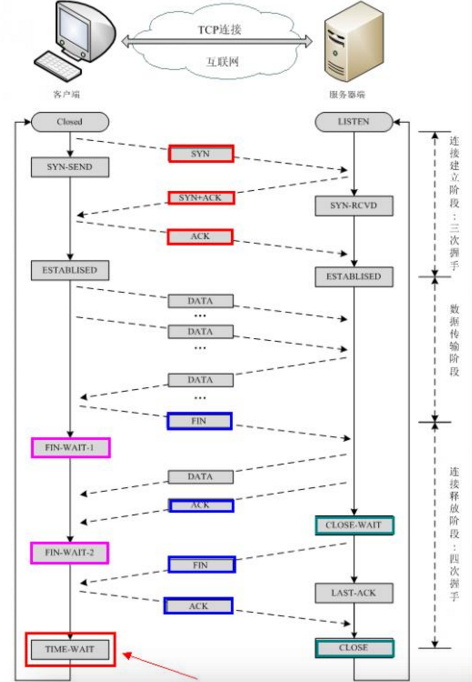
**time\_wait状态产生的原因，危害，如何避免**

请说说你对TCP连接中time\_wait状态的理解

**解答：**

先上TCP的状态变迁图



上面这个图片展示了TCP从连接建立到连接释放的过程中，客户端和服务端的状态变化图。如果只看连接释放阶段，四次握手

* 客户端先发送FIN，进入FIN\_WAIT1状态
* 服务端收到FIN，发送ACK，进入CLOSE\_WAIT状态，客户端收到这个ACK，进入FIN\_WAIT2状态
* 服务端发送FIN，进入LAST\_ACK状态
* 客户端收到FIN，发送ACK，进入TIME\_WAIT状态，服务端收到ACK，进入CLOSE状态
* 客户端TIME\_WAIT持续2倍MSL时长，在linux体系中大概是60s，转换成CLOSE状态

当然在这个例子和上面的图片中，使用客户端和服务端来描述是不准确的，TCP主动断开连接的一方可能是客户端，也可能是服务端。所以使用主动断开的一方，和被动断开的一方替换上面的图可能更为贴切。

不管怎么说，TIME\_WAIT的状态就是主动断开的一方，发送完最后一次ACK之后进入的状态。并且持续时间还挺长的。

能不能发送完ACK之后不进入TIME\_WAIT就直接进入CLOSE状态呢？不行的，这个是为了TCP协议的可靠性，由于网络原因，ACK可能会发送失败，那么这个时候，被动一方会主动重新发送一次FIN，这个时候如果主动方在TIME\_WAIT状态，则还会再发送一次ACK，从而保证可靠性。那么从这个解释来说，2MSL的时长设定是可以理解的，MSL是报文最大生存时间，如果重新发送，一个FIN＋一个ACK，再加上不定期的延迟时间，大致是在2MSL的范围。

所以从理论上说，网上调试参数降低TIME\_WAIT的持续时间的方法是一种以可靠性换取性能的一种方式。嗯，质量守恒定理还是铁律。

**1. time\_wait状态如何产生？**   
由上面的变迁图，首先调用close()发起主动关闭的一方，在发送最后一个ACK之后会进入time\_wait的状态，也就说该发送方会保持2MSL时间之后才会回到初始状态。MSL值得是数据包在网络中的最大生存时间。产生这种结果使得这个TCP连接在2MSL连接等待期间，定义这个连接的四元组（客户端IP地址和端口，服务端IP地址和端口号）不能被使用。

**2.time\_wait状态产生的原因**

**1）为实现TCP全双工连接的可靠释放**

由TCP状态变迁图可知，假设发起主动关闭的一方（client）最后发送的ACK在网络中丢失，由于TCP协议的重传机制，执行被动关闭的一方（server）将会重发其FIN，在该FIN到达client之前，client必须维护这条连接状态，也就说这条TCP连接所对应的资源（client方的local\_ip,local\_port）不能被立即释放或重新分配，直到另一方重发的FIN达到之后，client重发ACK后，经过2MSL时间周期没有再收到另一方的FIN之后，该TCP连接才能恢复初始的CLOSED状态。如果主动关闭一方不维护这样一个TIME\_WAIT状态，那么当被动关闭一方重发的FIN到达时，主动关闭一方的TCP传输层会用RST包响应对方，这会被对方认为是有错误发生，然而这事实上只是正常的关闭连接过程，并非异常。

**2）为使旧的数据包在网络因过期而消失**

为说明这个问题，我们先假设TCP协议中不存在TIME\_WAIT状态的限制，再假设当前有一条TCP连接：(local\_ip, local\_port, remote\_ip,remote\_port)，因某些原因，我们先关闭，接着很快以相同的四元组建立一条新连接。本文前面介绍过，TCP连接由四元组唯一标识，因此，在我们假设的情况中，TCP协议栈是无法区分前后两条TCP连接的不同的，在它看来，这根本就是同一条连接，中间先释放再建立的过程对其来说是“感知”不到的。这样就可能发生这样的情况：前一条TCP连接由local peer发送的数据到达remote peer后，会被该remot peer的TCP传输层当做当前TCP连接的正常数据接收并向上传递至应用层（而事实上，在我们假设的场景下，这些旧数据到达remote peer前，旧连接已断开且一条由相同四元组构成的新TCP连接已建立，因此，这些旧数据是不应该被向上传递至应用层的），从而引起数据错乱进而导致各种无法预知的诡异现象。作为一种可靠的传输协议，TCP必须在协议层面考虑并避免这种情况的发生，这正是TIME\_WAIT状态存在的第2个原因。

**3）总结**   
具体而言，local peer主动调用close后，此时的TCP连接进入TIME\_WAIT状态，处于该状态下的TCP连接不能立即以同样的四元组建立新连接，即发起active close的那方占用的local port在TIME\_WAIT期间不能再被重新分配。由于TIME\_WAIT状态持续时间为2MSL，这样保证了旧TCP连接双工链路中的旧数据包均因过期（超过MSL）而消失，此后，就可以用相同的四元组建立一条新连接而不会发生前后两次连接数据错乱的情况。

1.防止上一次连接中的包，迷路后重新出现，影响新连接（经过2MSL，上一次连接中所有的重复包都会消失）  
2. 可靠的关闭TCP连接。在主动关闭方发送的最后一个 ack(fin) ，有可能丢失，这时被动方会重新发fin, 如果这时主动方处于 CLOSED 状态 ，就会响应 rst 而不是 ack。所以主动方要处于 TIME\_WAIT 状态，而不能是 CLOSED 。另外这么设计TIME\_WAIT 会定时的回收资源，并不会占用很大资源的，除非短时间内接受大量请求或者受到攻击。

**3.time\_wait状态如何避免**

首先服务器可以设置SO\_REUSEADDR套接字选项来通知内核，如果端口忙，但**TCP连接位于TIME\_WAIT状态时可以重用端口**。在一个非常有用的场景就是，如果你的服务器程序停止后想立即重启，而新的套接字依旧希望使用同一端口，此时SO\_REUSEADDR选项就可以避免TIME\_WAIT状态。

所以如果将大量CLOSE\_WAIT的解决办法总结为一句话那就是：查代码。因为问题出在服务器程序里头啊。