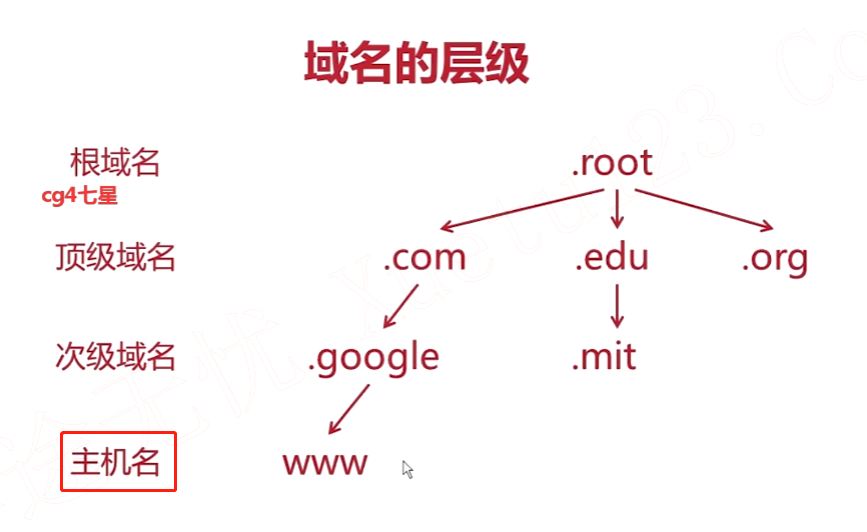




省略了根域名



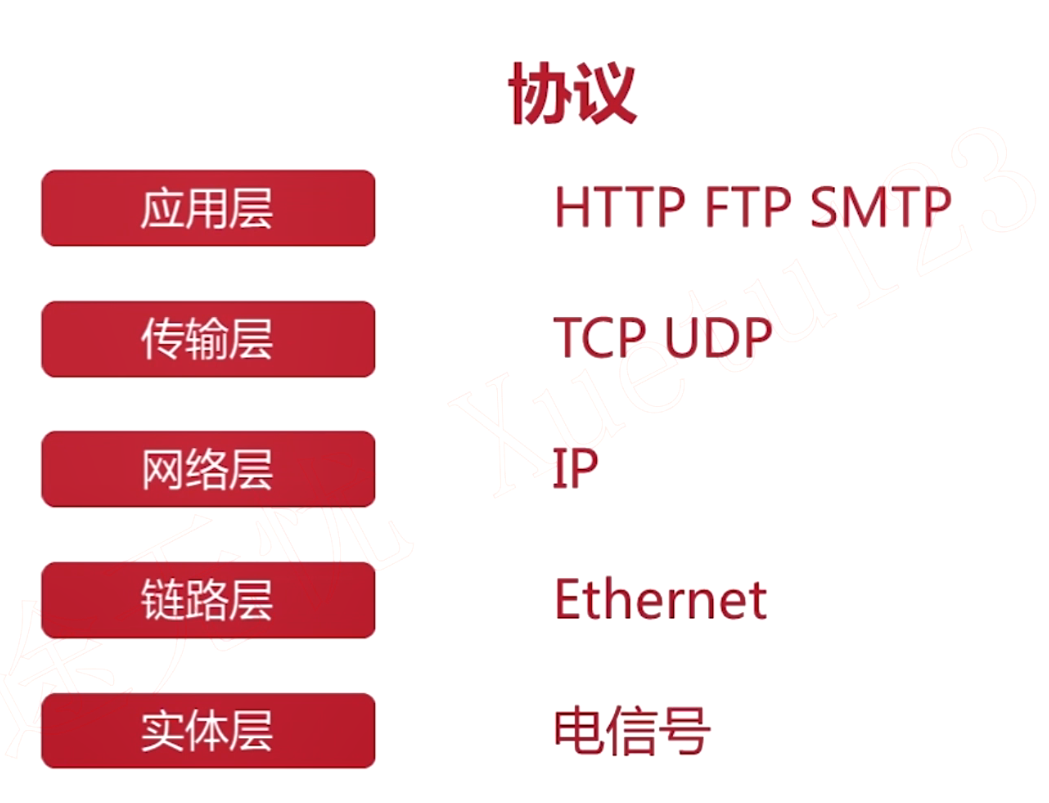


服务器的程序到客户端被解析

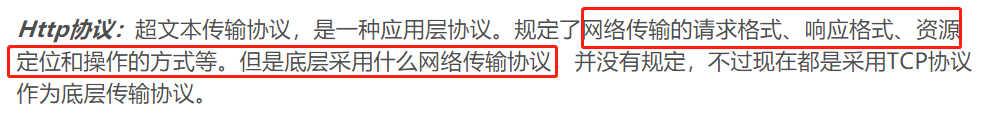
应用层（用户能接触到的应用对数据的处理方法和格式）->传输层（端口到端口）->网络层（主机到主机）->链路层（网卡到网卡）->实体层（光缆，物理设备）的联系

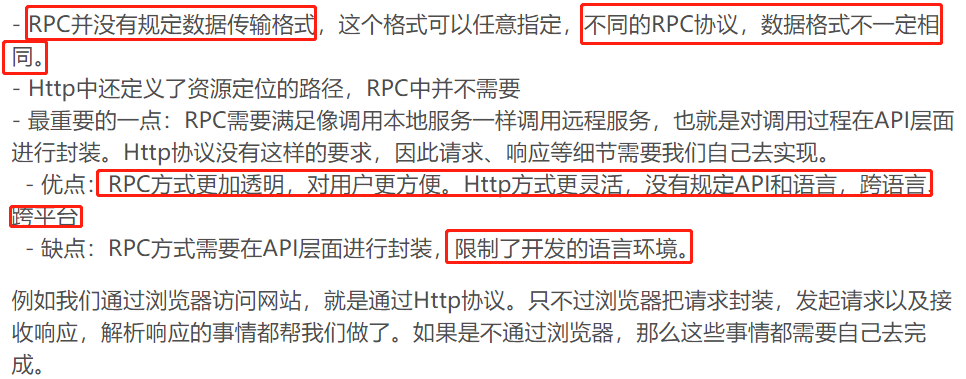
每一层只需要依赖于下一层，只需要适配于下一层（改动）

分层模型的好处是将影响降低到最小

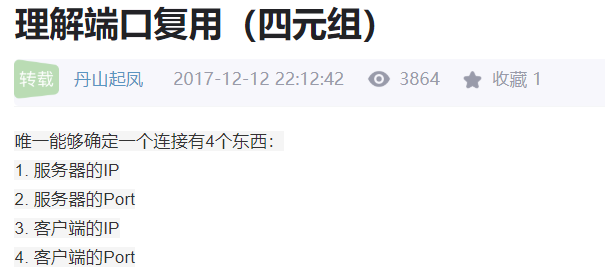


越往下离软件越远离硬件越近

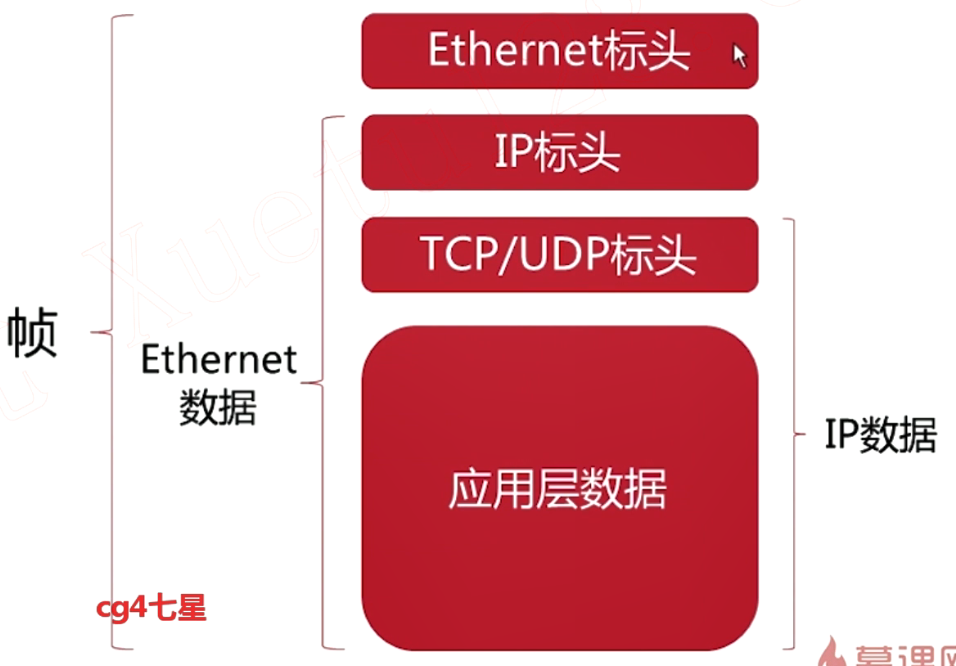




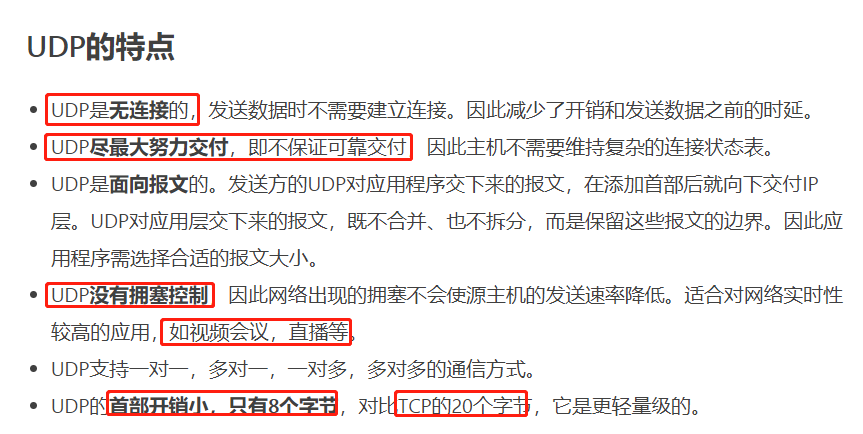
浏览器网页一般http，ftp客户端ftp协议，邮件就是smtp协议

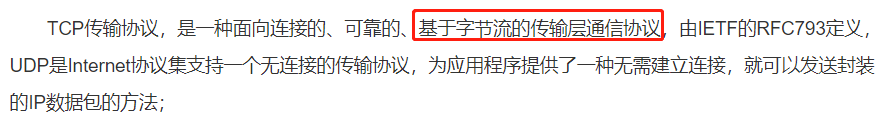


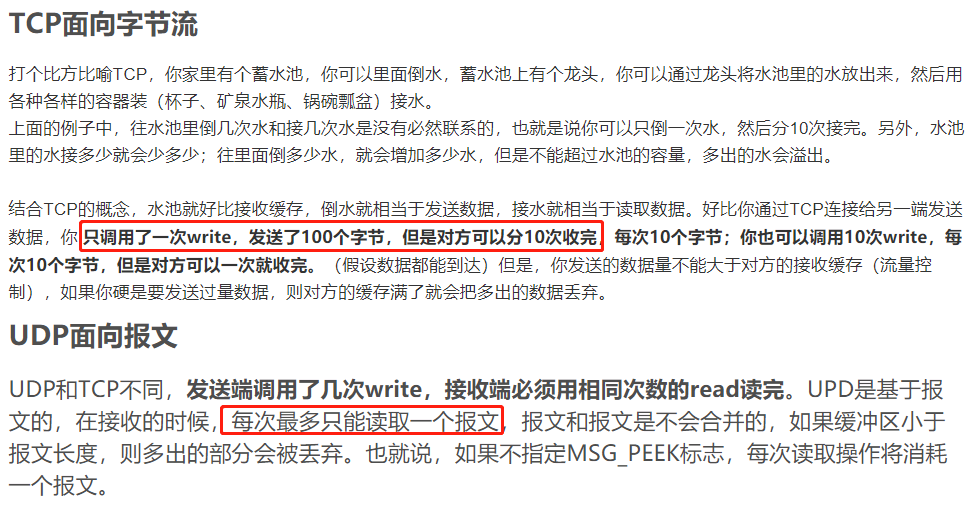


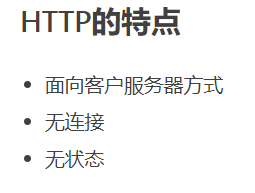


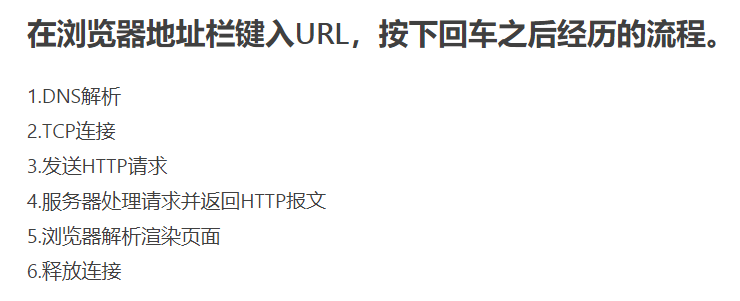


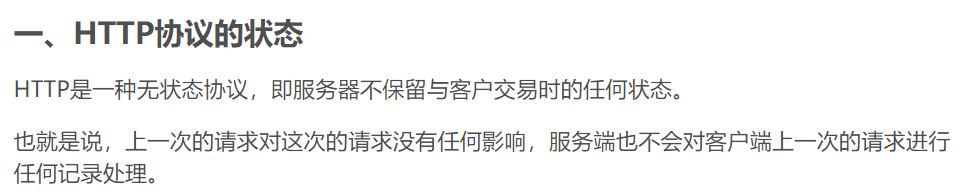


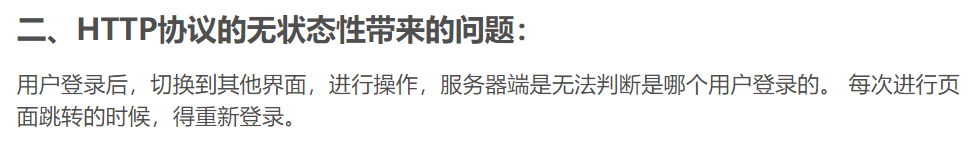




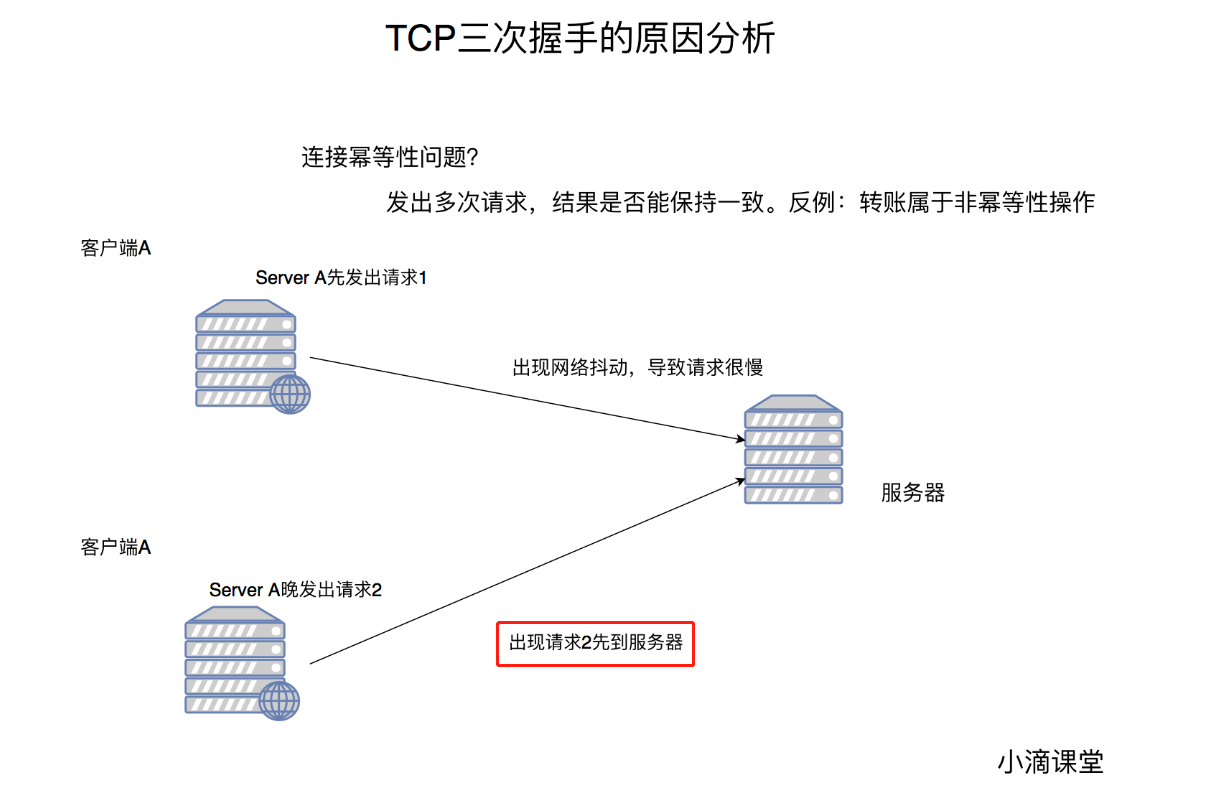








<https://blog.csdn.net/H_Expect/article/details/99967490>



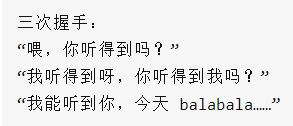


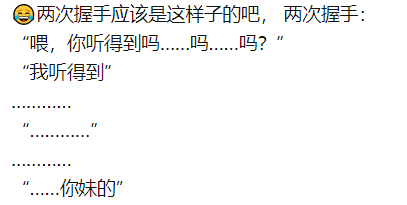
TCP作为一种可靠传输控制协议，其核心思想：既要保证数据可靠传输，又要提高传输的效率，而用三次恰恰可以满足以上两方面的需求！

三次握手机制用于解决了报文错乱的问题

A发出请求，网络原因一直没到达B

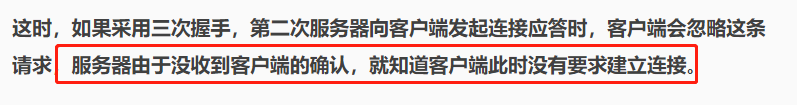
A再发了一次请求，此时，A之前的请求（脏请求）到达B了，B和它确定连接。浪费了资源

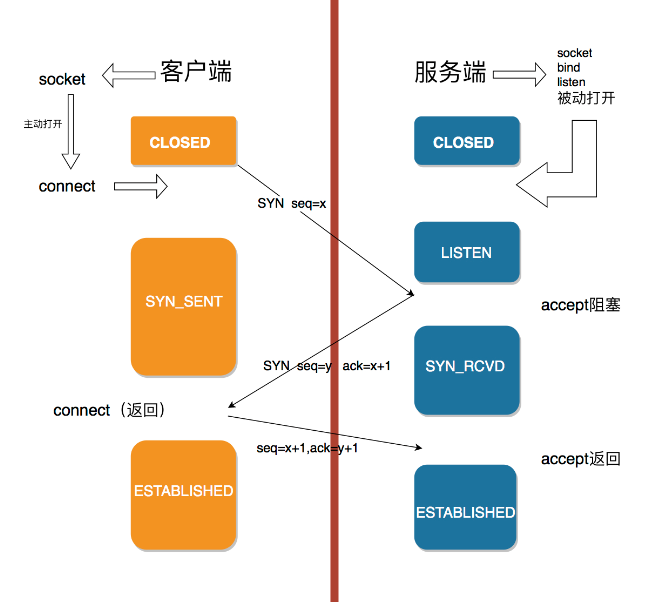


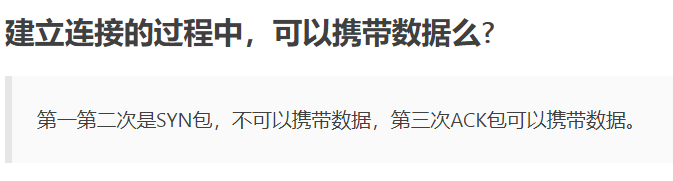


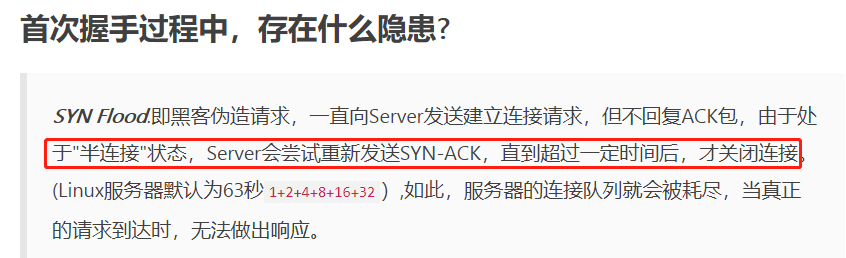
我听得到，然后就么有然后了，浪费资源

此时服务器并不能确定自己是不是能发送和客户端是否能接收。

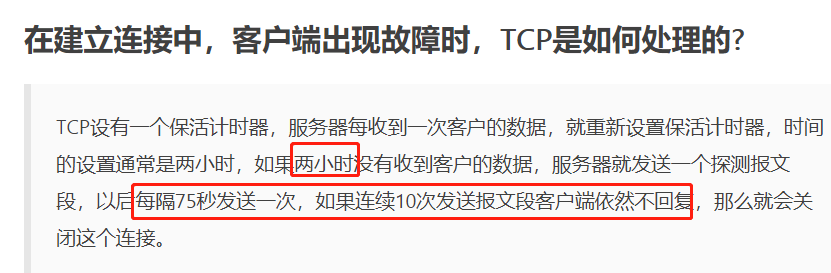


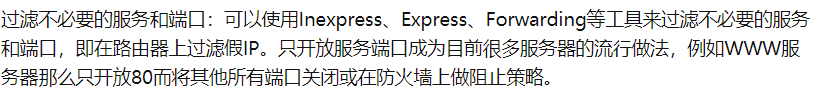


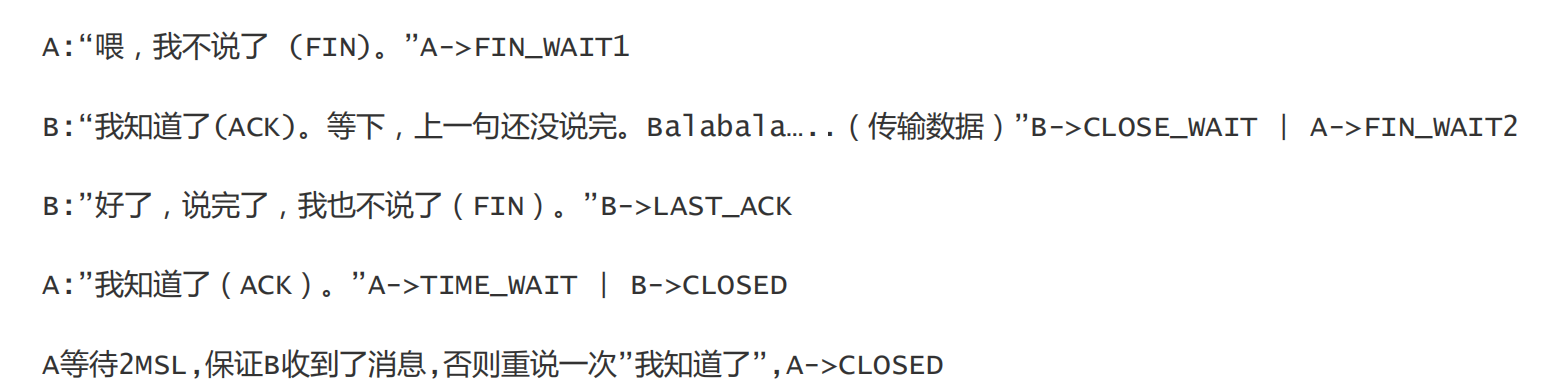


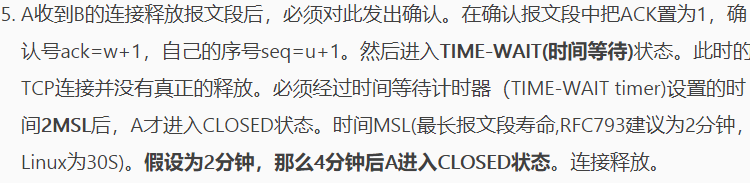


洪水攻击

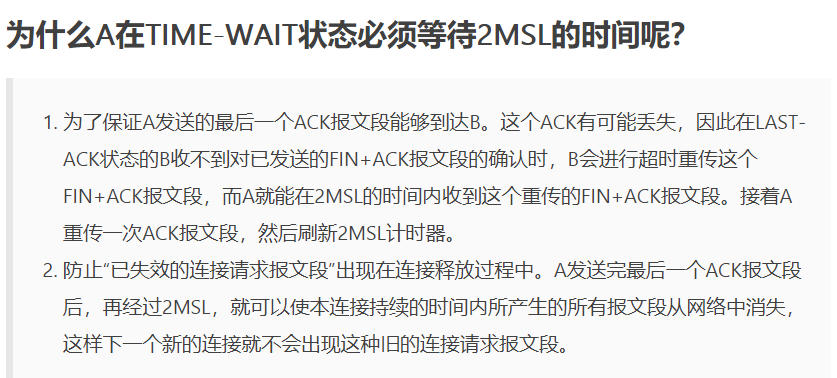


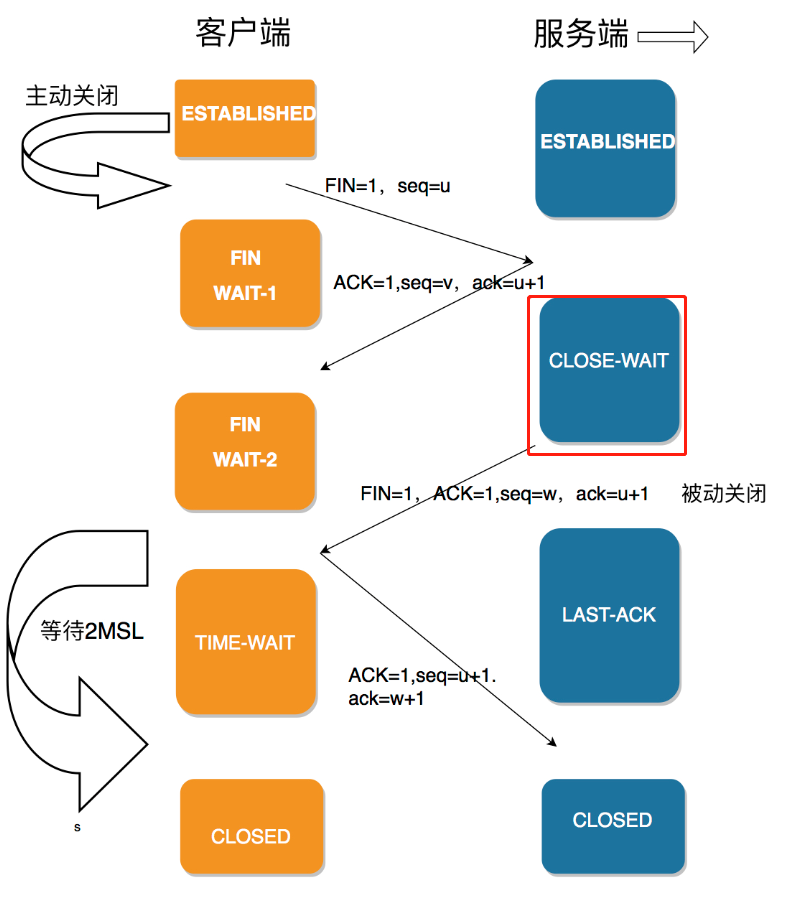






保证B收到A最后的关闭信息

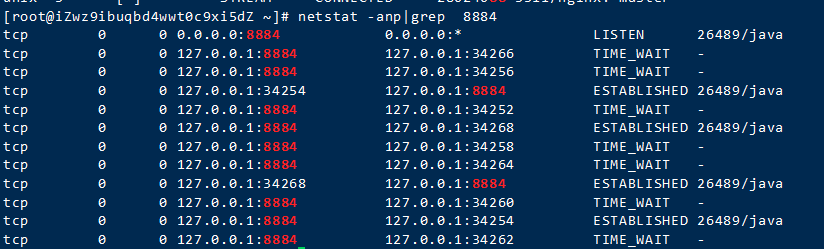




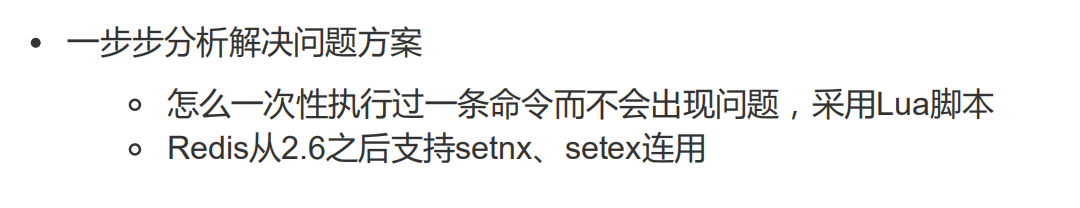
双方几乎在同时close()一个SOCKET的话，就出现了双方同时发送

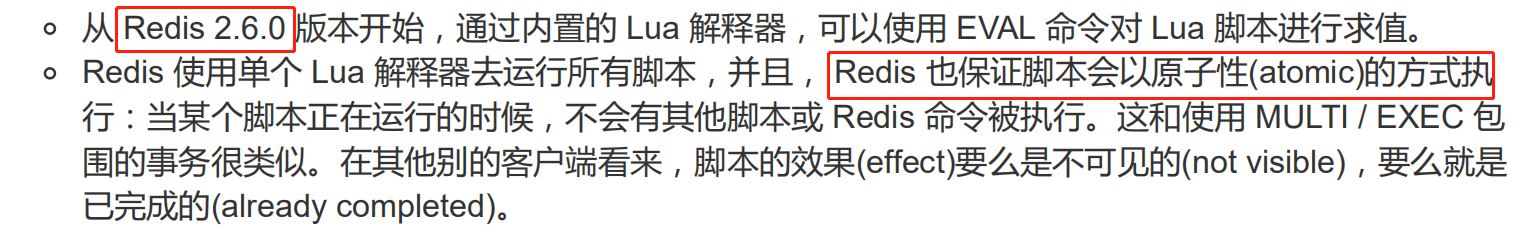
FIN报文的情况，这是就会出现CLOSING 状态，表示双方都正在关闭SOCKET连接。

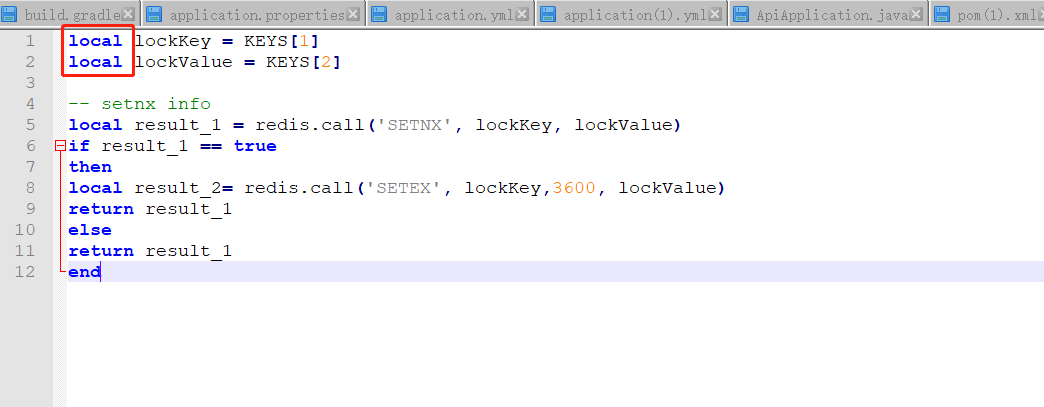




**其实return与finally并没有明显的谁强谁弱。在执行时，是return语句先把返回值写入但内存中，然后停下来等待finally语句块执行完，return再执行后面的一段**

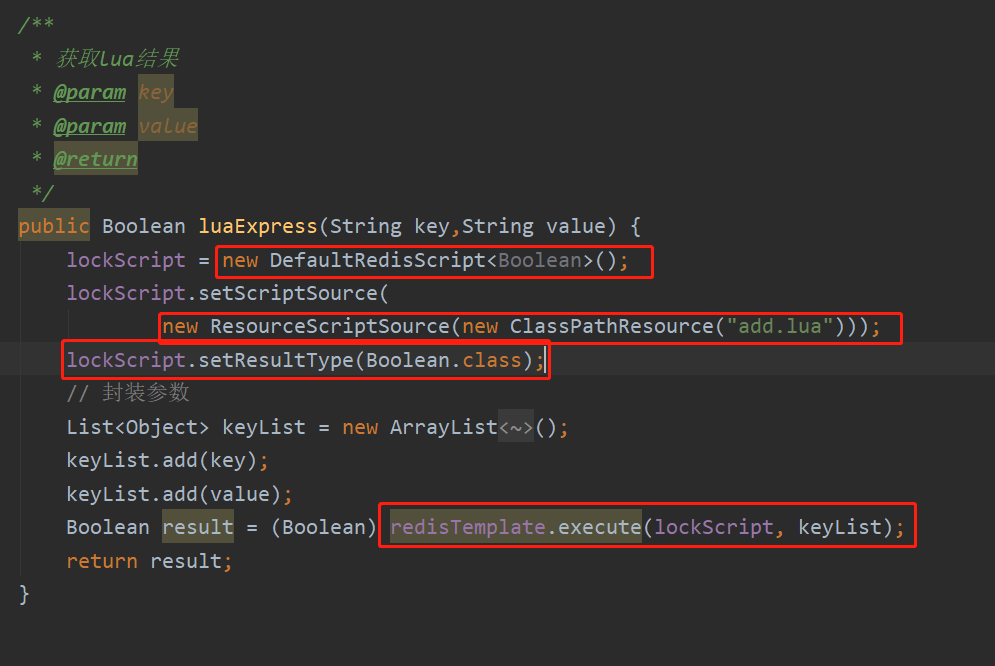






对变量的声明

redis.call（）调用redis的命令的方法

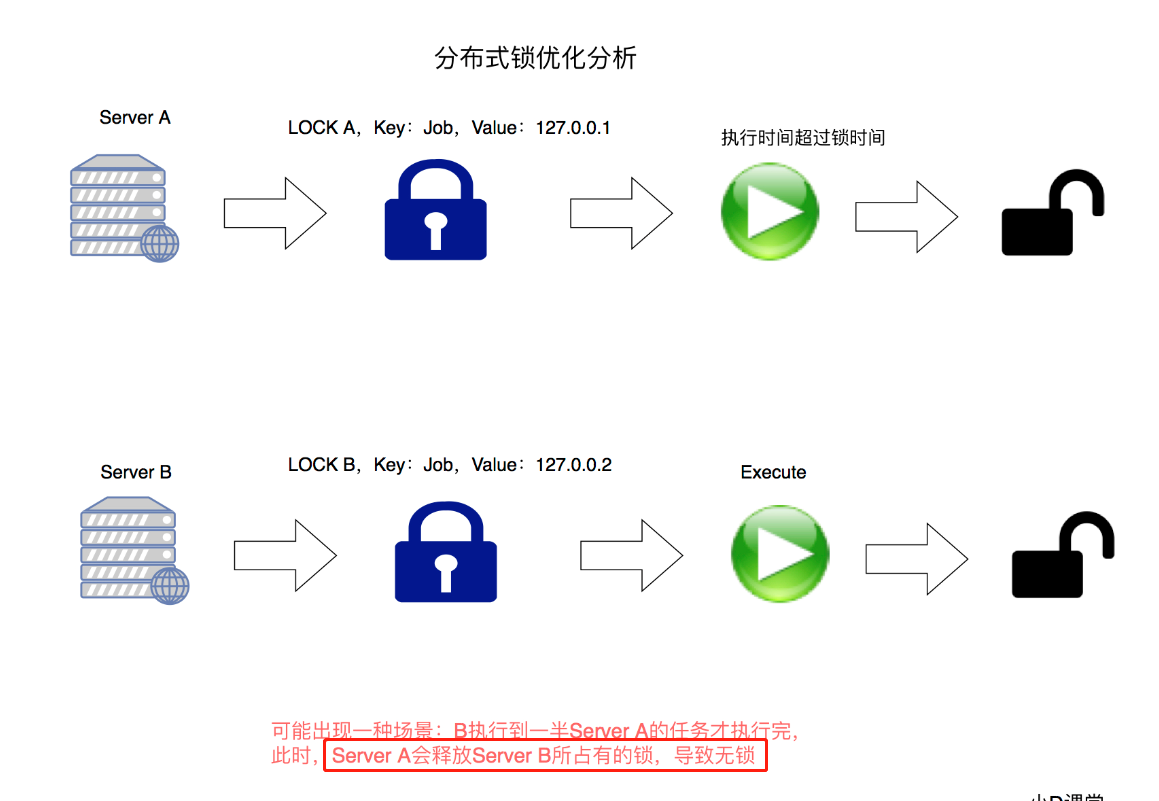


DefaultRedisScript,传入路径和设置放回值

RedisTemplate的execute执行



redis2.x的一个方法



根据key获取value看是否是自己的，不是的话就不能remove

