四次挥手（重要）

问题1：为什么需要CLOSED_WAIT?

问题2：为什么需要TIME_WAIT?

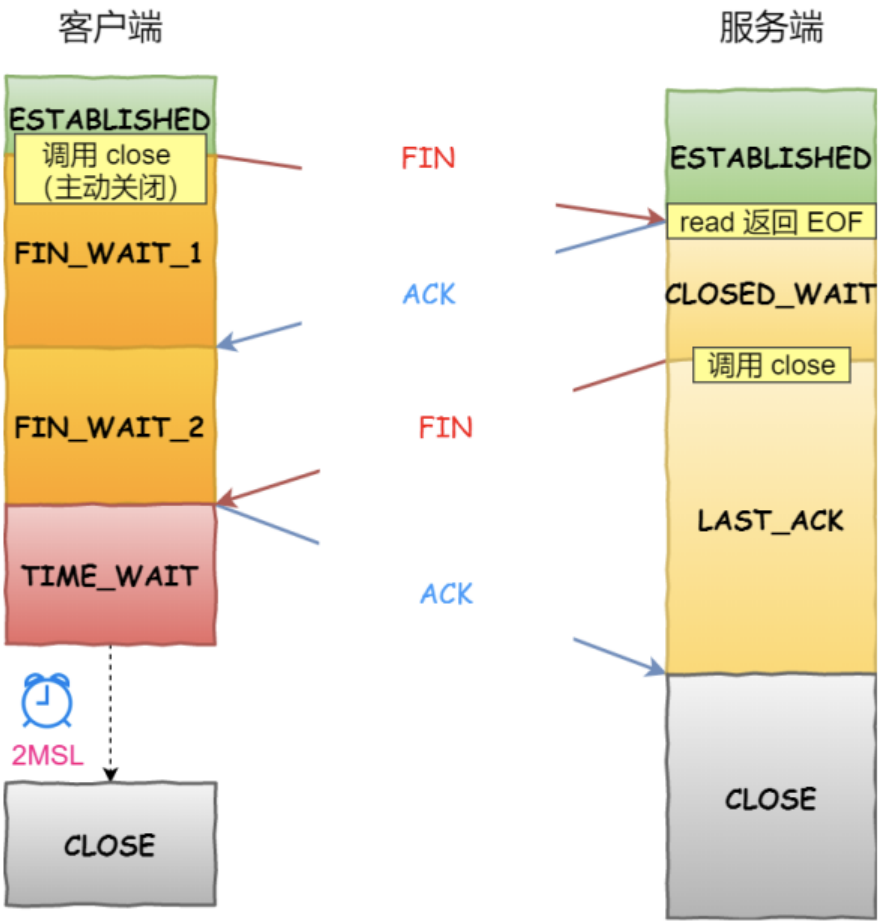
问题3：如果TIME_WAIT过短或者过长，有什么影响?

问题4：如果TIME_WAIT过多，会有什么危害?

问题5：如何优化TIME_WAIT?

问题6：如果已经建立了连接，但是客户端突然故障怎么办？？

> 注意一个小概念：客户端和服务端都可以主动断开连接，通常先关闭连接的一方叫主动方，后关闭连接的一方被称为被动方



1. 主动方主动关闭连接，发送一个FIN报文，然后客户端进入 FIN_WAIT_1状态；
2. 接收方收到FIN报文后，回复ACK报文，接收方进入CLOSED_WAIT 状态;

3. 主动方收到ACK信息后，进入 FIN_WAIT_2 状态；
4. 等待接收方处理完数据后，向主动方发送FIN报文，之后接收方进入LAST_ACK状态；
5. 主动方收到FIN报文后，回复ACK，然后进入TIME_WAIT状态；
6. 接收方接收到ACK报文后，进入CLOSED状态，至此接收方完成连接关闭；
7. 主动方在经过2MSL后，自动进入CLOSED状态，至此主动方完成连接关闭.

问题1：为什么需要CLOSED_WAIT?

因为此时接收方还有数据需要处理（接收），处理完后，再关闭

问题2：为什么需要TIME_WAIT?

首先要知道主动发起关闭的一方才有TIME_WAIT状态，需要这个状态主要是两个原因

1. **等待足够的时间确保被动接收关闭的一方可以正常接到主动一方发送的ACK，确保可以正常关闭连接(如果时间在2MSL后，被动方的FIN没收到，大于2MSL后收到了，这时候主动方已CLOSE，服务端发送超过设定的次数就关闭TCP链接)**
2. **处于TW状态的，如果主动方发送的ACK丢失，那么在2MSL中被动方可以重传FIN，再次重置2MSL等待时间后 ，TCP连接才断开，如果不存在TW或者小于2MSL，主动方立马断开连接，则应用程序可以立马与主动方建立连接一个新连接（新连接和原连接相似），重传的FIN可能会干扰这个新连接，导致TCP连接不可靠。**

参考文章：https://blog.csdn.net/qq_41804181/article/details/81941783

问题3：如果TIME_WAIT过短或者过长，有什么影响?

1. **如果TIME_WAIT过短或者没有**：客户端直接进入closed状态，则服务端会一直处于last_ack状态，当客户端发起建立连接的请求时，服务端就会发送RST报文给客户端，来中止连接
2. **如果TIME_WAIT过长**，就会遇到下面两种情况：
    - 服务端正常收到四次挥手的最后一个ACK报文，则服务端正常关闭连接
    - 没有收到ACK报文，则重发FIN，继续等待ACK

问题4：如果TIME_WAIT过多，会有什么危害?

如果是服务端（一般是接收方）有处于TIME_WAIT状态的TCP，则说明是由服务器方主动发起的断开请求。

TIME_WAIT过多主要危害有两种：

1. 内存资源占用
2. 端口资源占用：一个TCP连接至少消耗一个本地端口

第二个危害比较大，如果主动方TIME_WAIT状态过多，占满了所有端口资源，则会导致无法创建新连接，另外：

服务端受系统资源限制：

- 由于一个四元组表示 TCP 连接，理论上服务端可以建立很多连接，服务端确实只监听一个端口 但是会把连接扔给处理线程，所以理论上监听的端口可以继续监听。但是线程池处理不了那么多一直不断的连接了。所以当服务端出现大量 TIME_WAIT 时，系统资源被占满时，会导致处理不过来新的连接。

问题5：如何优化TIME_WAIT?

下面几种方法都有利有弊

1. 打开 net.ipv4.tcp_tw_reuse 和 net.ipv4.tcp_timestamps 选项；
    - 打开后可以复用处于TIME_WAIT的socket为新连接使用
    - 注意：tcp_tw_reuse 功能只能用客户端（连接发起方），因为开启了该功能，在调用 connect()函数时，内核会随机找一个 time_wait 状态超过 1 秒的连接给新的连接复用。
    - 注意，打开第一个tcp_tw_reuse，必须打开第二个时间戳设置
    - 引入时间戳后，就不存在2MSL问题了，重复数据包在时间戳过期后值自动丢弃
2. net.ipv4.tcp_max_tw_buckets
    - 默认18000，一旦TIME_WAIT超过，系统后面就会重置TIME_WAIT连接状态
    - 过于暴力，问题会很多
3. 程序中使用SO_LINGER，应用强制使用RST关闭。

```
struct linger so_linger;
so_linger.l_onoff = 1;
so_linger.l_linger = 0;
setsockopt(s, SOL_SOCKET, SO_LINGER, &so_linger,sizeof(so_linger));
```

如果 l_onoff 为非 0， 且 l_linger 值为 0，那么调用 close 后，会立该发送一个 RST 标志给对端，该 TCP 连接将跳过四次挥手，也就跳过了 TIME_WAIT 状态，直接关闭。但这为跨越 TIME_WAIT 状态提供了一个可能，不过是一个非常危险的行为，不值得提倡。

问题6：如果已经建立了连接，但是客户端突然故障怎么办？？

TCP是有一个**保活机制**的，原理如下

定义一个时间段，在这个时间段内，如果没有任何连接相关的活动， TCP 保活机制会开始作用，每隔一个时间间隔，发送一个探测报文，该探测报文包含的数据非常少，如果连续几个探测报文都没有得到

响应，则认为当前的TCP 连接已经死亡，系统内核将错误信息通知给上层应用程序。在 Linux 内核可以有对应的参数可以设置保活时间、保活探测的次数、保活探测的时间间隔，以下都为默认值：

```
net.ipv4.tcp_keepalive_time=7200
net.ipv4.tcp_keepalive_intvl=75
net.ipv4.tcp_keepalive_probes=9
```

- tcp_keepalive_time=7200：表示保活时间是 7200 秒（2小时），也就2小时内如果没有任何连接相关的活动，则会启动保活机制
- tcp_keepalive_intvl=75：表示每次检测间隔 75 秒；
- tcp_keepalive_probes=9：表示检测 9 次无响应，认为对方是不可达的，从而中断本次的连接。也就是说在 Linux 系统中，最少需要经过 2 小时 11 分 15 秒才可以发现一个「死亡」连接。

如果开启了TCP保活，需要考虑以下几种情况：

1. 对端程序是正常工作的。当 TCP 保活的探测报文发送给对端, 对端会正常响应，这样 TCP 保活时间会被重置，等待下一个 TCP 保活时间的到来
2. 对端程序崩溃并重启。当 TCP 保活的探测报文发送给对端后，对端是可以响应的，但由于没有该连接的有效信息， 会产生一个 RST 报文，这样很快就会发现 TCP 连接已经被重置。
3. 第三种，是对端程序崩溃，或对端由于其他原因导致报文不可达。当 TCP 保活的探测报文发送给对端后，石沉大海，没有响应，连续几次，达到保活探测次数后， TCP 会报告该 TCP 连接已经死亡。

参考：小林《图解网络》，王道《计算机网络》，《图解TCP/IP》