# time\_wait状态产生条件

只有在正常四次挥手关闭连接的情况下，在主动关闭连接的一方会出现一段时间的time\_wait。如果启用了快速回收功能，回收时间和网络延迟状况有关，正常情况下小于1s，如果没有开启time\_wait快速回收功能，则time\_wait回收时间默认60s。

三次挥手过程（FIN+ACK， FIN+ACK，ACK）的情况，例如杀掉一段进程，第一个发送FIN+ACK的一端也会产生time\_wait。

# Time\_wait状态相关参数说明

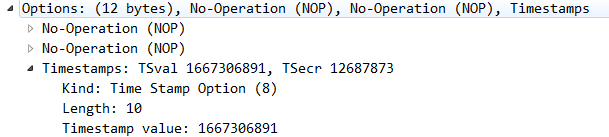
TCP中有和time\_wait状态相关的参数有以下四个：

|  |  |
| --- | --- |
| tcp\_tw\_recycle | 表示开启TCP连接中time\_wait的快速回收功能，默认为0，表示关闭；生效前提是必须启用本端和对端tcp\_timestamps配置。 |
| tcp\_timestamps | 时间戳选项，只有在该选项置1的时候tcp\_tw\_recycle才会生效。 |
| tcp\_max\_tw\_buckets | 表示系统同时保持time\_wait的最大数量，如果超过这个量，time\_wait将打印警告信息。超限的时候后面产生的time\_wait直接不处理，释放资源。注意：是新的连接直接释放资源，老的连接还是处于time\_wait状态。 |
| Tcp\_tw\_reuse | 客户端大量time\_wait状态存在时，端口被占用，当有新的连接，如果没有可用端口，则会连接失败。启用该功能后，可以复用time\_wait状态的连接。客户端tcp\_tw\_reuse生效前提是启用本端和对端tcp\_timestamp。  Tcp\_tw\_reuse端口重用功能一般只针对客户端，因为服务端一般都是监听固定端口，端口数是固定的，端口不会用完。而客户端每次连接端口一般都是由协议栈自动分配。 |

# Time\_wait快速回收

## 3.1 快速回收功能失效前提

Time\_wait快速回收功能生效前提：启用tcp\_tw\_recycle，并启动本端和对端tcp\_timestamps配置。启用timestamps功能时，报文中会携带时间戳选项信息，抓包如下：



## 3.2 启用time\_wait快速回收功能副作用

1. 如果启用了tcp\_tw\_recycle和tcp\_timestamps，如果接收报文四层选项字段带有时间戳信息，则会对时间戳进行检查，对不满足条件的包会直接丢弃，可能会造成客户端连接建立不成功。例如网络路由信息反复变化，移动cmwap网络发来的包的时间戳乱跳，同一局域网通过路由器做NAT访问服务器(因为做NAT后，源IP就变为路由器的IP了，如果局域网内各个电脑系统时间不一致，则会出现)等情况有可能会出现部分连接异常。原因是tcp\_tw\_recycle/tcp\_timestamps以及对端tcp\_timestamps都开启的条件下，60s内同一源ip主机的socket connect请求中的timestamp必须是递增的。不同主机经过路由器做NAT后，报文的源IP地址就变为路由器的IP地址了。

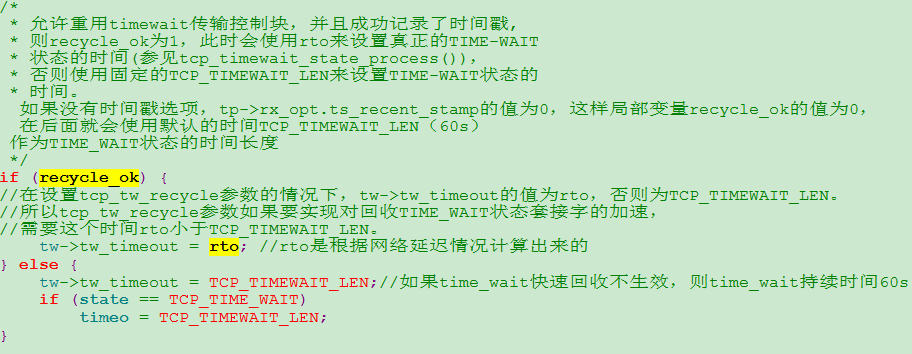
## 3.3 timestack数据包-Wireshark

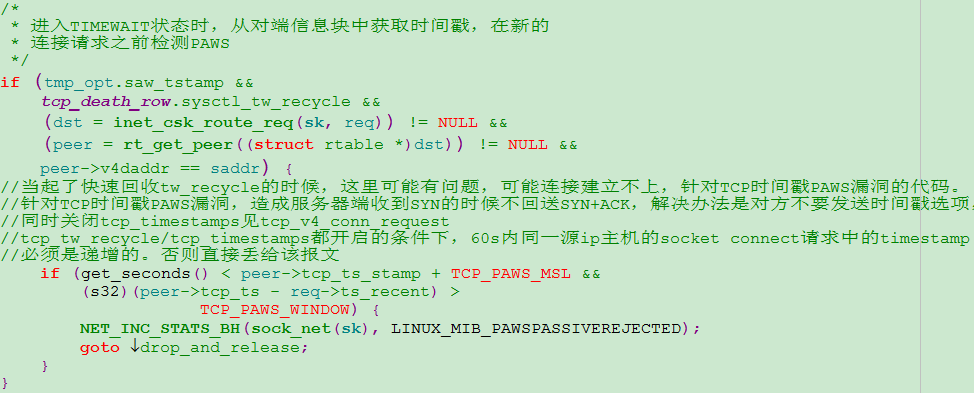


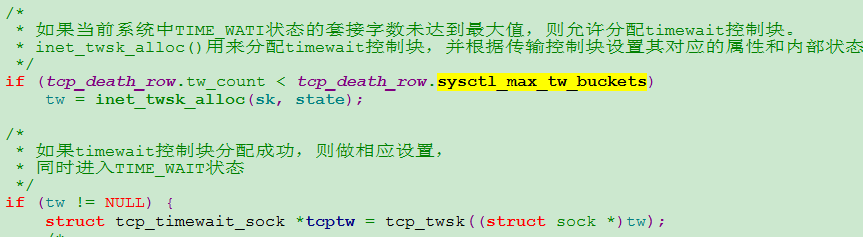
## 3.4 内核协议栈相关主要源码

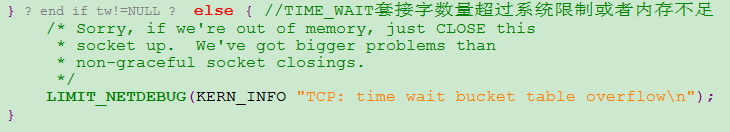
Time\_wait状态生成及快速回收相关代码：



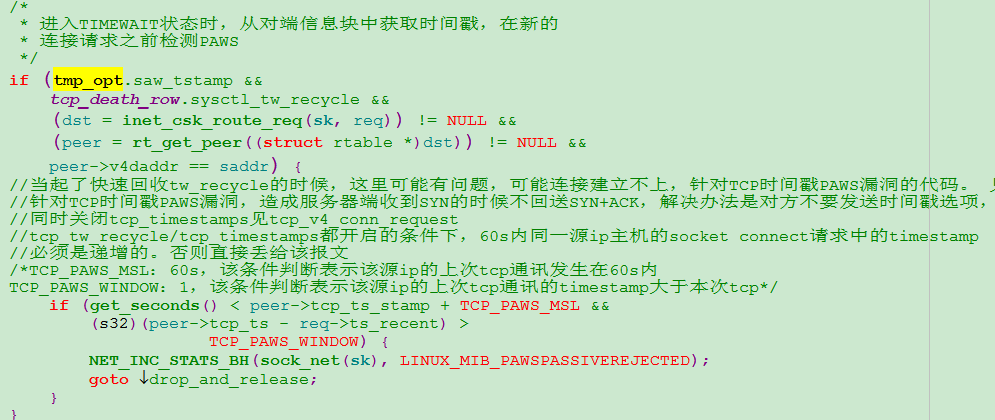








开启timestamps引起的丢包相关源码如下:

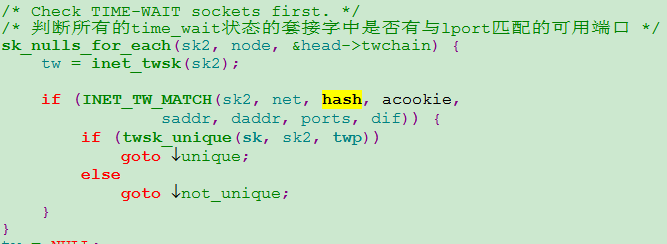


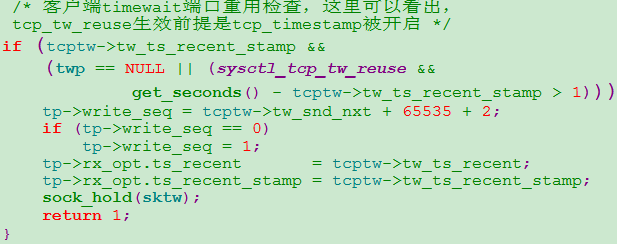
# 客户端端口重用

## 4.1 客户端大量time\_wait，端口重用前提

启用tcp\_tw\_reuse，并启动本端和对端tcp\_timestamps配置。

## 4.2 内核协议栈相关主要源码







# 大量timewait对客户端、服务端影响

## 5.1 客户端大量time\_wait影响

1. 大量time\_wait会造成连接资源不释放，内存无法回收。
2. 由于客户端端口一般采用协议栈随机分配的方式，协议栈会给每个客户端连接分配一个未使用的端口，因此如果客户端同一IP对应的time\_wait数量超过ip\_local\_port\_range设置的最大值（也就是65000），端口将被用完，连接会无法建立。

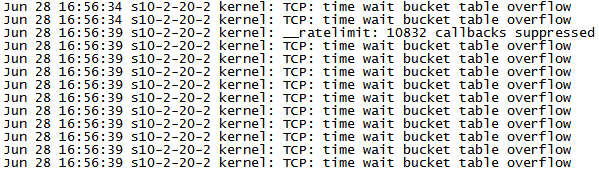
## 5.2 服务端大量time\_wait影响

由于服务端只占用监听端口，因此不存在端口用完的现象。服务端大量time\_wait唯一影响是：资源不释放，内存无法回收。

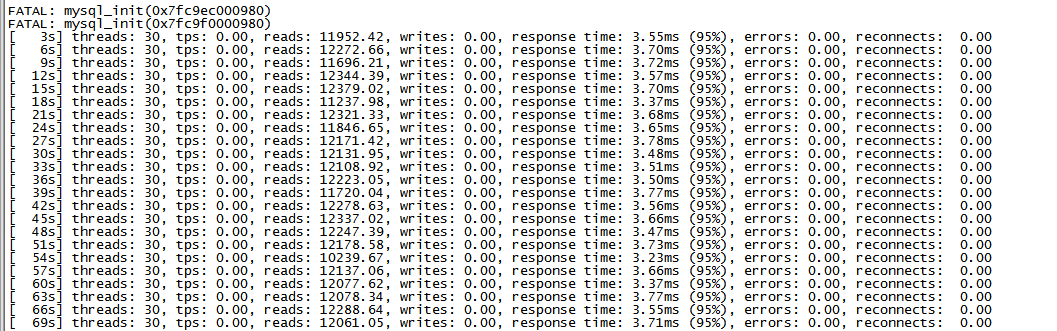
# cobar测试验证

本次测试结果采用sysbench.short来压测cobar来验证，客户端物理设备和服务端物理设备的ip\_local\_port\_range(1024~65000)和tcp\_max\_tw\_buckets(81920)参数都是默认值，测试结果如下：

1. 当cobar服务端time\_wait数达到81920的时候，任然可以继续接收客户端连接，能够正常提供连接服务。
2. 当客户端测试工具sysbench.short服务器上的time\_wait数达到60000多的时候，客户端连接失败，无法连接，因为端口用完。打印：Cannot assign requested address； Cobar服务器time\_wait超限时打印：



Cobar服务器time\_wait超限的情况下，客户端sysbench压测结果基本不受影响，如下：



从上面测试可以看出，服务端time\_wait不会影响客户端建链，只是占用内存。如果是客户端出现大量time\_wait状态，此时端口用完，则无法建立连接。以上测试结论符合理论、代码分析

# 三种解决time\_wait方法总结

|  |  |  |  |
| --- | --- | --- | --- |
|  | Time\_wait快速回收 | 端口重用 | 限制Tcp\_max\_tw\_buckets |
| 配置方法 | 在需要进行time\_wait快速回收的一端进行一下配置：  tcp\_tw\_recycle：1  本端tcp\_timestamps：1  对端tcp\_timestamps：1 | 在需要进行time\_wait快速回收的一端进行一下配置：  tcp\_tw\_reuse：1  本端tcp\_timestamps：1  对端tcp\_timestamps：1 | 配置Tcp\_max\_tw\_buckets  的值在60000以下。例如配置为30000 |
| 副作用 | 在某些情况下可能引起用户建连接失败（例如需要直接返回给用户信息的服务器）  比较暴力，不符合TCP协议规范 | 在某些情况下可能引起用户建连接失败（例如需要直接返回给用户信息的服务器）  部署复杂，需要同时改服务端，而服务端比较多。  服务器时间戳会带出IDC，经过中间各种网络设备，尤其是运营商的无线基站设备等，如果某个设备对时戳有校验，则会产生丢包问题。 | 比较暴力，不符合TCP协议规范  应急的处理，立竿见影。  建议这种。 |

# 使用建议

1. 开启tcp\_tw\_reuse功能，这是端口重用的总开关，打开。
2. tcp\_timestamps需要根据实际情况进行设置，如果客户端和服务端都不会暴露给用户，则可以设置为1，如果是需要暴露给用户，只是内部使用，则建议开启端口重用功能。
3. 服务端Tcp\_max\_tw\_buckets参数根据内存情况进行调整，如果内存充足，采用默认值，内存不足可以适当调小该值。
4. tcp\_tw\_recycle，快速回收功能，不建议开启，比较暴力。可以选择开启端口重用功能。
5. 官方说明：

<https://www.kernel.org/doc/Documentation/networking/ip-sysctl.txt>

