哨兵

一、哨兵机制

在Redis主从集群中，哨兵机制是实现主从库自动切换的关键机制，它有效地解决了主从复制模式下故障转移的这三个问题。

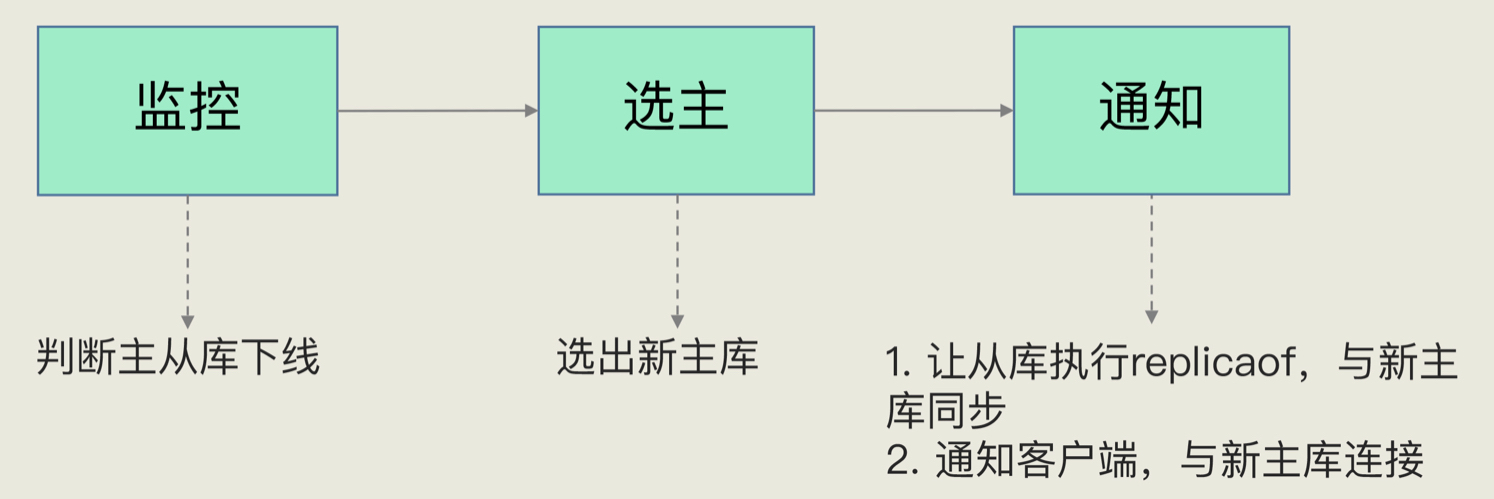
1.1 基本流程

哨兵其实就是一个运行在特殊模式下的Redis进程，主从库实例运行的同时，它也在运行。哨兵主要负责的就是三个任务：监控、选主（选择主库）和通知。

- [ ] 监控。哨兵进程在运行时，周期性地给所有的主从库发送PING命令，检测它们是否仍然在线运行。如果从库没有在规定时间内响应哨兵的PING命令，哨兵就会把它标记为“下线状态”；同样，如果主库也没有在规定时间内响应哨兵的PING命令，哨兵就会判定主库下线，然后开始自动切换主库的流程。

- [ ] 选主。主库挂了以后，哨兵就需要从很多个从库里，按照一定的规则选择一个从库实例，把它作为新的主库。这一步完成后，现在的集群里就有了新主库。

- [ ] 通知。在执行通知任务时，哨兵会把新主库的连接信息发给其他从库，让它们执行replicaof命令，和新主库建立连接，并进行数据复制。同时，哨兵会把新主库的连接信息通知给客户端，让它们把请求操作发到新主库上。

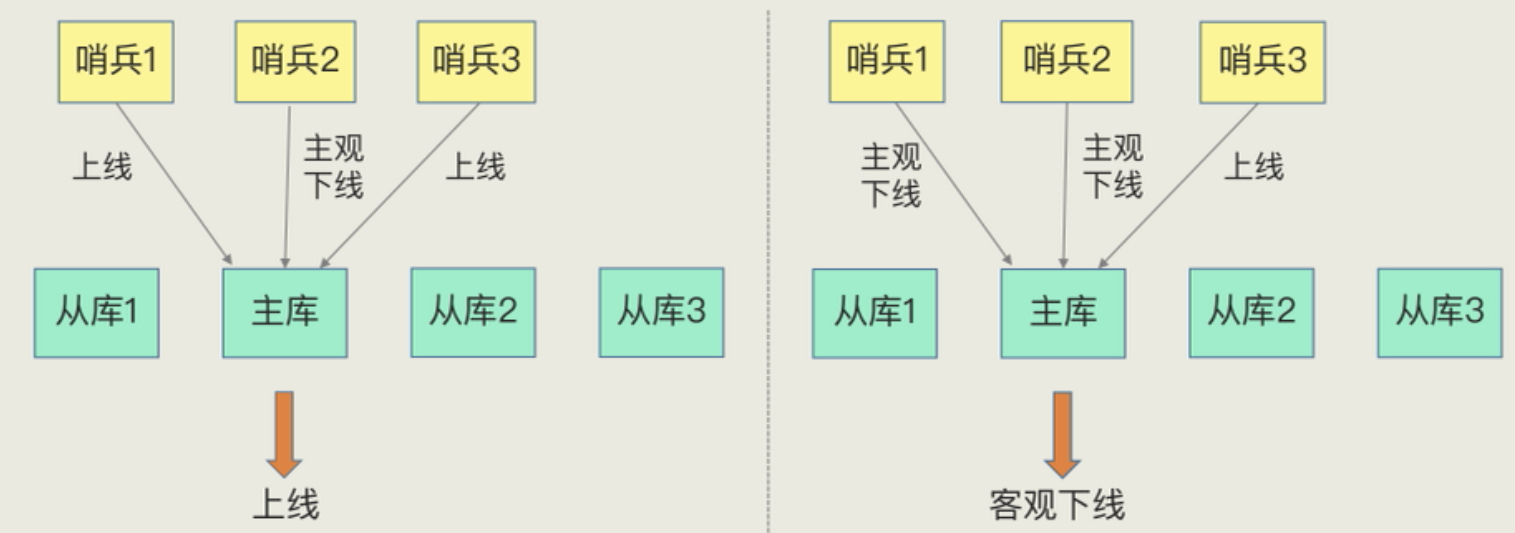


1.2 主观下线和客观下线

哨兵进程会使用PING命令检测它自己和主、从库的网络连接情况，用来判断实例的状态。如果哨兵发现主库或从库对PING命令的响应超时了，那么，哨兵就会先把它标记为“主观下线”。

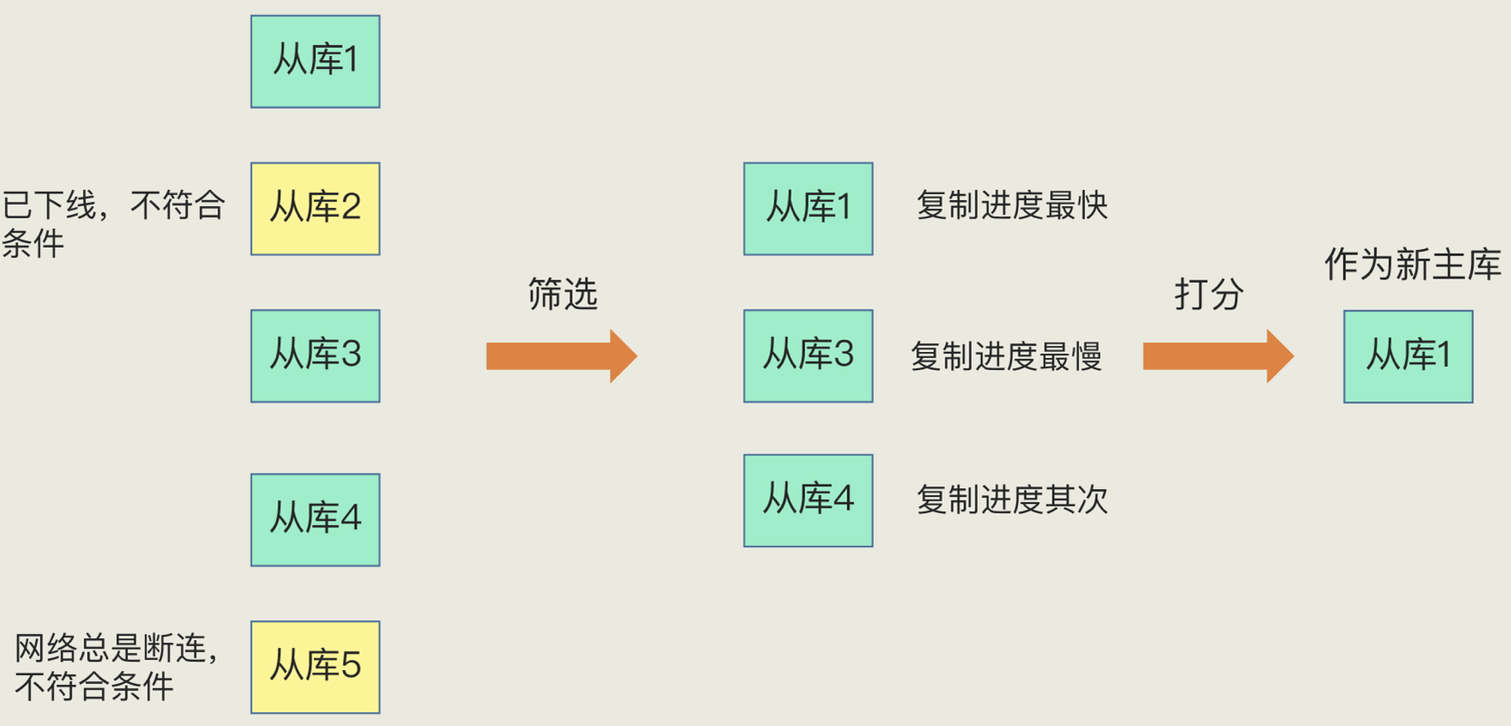
哨兵机制通常会采用多实例组成的集群模式进行部署，这也被称为哨兵集群。引入多个哨兵实例一起来判断，就可以避免单个哨兵因为自身网络状况不好，而误判主库下线的情况。同时，多个哨兵的网络同时不稳定的概率较小，由它们一起做决策，误判率也能降低。

在判断主库是否下线时，不能由一个哨兵说了算，只有大多数的哨兵实例，都判断主库已经“主观下线”了，主库才会被标记为“客观下线”，这个叫法也是表明主库下线成为一个客观事实了。这个判断原则就是：少数服从多数。同时，这会进一步触发哨兵开始主从切换流程。



1.3 选定新主库

哨兵选择新主库的过程称为“筛选+打分”。简单来说，在多个从库中，先按照一定的筛选条件，把不符合条件的从库去掉。然后，再按照一定的规则，给剩下的从库逐个打分，将得分最高的从库选为新主库，如下图所示：



- [ ] 在选主时，除了要检查从库的当前在线状态，还要判断它之前的网络连接状态。使用配置项down-after-milliseconds \* 10。其中，down-after-milliseconds是认定主从库断连的最大连接超时时间。如果在down-after-milliseconds毫秒内，主从节点都没有通过网络联系上，我们就可以认为主从节点断连了。如果发生断连的次数超过了10次，就说明这个从库的网络状况不好，不适合作为新主库。

- [ ] 剩余的从库打分：分别按照三个规则依次进行三轮打分，这三个规则分别是从库优先级、从库复制进度以及从库ID号。

第一轮：优先级最高的从库得分高。

用户可以通过slave-priority配置项，给不同的从库设置不同优先级。

第二轮：和旧主库同步程度最接近的从库得分高。

主从库同步时有个命令传播的过程。在这个过程中，主库会用master\_repl\_offset记录当前的最新写操作在repl\_backlog\_buffer中的位置，而从库会用slave\_repl\_offset这个值记录当前的复制进度。如果在所有从库中，有从库的slave\_repl\_offset最接近master\_repl\_offset，那么它的得分就最高，可以作为新主库。

第三轮：ID号小的从库得分高。

每个实例都会有一个ID，这个ID就类似于这里的从库的编号。目前，Redis在选主库时，有一个默认的规定：在优先级和复制进度都相同的情况下，ID号最小的从库得分最高，会被选为新主库。

二、哨兵集群

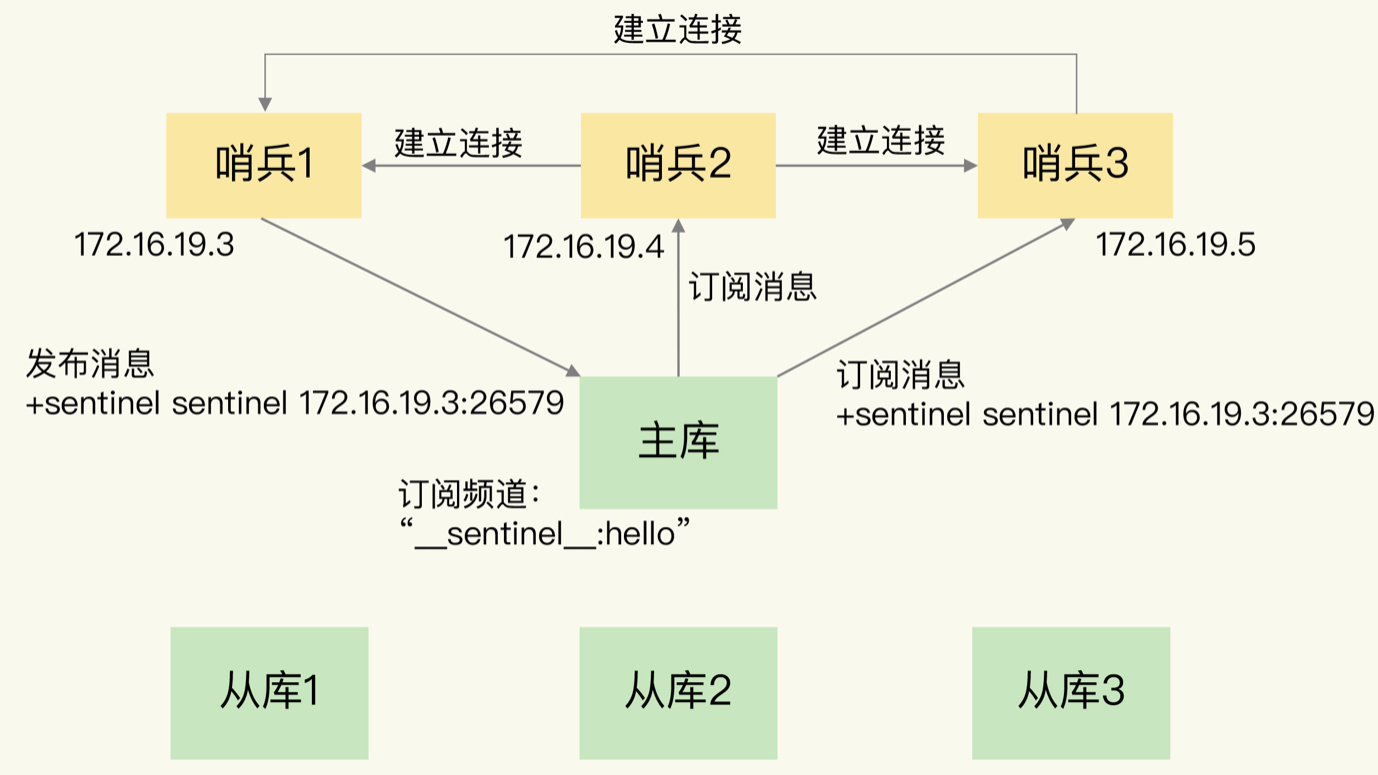
2.1 基于pub/sub（发布/订阅）机制的哨兵集群组成

哨兵只要和主库建立起了连接，就可以在主库上发布消息了，比如说发布它自己的连接信息（IP和端口）。同时，它也可以从主库上订阅消息，获得其他哨兵发布的连接信息。当多个哨兵实例都在主库上做了发布和订阅操作后，它们之间就能知道彼此的IP地址和端口。

除了哨兵实例，我们自己编写的应用程序也可以通过Redis进行消息的发布和订阅。所以，为了区分不同应用的消息，Redis会以频道的形式，对这些消息进行分门别类的管理。

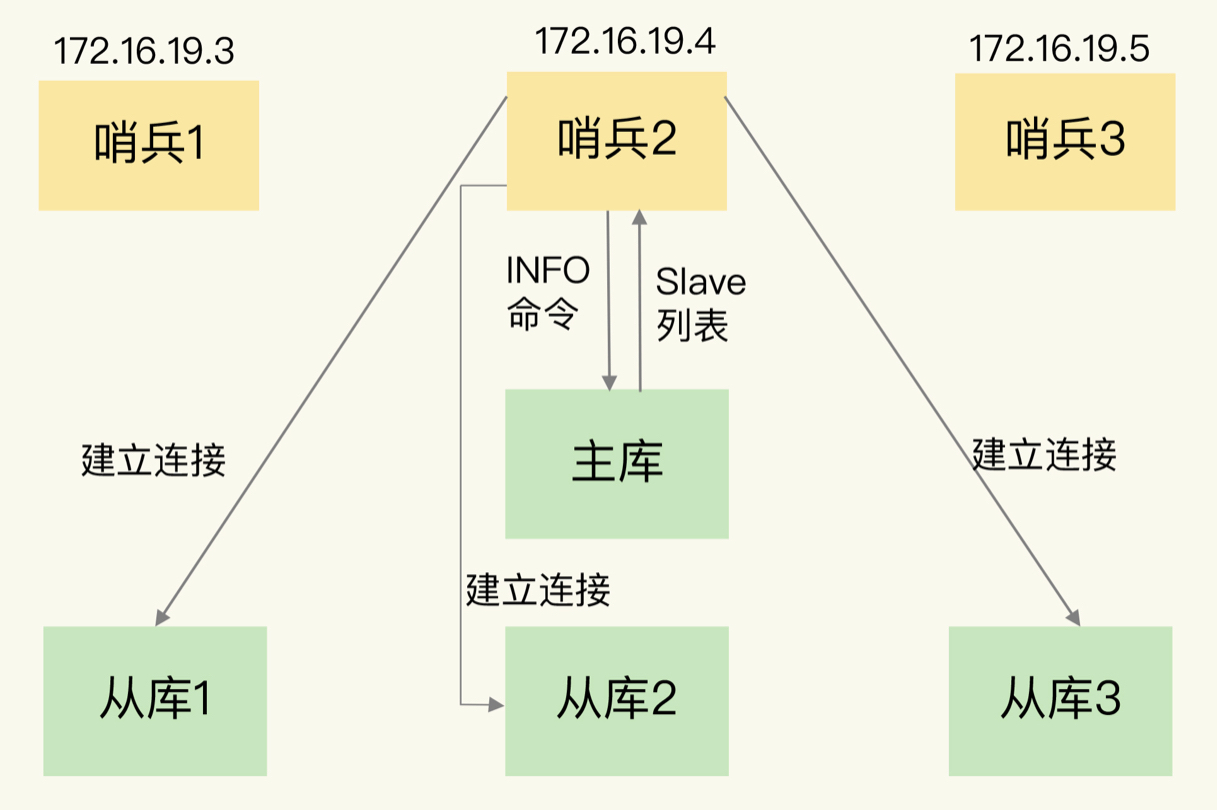
所谓的频道，实际上就是消息的类别。当消息类别相同时，它们就属于同一个频道。反之，就属于不同的频道。只有订阅了同一个频道的应用，才能通过发布的消息进行信息交换。

在主从集群中，主库上有一个名为“\_\_sentinel\_\_:hello”的频道，不同哨兵就是通过它来相互发现，实现互相通信的。



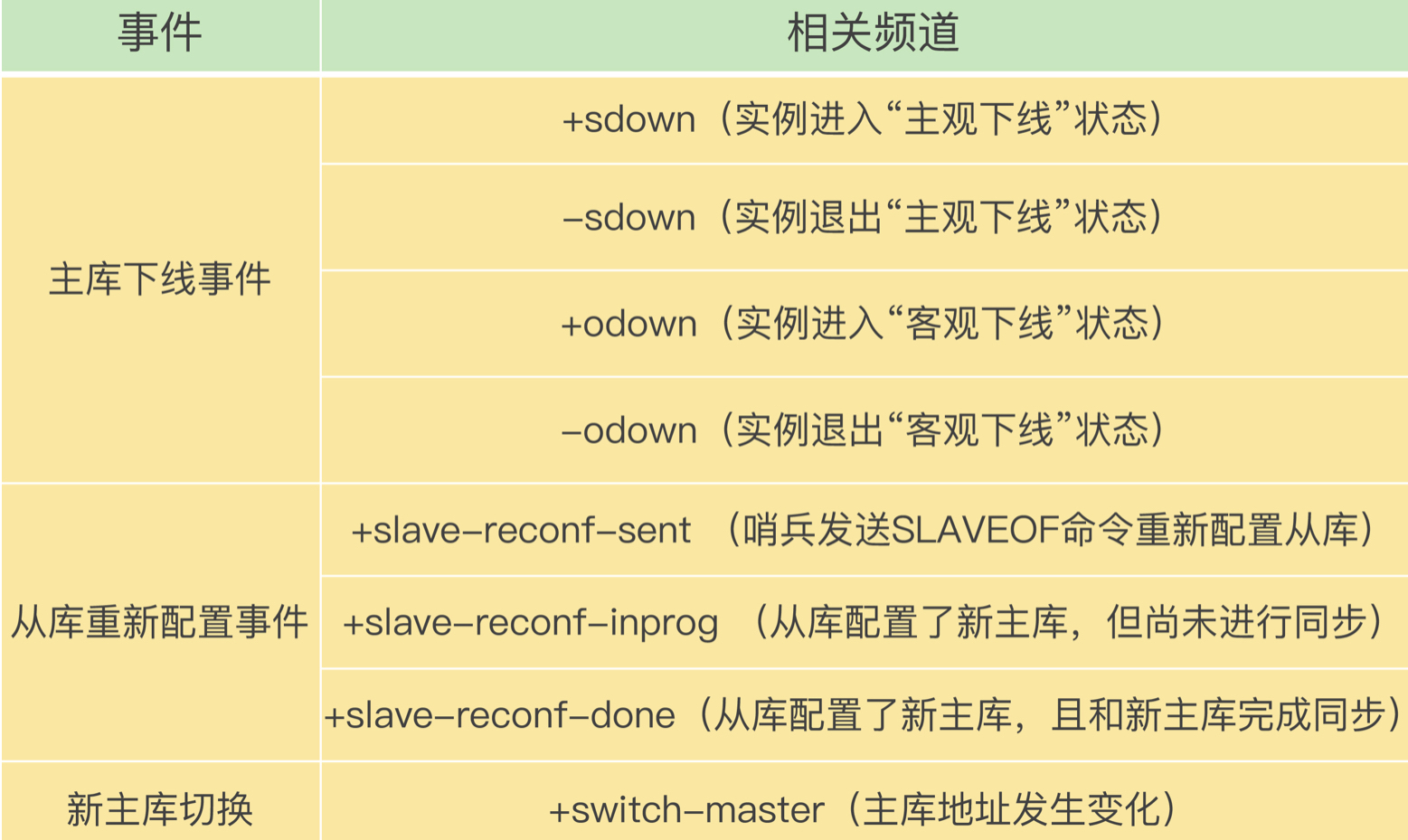
哨兵除了彼此之间建立起连接形成集群外，还需要和从库建立连接。这是因为，在哨兵的监控任务中，它需要对主从库都进行心跳判断，而且在主从库切换完成后，它还需要通知从库，让它们和新主库进行同步。

由哨兵向主库发送INFO命令来完成。就像下图所示，哨兵2给主库发送INFO命令，主库接受到这个命令后，就会把从库列表返回给哨兵。接着，哨兵就可以根据从库列表中的连接信息，和每个从库建立连接，并在这个连接上持续地对从库进行监控。哨兵1和3可以通过相同的方法和从库建立连接。



2.2 基于pub/sub机制的客户端事件通知

从本质上说，哨兵就是一个运行在特定模式下的Redis实例，只不过它并不服务请求操作，只是完成监控、选主和通知的任务。所以，每个哨兵实例也提供pub/sub机制，客户端可以从哨兵订阅消息。哨兵提供的消息订阅频道有很多，不同频道包含了主从库切换过程中的不同关键事件。



客户端读取哨兵的配置文件后，可以获得哨兵的地址和端口，和哨兵建立网络连接。然后，在客户端执行订阅命令，来获取不同的事件消息。

举个例子，

订阅“所有实例进入客观下线状态的事件”：

SUBSCRIBE +odown

订阅所有的事件：

PSUBSCRIBE \*

当哨兵把新主库选择出来后，客户端就会看到下面的switch-master事件。这个事件表示主库已经切换了，新主库的IP地址和端口信息已经有了。这个时候，客户端就可以用这里面的新主库地址和端口进行通信了。

switch-master <master name> <oldip> <oldport> <newip> <newport>

有了这些事件通知，客户端不仅可以在主从切换后得到新主库的连接信息，还可以监控到主从库切换过程中发生的各个重要事件。这样，客户端就可以知道主从切换进行到哪一步了，有助于了解切换进度。

2.3 哨兵执行主从切换

- [ ] 任何一个实例只要自身判断主库“主观下线”后，就会给其他实例发送is-master-down-by-addr命令。接着，其他实例会根据自己和主库的连接情况，做出Y或N的响应，Y相当于赞成票，N相当于反对票。

- [ ] 一个哨兵获得了仲裁所需的赞成票数后，就可以标记主库为“客观下线”。这个所需的赞成票数是通过哨兵配置文件中的quorum配置项设定的。

- [ ] 此时，这个哨兵就可以再给其他哨兵发送命令，表明希望由自己来执行主从切换，并让所有其他哨兵进行投票。这个投票过程称为“Leader选举”。因为最终执行主从切换的哨兵称为Leader，投票过程就是确定Leader。在投票过程中，任何一个想成为Leader的哨兵，要满足两个条件：第一，拿到半数以上的赞成票；第二，拿到的票数同时还需要大于等于哨兵配置文件中的quorum值。

哨兵集群能够进行成功投票，很大程度上依赖于选举命令的正常网络传播。如果网络压力较大或有短时堵塞，就可能导致没有一个哨兵能拿到半数以上的赞成票。所以，等到网络拥塞好转之后，再进行投票选举，成功的概率就会增加。

经过投票仲裁，选举一个Leader出来，由它负责实际的主从切换，即由它来完成新主库的选择以及通知从库与客户端。