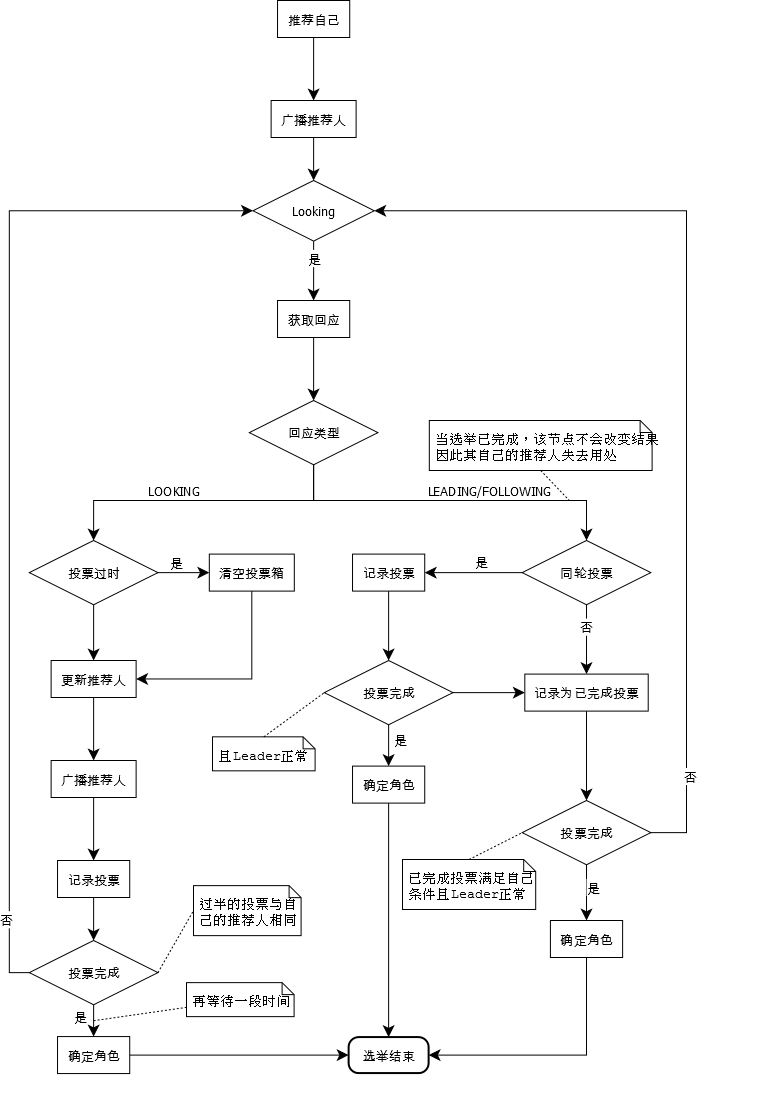
[FastLeader选主算法](http://blog.csdn.net/xhh198781/article/details/6619203)：

看网上关于 zookeeper选主节点fast算法的描述，虽然有几篇写的非常不错，但是总感觉描述的差一些，因此打算写一个我认为的较为详细的版本让大家提点意见。当然如果有Paxos基础那自然看起来非常很简单。

具体的FAST算法流程如下所示:



下面详细解释一下这个流程：

首先给出几个名词定义：

（1）Serverid：在配置server时，给定的服务器的标示id。

（2）Zxid:服务器在运行时产生的数据id，zxid越大，表示数据越新。

（3）Epoch：选举的轮数，即逻辑时钟。随着选举的轮数++

（4）Server状态：LOOKING,FOLLOWING,OBSERVING,LEADING

步骤：

1. Server刚启动（宕机恢复或者刚启动）准备加入集群，此时读取自身的zxid等信息。
2. 所有Server加入集群时都会推荐自己为leader，然后将（leader id 、 zixd 、 epoch）作为广播信息，广播到集群中所有的服务器(Server)。然后等待集群中的服务器返回信息。
3. 收到集群中其他服务器返回的信息，此时要分为两类：该服务器处于looking状态，或者其他状态。
4. 服务器处于looking状态

首先判断逻辑时钟 Epoch:

* 1. 如果接收到Epoch大于自己目前的逻辑时钟（说明自己所保存的逻辑时钟落伍了）。更新本机逻辑时钟Epoch，同时 Clear其他服务发送来的选举数据（这些数据已经OUT了）。然后判断是否需要更新当前自己的选举情况（一开始选择的leader id 是自己）

判断规则rules judging：保存的zxid最大值和leader Serverid来进行判断的。先看数据zxid,数据zxid大者胜出;其次再判断leader Serverid, leader Serverid大者胜出；然后再将自身最新的选举结果(也就是上面提到的三种数据（leader Serverid，Zxid，Epoch）广播给其他server)

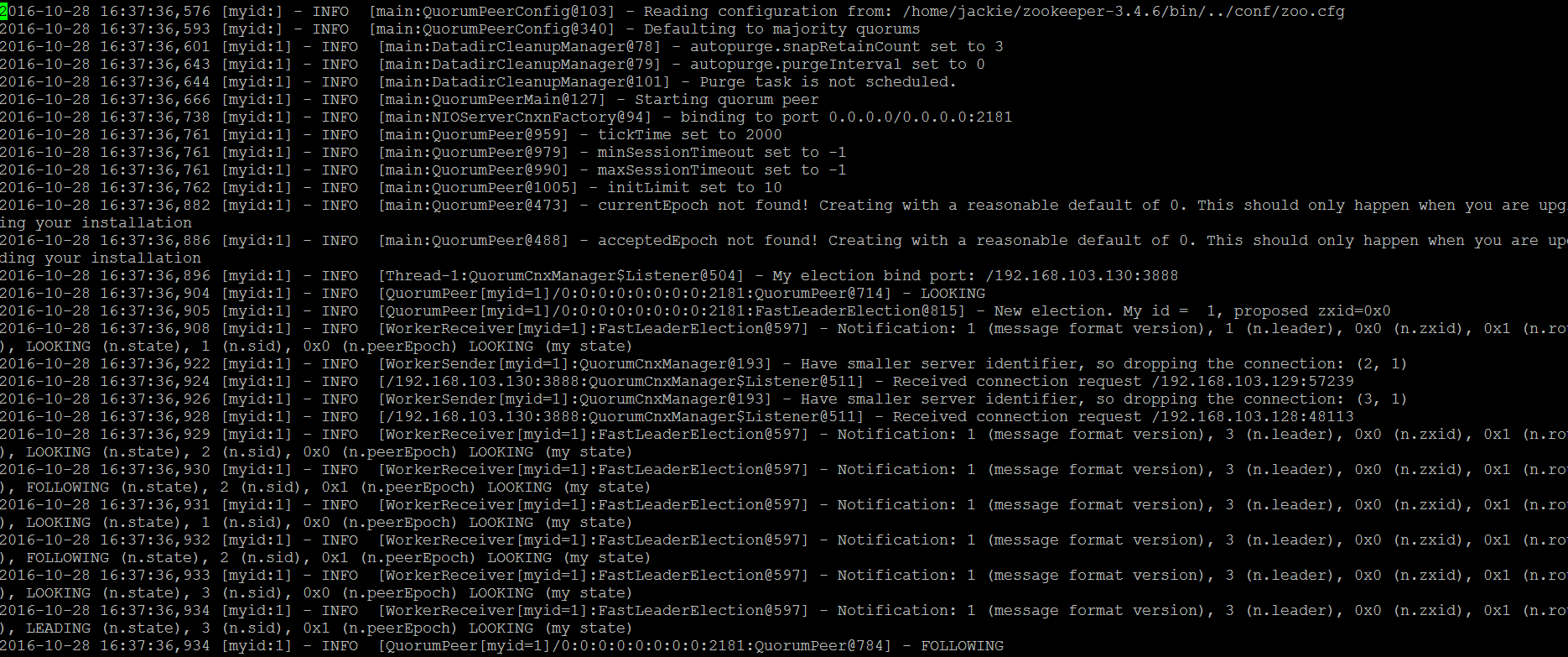
* 1. 如果接收到的Epoch小于目前的逻辑时钟。说明对方处于一个比较OUT的选举轮数，这时只需要将自己的 （leader Serverid，Zxid，Epoch）发送给他即可。
  2. 如果接收到的Epoch等于目前的逻辑时钟。再根据a)中的判断规则，将自身的最新选举结果广播给其他 server。

同时Server还要处理2种情况：

1. 如果Server接收到了其他所有服务器的选举信息，那么则根据这些选举信息确定自己的状态（Following,Leading），结束Looking，退出选举。
2. 即使没有收到所有服务器的选举信息，也可以判断一下根据以上过程之后最新的选举leader是不是得到了超过半数以上服务器的支持，如果是则尝试接受最新数据，倘若没有最新的数据到来，说明大家都已经默认了这个结果,同样也设置角色退出选举过程。
3. 服务器处于其他状态（Following, Leading）
   1. 如果逻辑时钟Epoch相同,将该数据保存到recvset,如果所接收服务器宣称自己是leader,那么将判断是不是有半数以上的服务器选举它,如果是则设置选举状态退出选举过程
   2. 否则这是一条与当前逻辑时钟不符合的消息，那么说明在另一个选举过程中已经有了选举结果，于是将该选举结果加入到outofelection集合中，再根据outofelection来判断是否可以结束选举,如果可以也是保存逻辑时钟，设置选举状态，退出选举过程。

以上就是FAST选举过程。

Zookeeper具体的启动日志如下图所示：



以上就是我自己配置的Zookeeper选主日志，从一开始LOOKING,然后new election, my id = 1, proposed zxid=0x0 也就是选自己为Leader,之后广播选举并重复之前Fast选主算法，最终确定Leader。

参考： <http://blog.csdn.net/xhh198781/article/details/6619203>

<http://blog.itpub.net/30109892/viewspace-2089610/>

http://www.open-open.com/lib/view/open1413796647528.html