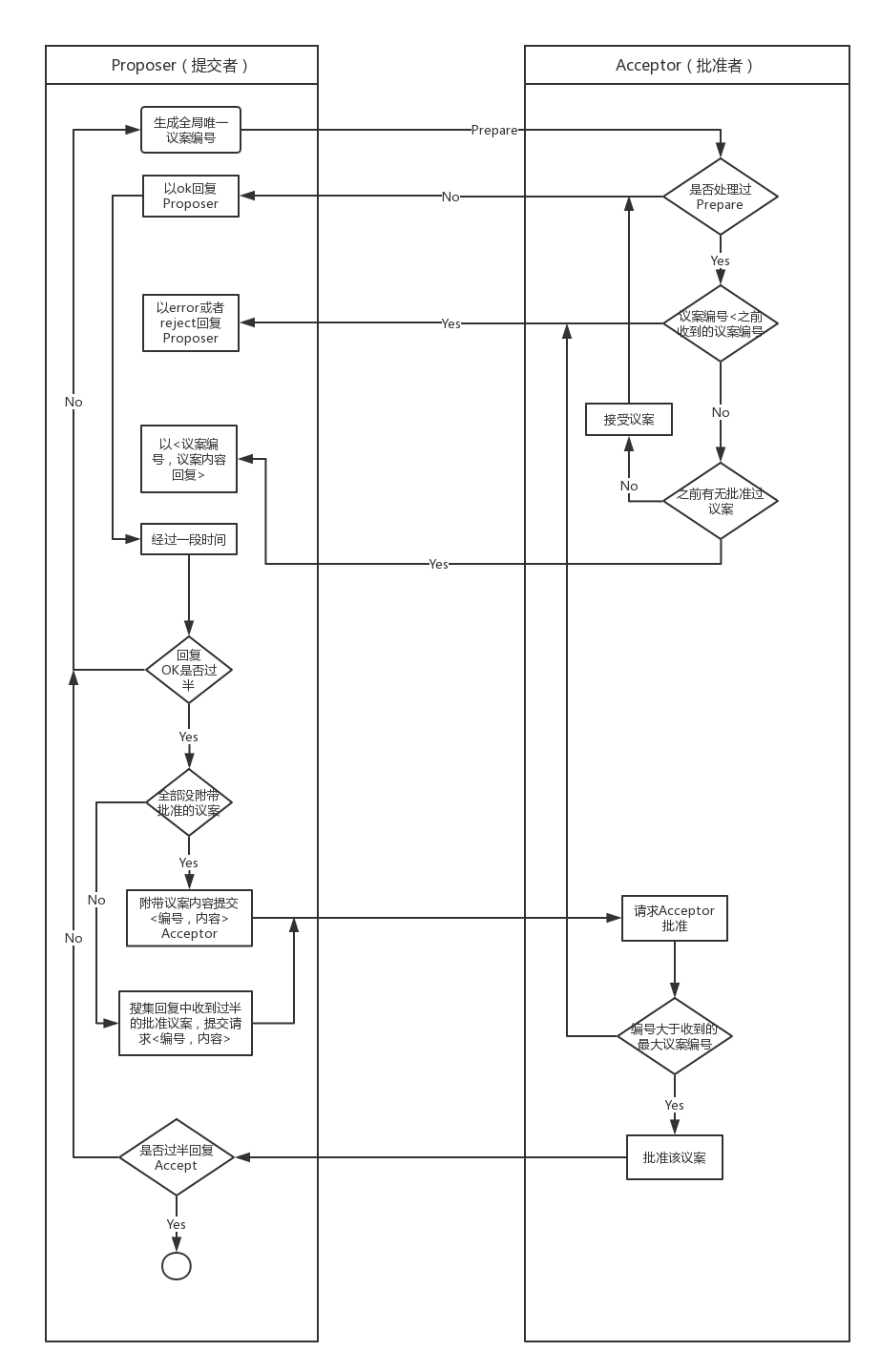
# 温习Paxos协议



# Zookeeper ZAB协议

Zookeeper Automic Broadcast(ZAB)，是paxos经典实现。

术语：

quorum：集群过半数的集合

ZAB(zookeeper)中节点分三种状态：

looking：选举Leader的状态（崩溃恢复下）

following：跟随者（follower）的状态，服从Leader命令

leading：当前节点是Leader，负责协调工作。

observing：observer(观察者)，不参与选举，只读节点。

ZAB中的两个模式：

崩溃恢复、消息广播

启动

正常运行

崩溃恢复

数据同步

**崩溃恢复**

**崩溃恢复**

**消息广播**

启动

正常运行

崩溃恢复

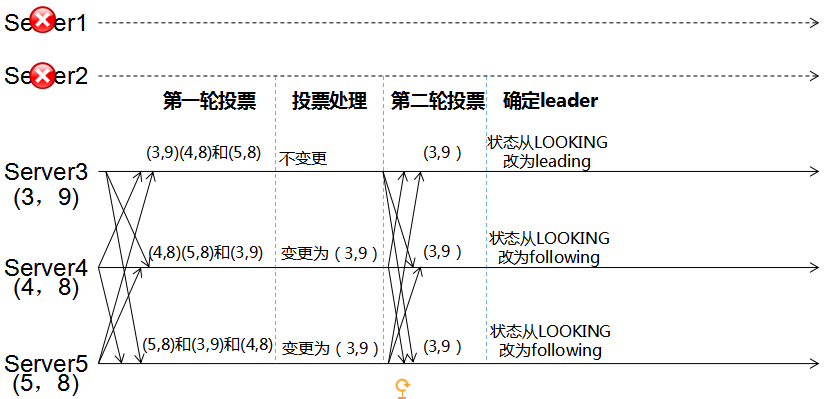
数据同步

**崩溃恢复**

**崩溃恢复**

**消息广播**

崩溃恢复：



1. 每个server都有一张选票<myid，zxid>，选票投自己。
2. 搜集各个服务器的投票。
3. 比较投票，比较逻辑：优先比较zxid，然后才比较myid。
4. 改变服务器状态（崩溃恢复=》数据同步，或者崩溃恢复=》消息广播）

epoch周期值

acceptedEpoch（比喻：年号）：follower已经接受leader更改年好的（newepoch）提议。

currentEpoch（比喻：当前的年号）：当前的年号

lastZxid：history中最近接收到的提议zxid(最大的值)

history：当前节点接受到事务提议的log

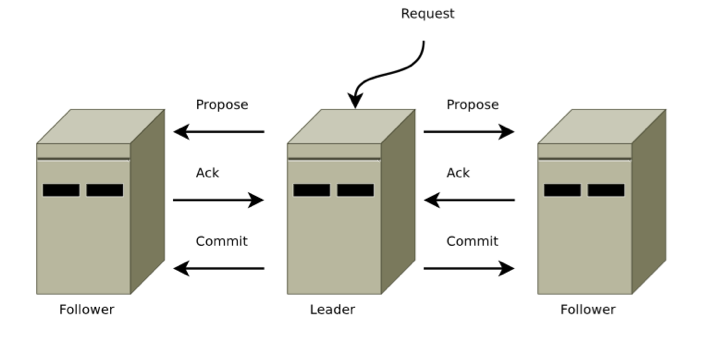
Zxid(64位的数据结构)

前32位：39000 Leader 周期编号 myid

低32位：0000f 事务的自增序列（单调递增的序列）只要客户端有请求，就+1

当产生新Leader的时候，就从这个Leader服务器上取出本地log中最大事务zxid，从里面读出epoch+1，作为一个新epoch，并将低32位置0（保证id绝对自增）。

消息广播(类似2P提交)：



1. Leader接受请求后，讲这个请求赋予全局的唯一64位自增Id（zxid）。
2. 将zxid作为议案发给所有follower。
3. 所有的follower接受到议案后，想将议案写入硬盘后，马上回复Leader一个ACK（OK）。
4. 当Leader接受到合法数量Acks，Leader给所有follower发送commit命令。
5. follower执行commit命令。

ps:到了这个阶段，ZK集群才正式对外提供服务，并且Leader可以进行消息广播，如果有新节点加入，还需要进行同步。

数据同步：

1. 取出Leader最大lastZxid（从本地log日志来）
2. 找到对应zxid的数据，进行同步（数据同步过程保证所有follower一致）。
3. 只有满足quorum同步完成，准Leader才能成为真正的Leader。

# ZkClient

优点1：递归创建

优点2：递归删除

优点3：避免不存在异常

ZkClient注册事件：

subscribeChildChanges/unsubscribeChildChanges(节点变化)

subscribeDataChanges/unsubscribeDataChanges（数据变化）

# Curator

curator连接ZK应用最广泛的工具。

1. zk应用场景（分布式锁，Master选举等等），curator包含了这些场景。
2. 应用场景出现极端的情况下，curator考虑。

Backoff退避算法：

1. 每1分钟重试一次，如果网络出现阻塞。

22：25 request1（block）

22：26 request2（毫无意义）

22：27 request3（毫无意义）

22：28 request4（通顺）request2、3、4

1. 按照指数间隔重试，比如第一次1分钟，第二次2分钟......随着时间的推移，重试间隔越长。

Curator时间监听：

NodeCache：节点处理监听（会使用缓存）。回调接口NodeCacheListener

PathChildrenCache：子节点缓存，处理子节点变化。回调接口PathChildrenCacheListener

TreeCache：NodeCache和PathChildrenCache的结合体。回调接口TreeCacheCacheListener

# Zookeeper在大型分布式系统中的应用

Hadoop

Hbase

Kafka