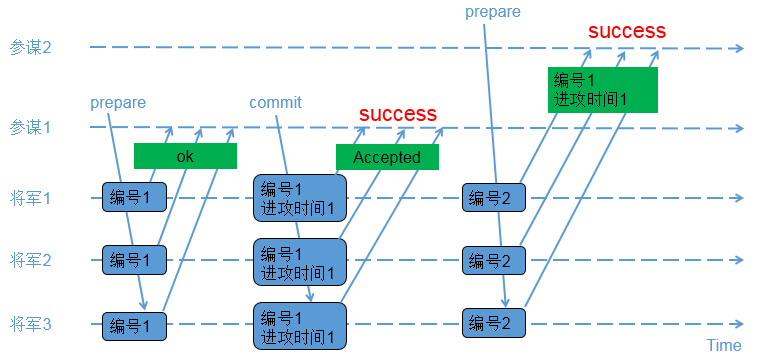
**设定3：**接受编号大的议案，如果小于之前接受议案编号，那么不接受

**设定4：**议案有2种(提交的议案，批准的议案)



1. Prepare阶段（决议提交）
2. Proposer希望议案V。首先发出Prepare请求至大多数Acceptor。Prepare请求内容为序列号K
3. Acceptor收到Prepare请求为编号K后，检查自己手里是否有处理过Prepare请求。
4. 如果Acceptor没有接受过任何Prepare请求，那么用OK来回复Proposer，代表Acceptor必须接受收到的第一个议案（设定1）
5. 否则，如果Acceptor之前接受过任何Prepare请求（如：MaxN），那么比较议案编号，如果K<MaxN，则用reject或者error回复Proposer
6. 如果K>=MaxN，那么检查之前是否有批准的议案，如果没有则用OK来回复Proposer，并记录K
7. 如果K>=MaxN，那么检查之前是否有批准的议案，如果有则回复批准的议案编号和议案内容（如：<AcceptN, AcceptV>， AcceptN为批准的议案编号，AcceptV为批准的议案内容）
8. Accept阶段（批准阶段）
9. Proposer收到过半Acceptor发来的回复，回复都是OK，且没有附带任何批准过的议案编号和议案内容。那么Proposer继续提交批准请求，不过此时会连议案编号K和议案内容V一起提交（<K, V>这种数据形式）
10. Proposer收到过半Acceptor发来的回复，回复都是OK，且附带批准过的议案编号和议案内容（<pok，议案编号，议案内容>）。那么Proposer找到所有回复中超过半数的那个（假设为<pok，AcceptNx，AcceptVx>）作为提交批准请求（请求为<K，AcceptVx>）发送给Acceptor。
11. Proposer没有收到过半Acceptor发来的回复，则修改议案编号K为Kx，并将编号重新发送给Acceptors（重复Prepare阶段的过程）
12. Acceptor收到Proposer发来的Accept请求，如果编号K<MaxN则不回应或者reject。
13. Acceptor收到Proposer发来的Accept请求，如果编号K>=MaxN则批准该议案，并设置手里批准的议案为<K，接受议案的编号，接受议案的内容>，回复Proposer。
14. 经过一段时间Proposer对比手里收到的Accept回复，如果超过半数，则结束流程（代表议案被批准），同时通知Leaner可以学习议案。
15. 经过一段时间Proposer对比手里收到的Accept回复，如果未超过半数，则修改议案编号重新进入Prepare阶段。

Paxos例子



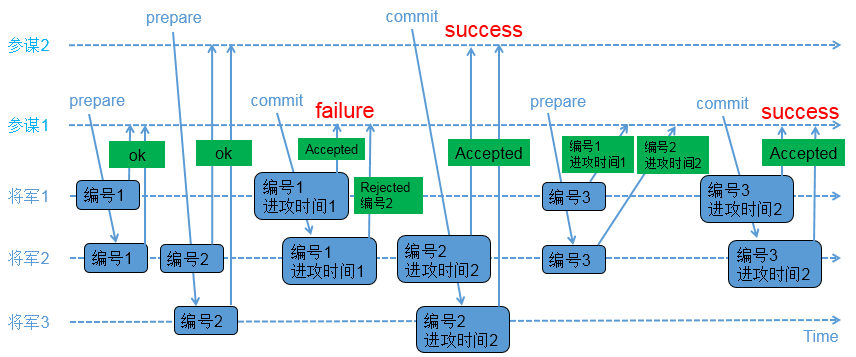
1. 先后提议的场景

角色：

proposer：参谋1，参谋2

acceptor：将军1，将军2，将军3（决策者）

1. 参谋1发起提议，派通信兵带信给3个将军，内容为（编号1）；
2. 3个将军收到参谋1的提议，由于之前还没有保存任何编号，因此把（编号1）保存下来，避免遗忘；同时让通信兵带信回去，内容为（ok）；
3. 参谋1收到至少2个将军的回复，再次派通信兵带信给3个将军，内容为（编号1，进攻时间1）；
4. 3个将军收到参谋1的时间，把（编号1，进攻时间1）保存下来，避免遗忘；同时让通信兵带信回去，内容为（Accepted）；
5. 参谋1收到至少2个将军的（Accepted）内容，确认进攻时间已经被大家接收；
6. 参谋2发起提议，派通信兵带信给3个将军，内容为（编号2）；
7. 3个将军收到参谋2的提议，由于（编号2）比（编号1）大，因此把（编号2）保存下来，避免遗忘；又由于之前已经接受参谋1的提议，因此让通信兵带信回去，内容为（编号1，进攻时间1）；
8. 参谋2收到至少2个将军的回复，由于回复中带来了已接受的参谋1的提议内容，参谋2因此不再提出新的进攻时间，接受参谋1提出的时间；



1. 交叉场景

proposer：参谋1，参谋2

acceptor：将军1，将军2，将军3（决策者）

1. 参谋1发起提议，派通信兵带信给3个将军，内容为（编号1）；
2. 3个将军的情况如下
3. 将军1和将军2收到参谋1的提议，将军1和将军2把（编号1）记录下来，如果有其他参谋提出更小的编号，将被拒绝；同时让通信兵带信回去，内容为（ok）；
4. 负责通知将军3的通信兵被抓，因此将军3没收到参谋1的提议；
5. 参谋2在同一时间也发起了提议，派通信兵带信给3个将军，内容为（编号2）；
6. 3个将军的情况如下
7. 将军2和将军3收到参谋2的提议，将军2和将军3把（编号2）记录下来，如果有其他参谋提出更小的编号，将被拒绝；同时让通信兵带信回去，内容为（ok）；
8. 负责通知将军1的通信兵被抓，因此将军1没收到参谋2的提议；

1. 参谋1收到至少2个将军的回复，再次派通信兵带信给有答复的2个将军，内容为（编号1，进攻时间1）；
2. 2个将军的情况如下
3. 将军1收到了（编号1，进攻时间1），和自己保存的编号相同，因此把（编号1，进攻时间1）保存下来；同时让通信兵带信回去，内容为（Accepted）；
4. 将军2收到了（编号1，进攻时间1），由于（编号1）小于已经保存的（编号2），因此让通信兵带信回去，内容为（Rejected，编号2）；

1. 参谋2收到至少2个将军的回复，再次派通信兵带信给有答复的2个将军，内容为（编号2，进攻时间2）；
2. 将军2和将军3收到了（编号2，进攻时间2），和自己保存的编号相同，因此把（编号2，进攻时间2）保存下来，同时让通信兵带信回去，内容为（Accepted）；
3. 参谋2收到至少2个将军的（Accepted）内容，确认进攻时间已经被多数派接受；
4. 参谋1只收到了1个将军的（Accepted）内容，同时收到一个（Rejected，编号2）；参谋1重新发起提议，派通信兵带信给3个将军，内容为（编号3）；
5. 3个将军的情况如下
6. 将军1收到参谋1的提议，由于（编号3）大于之前保存的（编号1），因此把（编号3）保存下来；由于将军1已经接受参谋1前一次的提议，因此让通信兵带信回去，内容为（编号1，进攻时间1）；
7. 将军2收到参谋1的提议，由于（编号3）大于之前保存的（编号2），因此把（编号3）保存下来；由于将军2已经接受参谋2的提议，因此让通信兵带信回去，内容为（编号2，进攻时间2）；
8. 负责通知将军3的通信兵被抓，因此将军3没收到参谋1的提议；
9. 参谋1收到了至少2个将军的回复，比较两个回复的编号大小，选择大编号对应的进攻时间作为最新的提议；参谋1再次派通信兵带信给有答复的2个将军，内容为（编号3，进攻时间2）；
10. 将军1和将军2收到了（编号3，进攻时间2），和自己保存的编号相同，因此保存（编号3，进攻时间2），同时让通信兵带信回去，内容为（Accepted）；
11. 参谋1收到了至少2个将军的（accepted）内容，确认进攻时间已经被多数派接受。

# ZAB协议