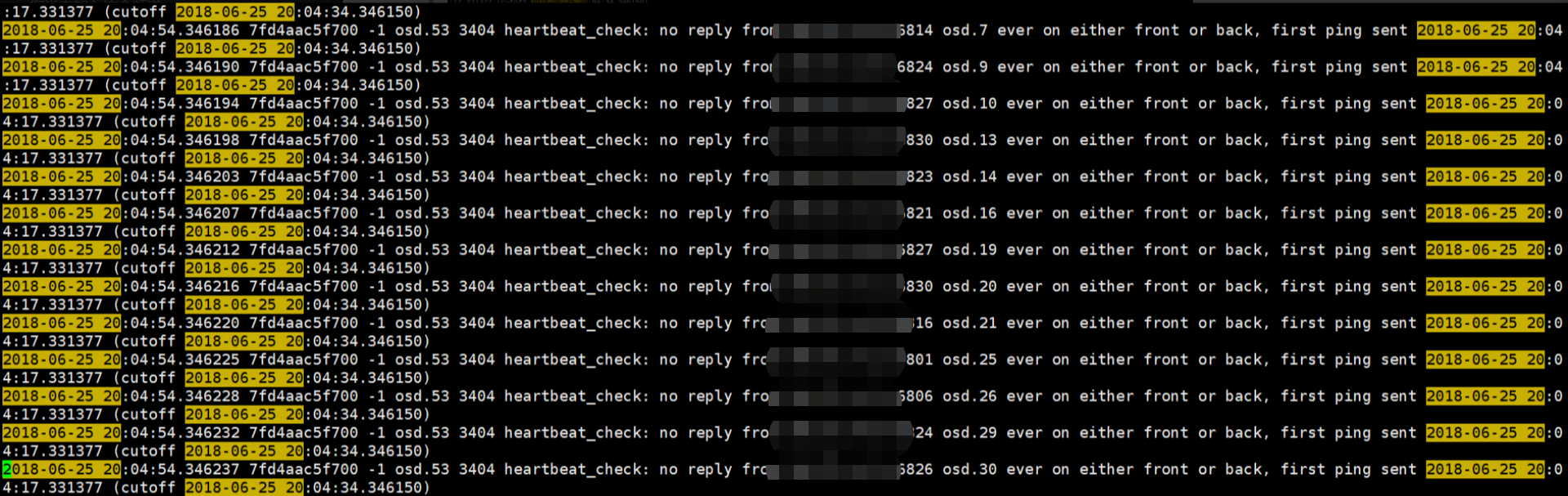
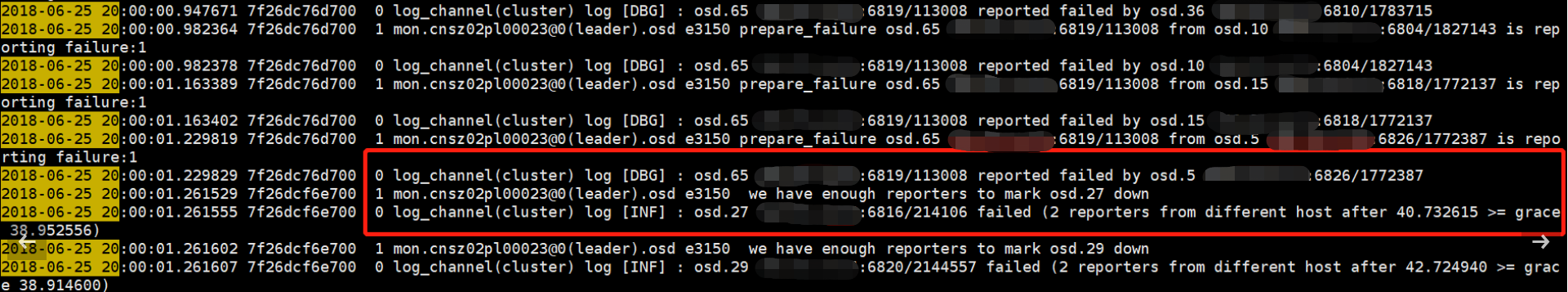
最近在维护ceph集群时发现有时osd时而UP时而Down, 发现是由于网络不稳定导致OSD之间不能及时的心跳汇报，导致误认为对方down掉了，故而想monitor报告OSD挂掉了。





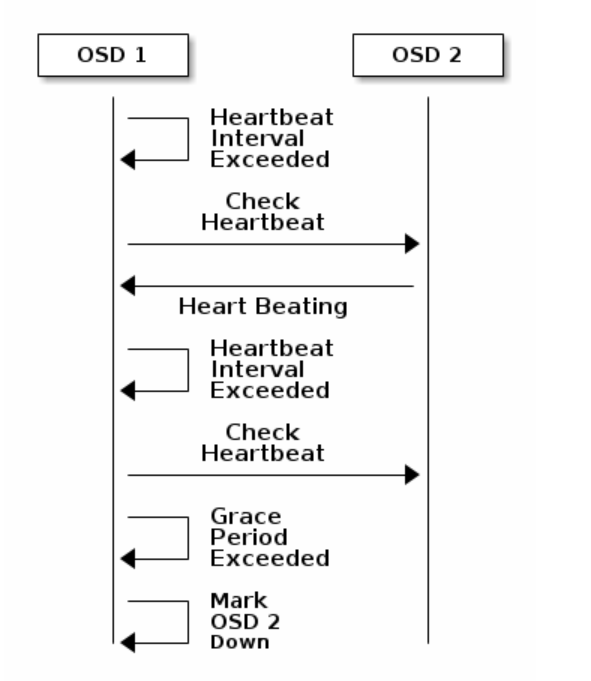
本章将讲解ceph之心跳检测实现原理。

ceph的心跳检测原理，官方文档写的很详细：

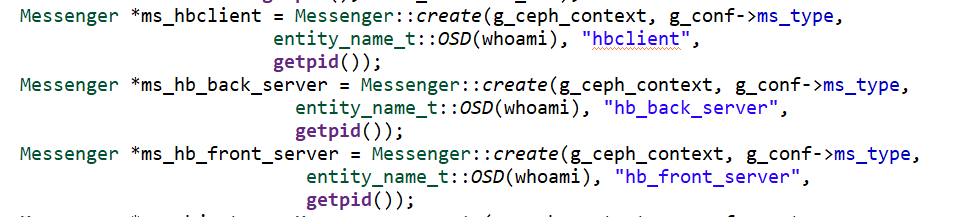
<http://docs.ceph.com/docs/master/rados/configuration/mon-osd-interaction/>

本文从代码级别去解释其实现过程：

osd 与 osd 之间的心跳：



在OSD启动时会建立3个创造了三个messenger用于心跳通信

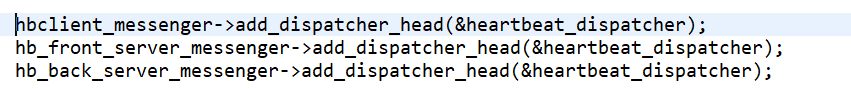


其中ms\_hbclient用来发送ping心跳

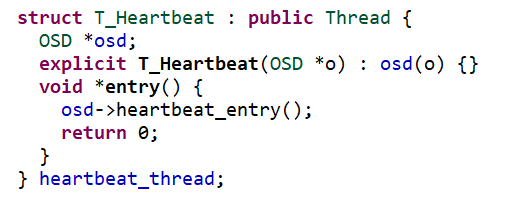
ms\_hb\_back\_server 用于接收来自back地址的ping心跳，即cluster network地址的心跳

ms\_hb\_front\_server 用于接收front地址的ping心跳。即public network地址的心跳。

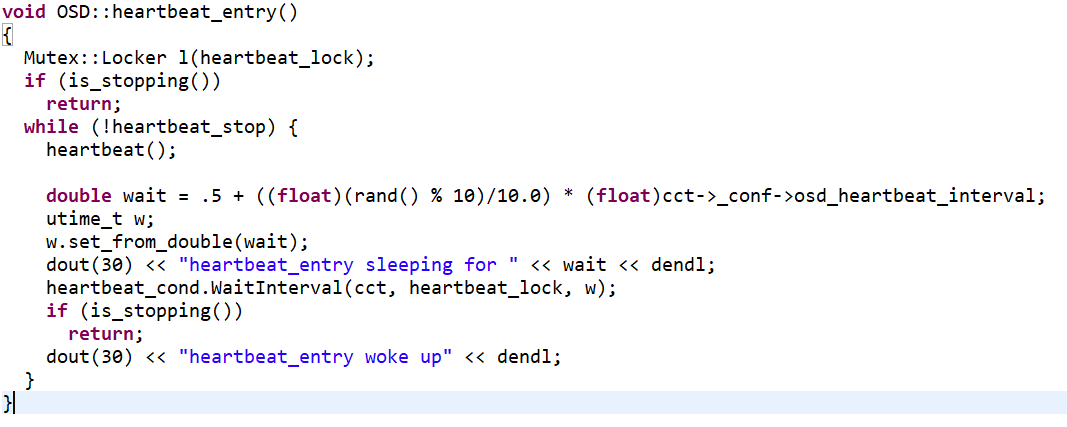
然后注册heartbeat\_dispatcher



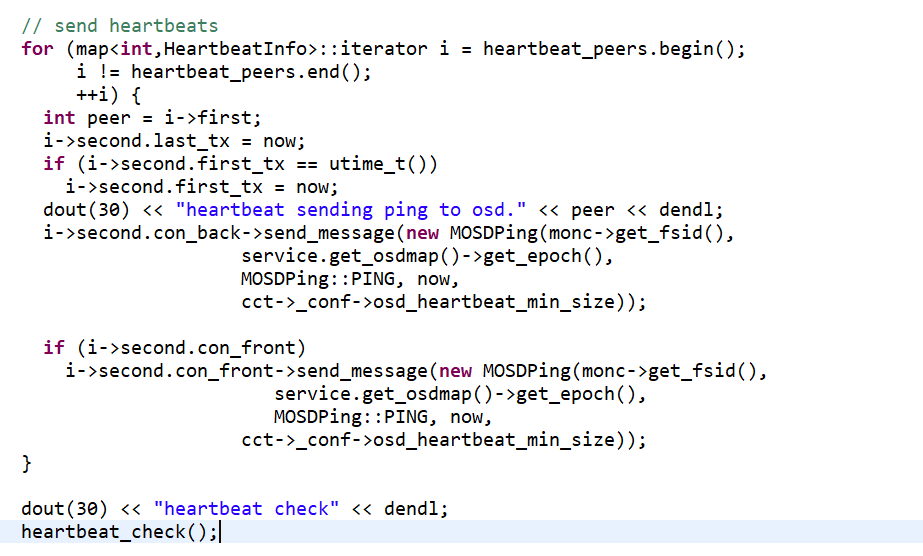
OSD会启动一个独立的线程用于定时向其他OSD发送ping 信息，



heartbeat\_entry()函数定时（0.5~6.5）秒调用heartbeat()发送ping信息

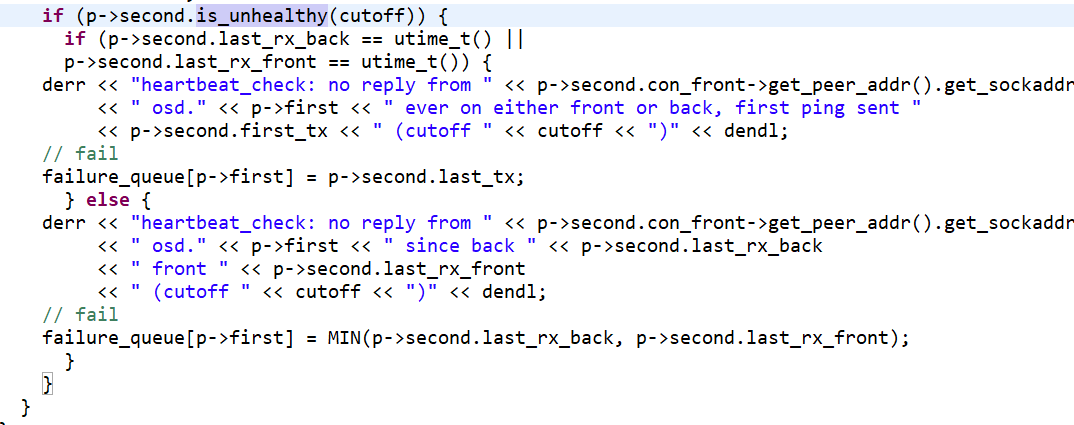


heartbeat()函数就是选择osd发送ping消息：



代码可以看到其分别向back地址和front地址发送ping消息。

同时发送消息后会检查一下是否超时：heartbeat\_check

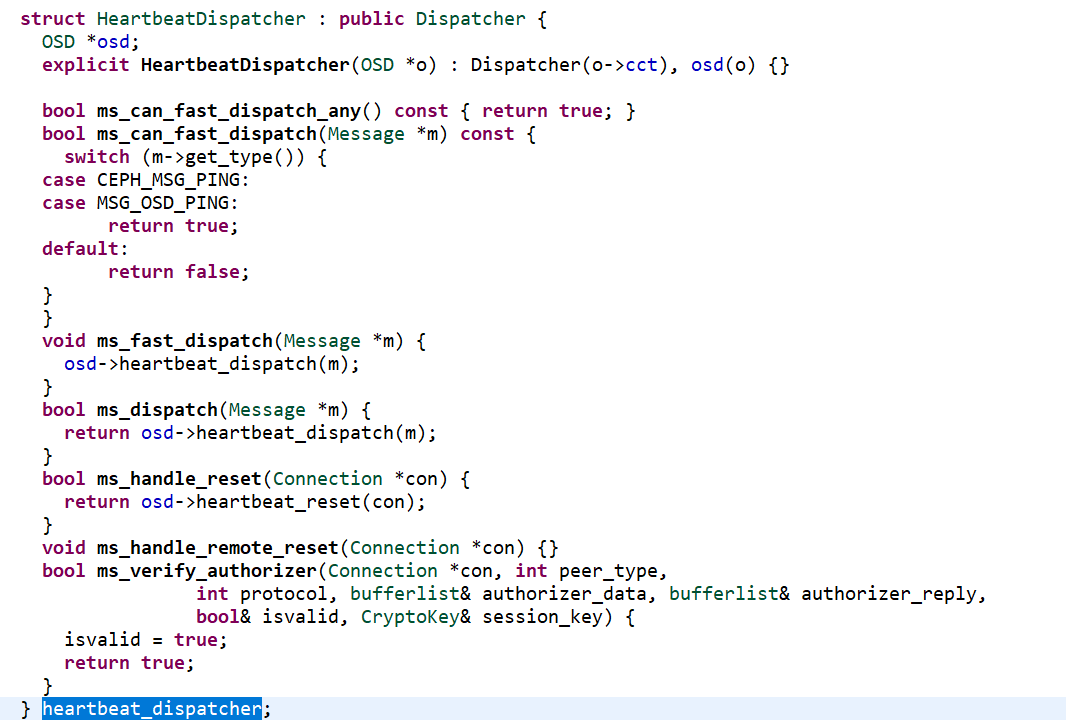


检测超时就加入failure\_queue，等待上报给monitor

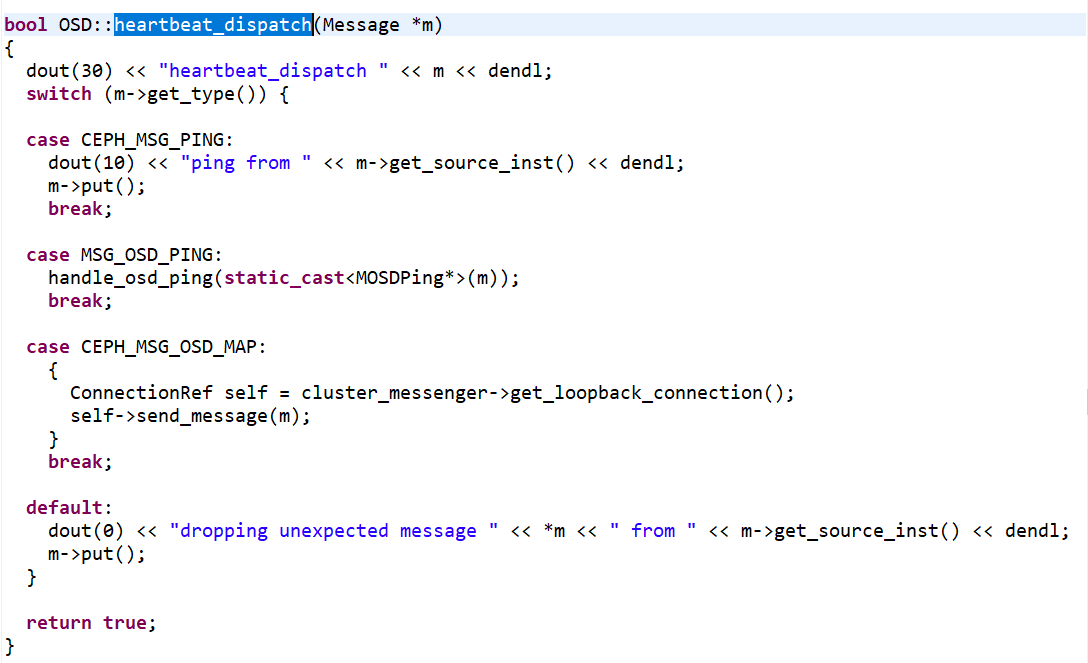
心跳是否超时的检查，另外还有专门的tick线程，也会检查心跳是否超时



上面讲到心跳消息的发送，那对于心跳消息的接收则有专门的注册的dispatcher类处理：



ms\_dispatch(Message \*m)处理:

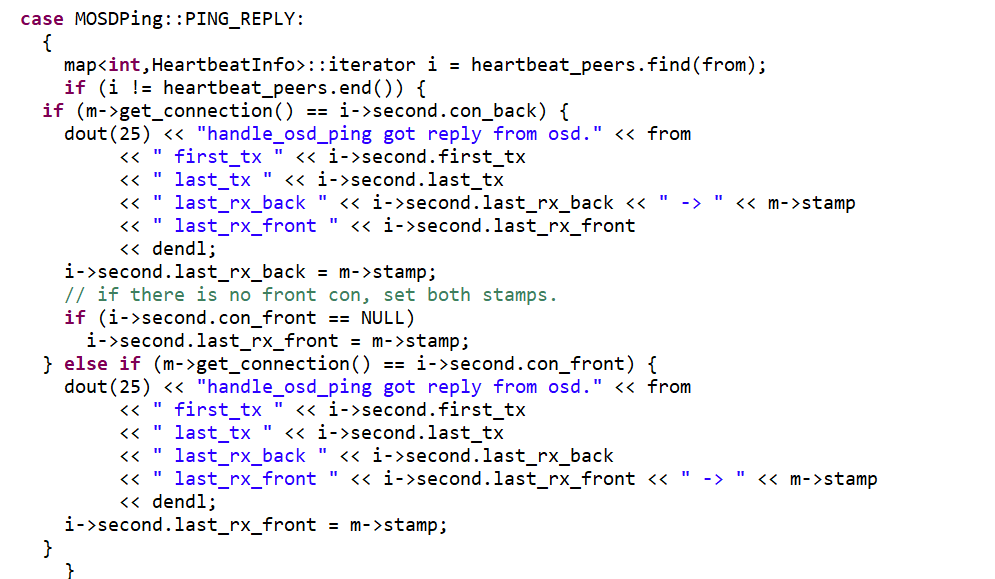


在分支MSG\_OSD\_ PING处理收到的ping消息：

处理心跳信息：

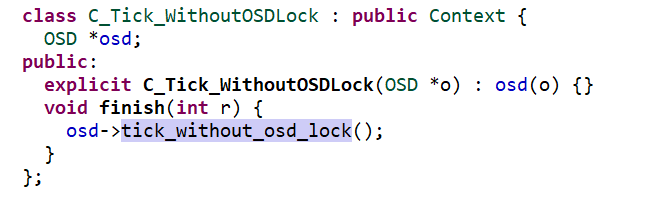


处理心跳回包：

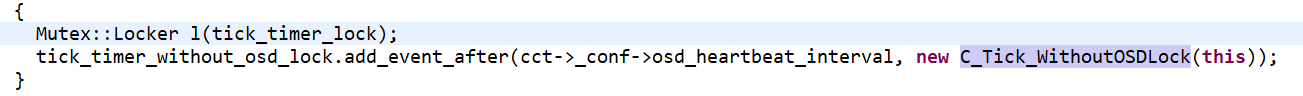


更新时间戳。判断是否超时交由专门的定时检查heartbeat\_check，上面已经讲述了。

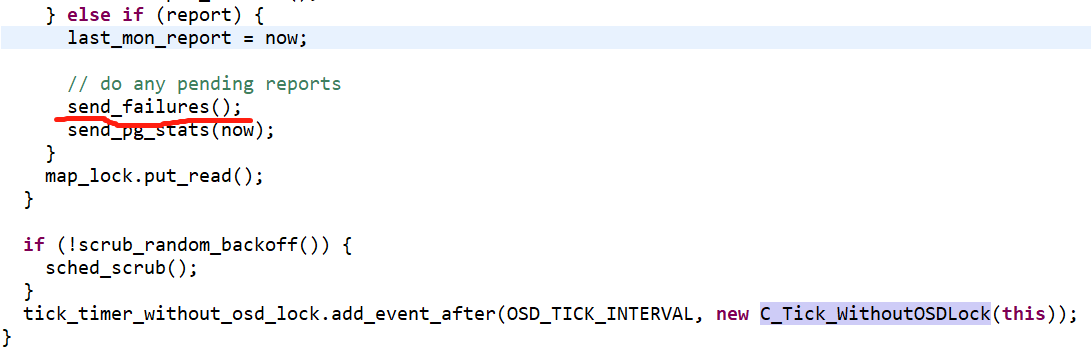
而对于向monitor心跳超时上报的时候，也是在tick线程内完成

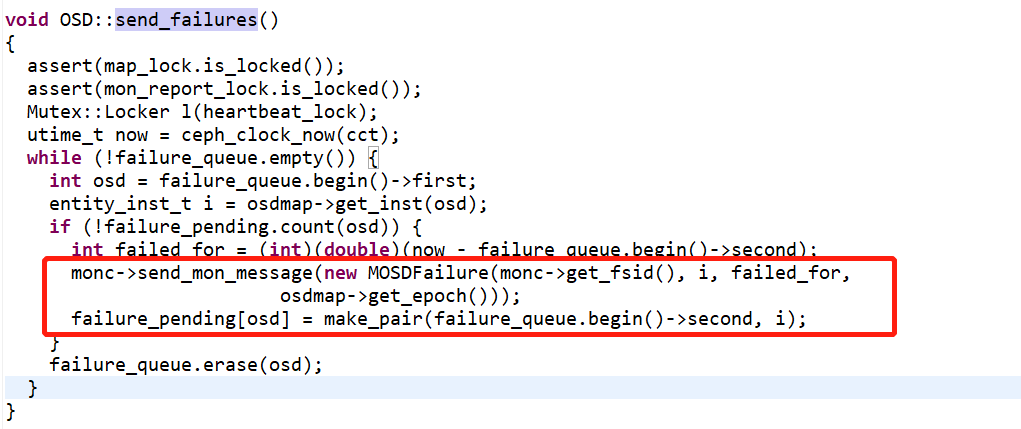


osd有专门的定时器负责检测心跳上报超时检查。



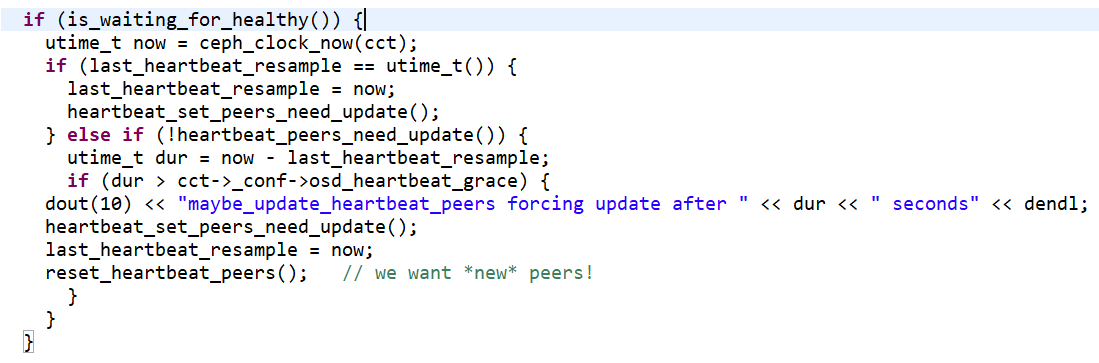
在tick\_without\_osd\_lock()中send\_failures就是向monitor上报OSD状态。





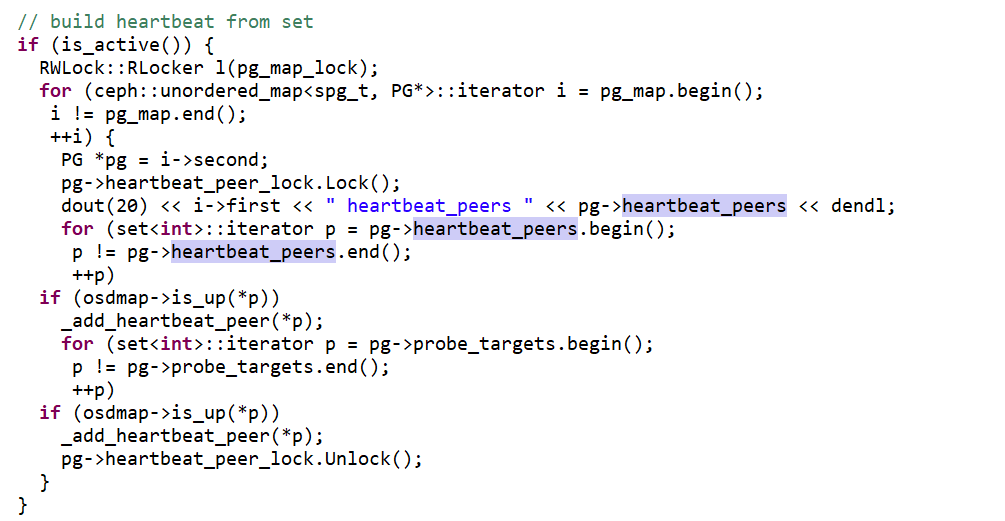
至于OSD向哪些OSD peer发送心跳包，则有函数maybe\_update\_heartbeat\_peers负责选择

在osd启动的过程中，或者在osd收到更新osdmap的消息，osd状态可能变为waiting，此时需要更新peers集合，reset\_heartbeat\_peers()就是清空当前map中保存的peers.

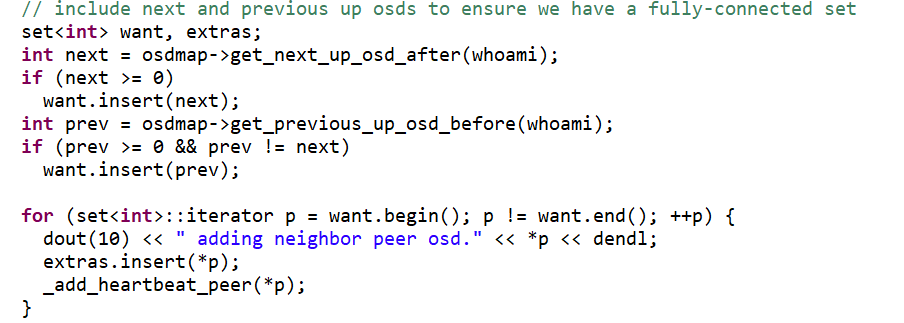


遍历osd负责的所有pg, 遍历pg对应的peers, 如果为up，则加入心跳集合

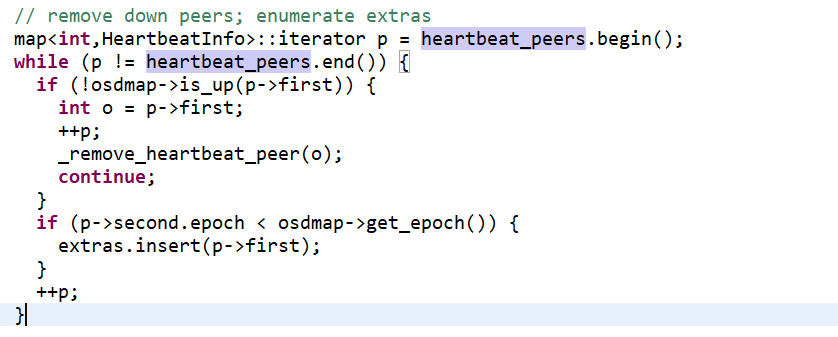
遍历probe目标集合, 如果为up，则加入心跳集合



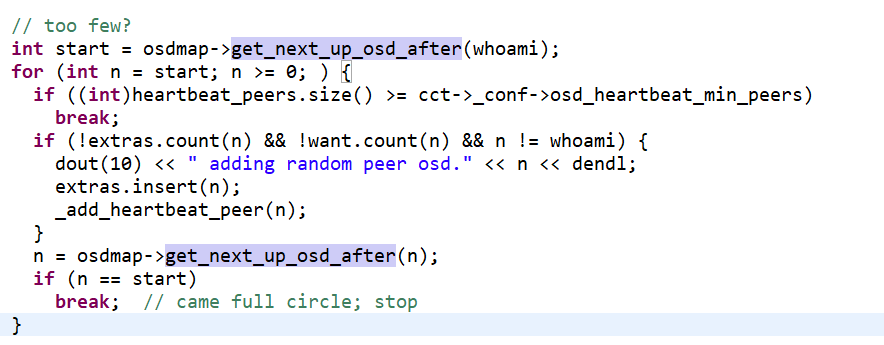
将邻居的OSD也加入到心跳当中。



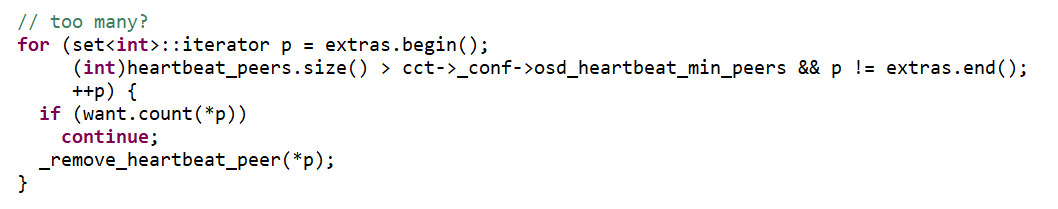
删除down 节点



若加入心跳的节点数不够，则从比当前OSD编号大的OSD加入map中：



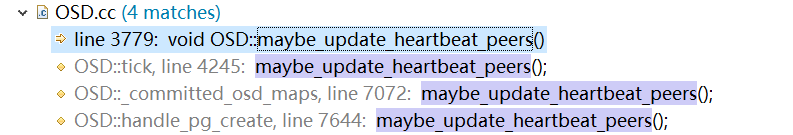
若加入的心跳节点数太多，则删除一些，删除邻居或者osd osdmap过时的或者通过get\_next\_up\_osd\_after()加入的。



至于什么时候更新osd peers,主要在3个时刻：  
pg创建的时候；

osdmap变更的时候；

tick定时器周期性检查；



总结一下就是，osd启动或者异常退出，monitor会收到消息，然后进行paxos，将结果会反应在osdmap上，进而通知相关osd进程， osd进程收到消息后，会处理map的变更，可能导致pg重新peering。monitor也会收到创建pool或修改pg\_num的消息，最终会导致创建pg， osd收到消息创建pg，也会导致peering。osd启动的过程中，load\_pg也会导致peering，一旦有peering发生，osd进程的状态就是STATE\_WAITING\_FOR\_HEALTHY， 就可能导致更新peer集合。