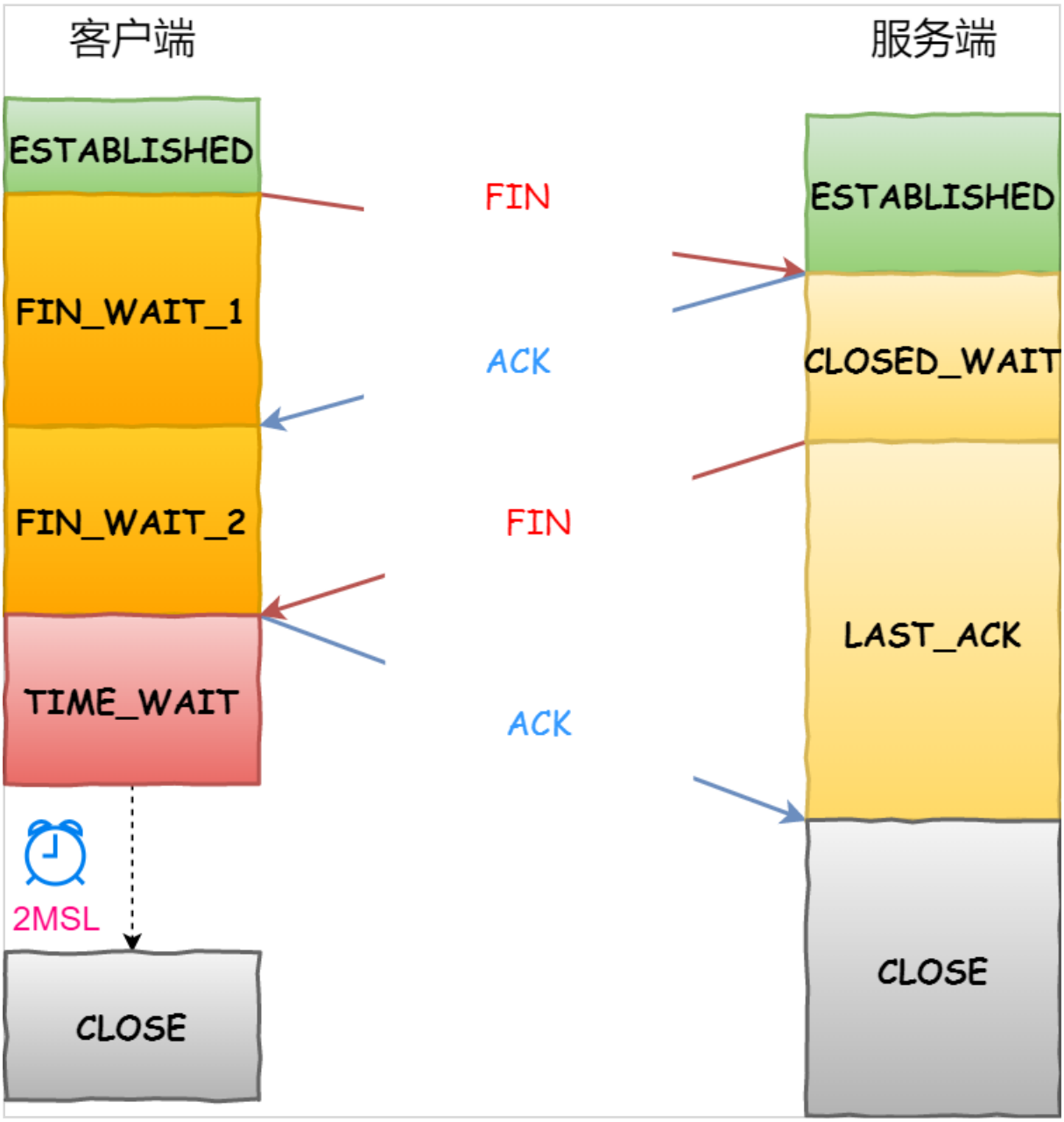
五、TCP四次挥手

TCP四次挥手过程和状态变迁

TCP 断开连接是通过**四次挥手**方式

双方都可以主动断开连接，断开连接后主机中的「资源」将被释放



- 客户端打算关闭连接，此时会发送一个 TCP 首部 `FIN` 标志位被置为 1 的报文，也即 FIN 报文，之后客户端进入 `FIN_WAIT_1` 状态
- 服务端收到该报文后，就向客户端发送 ACK 应答报文，接着服务端进入 `CLOSED_WAIT` 状态
- 客户端收到服务端的 ACK 应答报文后，之后进入 `FIN_WAIT_2` 状态
- 等待服务端处理完数据后（可能有数据没发完），也向客户端发送 `FIN` 报文，之后服务端进入 `LAST_ACK` 状态
- 客户端收到服务端的 `FIN` 报文后，回一个 `ACK` 应答报文，之后进入 `TIME_WAIT` 状态

- 服务器收到了 `ACK` 应答报文后，就进入了 `CLOSED` 状态，至此服务端已经完成连接的关闭
- 客户端在经过 `2MSL` 一段时间后，自动进入 `CLOSED` 状态，至此客户端也完成连接的关闭

> 你可以看到，每个方向都需要**一个FIN和一个ACK**，因此通常被称为**四次挥手**
>
> 这里一点需要注意是:**主动关闭连接的，才有TIME_WAIT状态**

为什么需要挥手四次?

- 关闭连接时，客户端向服务端发送 FIN 时，仅仅表示客户端不再发送数据了但是还能接收数据。
- 服务器收到客户端的 FIN 报文时，先回一个 ACK 应答报文，而服务端可能还有数据需要处理和发送，等服务端不再发送数据时，才发送 FIN 报文给客户端来表示同意现在关闭连接

从上面过程可知，服务端通常需要等待完成数据的发送和处理，所以服务端的 `ACK` 和 `FIN` 一般都会分开发送，从而比三次握手导致多了一次

为什么TIME_WAIT的等待时间是2MSL?

`MSL` 是 Maximum Segment Lifetime，**报文最大生存时间**，它是任何报文在网络上存在的最长时间， 超过这个时间报文将被丢弃。因为 TCP 报文基于是 IP 协议的，而 IP 头中有一个 `TTL` 字段，是 IP 数据报可以经过的最大路由数，每经过一个处理他的路由器此值就减 1，当此值为 0 则数据报将被丢弃， 同时发送 ICMP 报文通知源主机

MSL 与 TTL 的区别: MSL 的单位是时间，而 TTL 是经过路由跳数。所以 **MSL应该要大于等于TTL消耗为0的时间**，以确保报文已被自然消亡

TIME_WAIT 等待 2 倍的 MSL，比较合理的解释是: 网络中可能存在来自发送方的数据包，当这些发送方的数据包被接收方处理后又会向对方发送响应，所以**一来一回需要等待2倍的时间**

比如如果被动关闭方没有收到断开连接的最后的 ACK 报文，就会触发超时重发 Fin 报文，主动方接收到 FIN 后，会重发 ACK 给被动关闭方， 一来一去正好 2 个 MSL

`2MSL` 的时间是从**客户端接收到FIN后发送ACK开始计时的**。如果在 TIME-WAIT 时间内，因为客户端的 ACK 没有传输到服务端，客户端又接收到了服务端重发的 FIN 报文，那么 **2MSL时间将重新计时**。

在 Linux 系统里 `2MSL` 默认是 `60` 秒，那么一个 `MSL` 也就是 `30` 秒。**Linux系统停留在TIME_WAIT的时间为固定的60秒**

其定义在 Linux 内核代码里的名称为 TCP_TIMEWAIT_LEN：

```
#define TCP_TIMEWAIT_LEN (60*HZ) /* how long to wait to destroy TIME-WAIT
                                    state, about 60 seconds  */
```

如果要修改 TIME_WAIT 的时间长度，只能修改 Linux 内核代码里 TCP_TIMEWAIT_LEN 的值，并重新编译 Linux 内核。

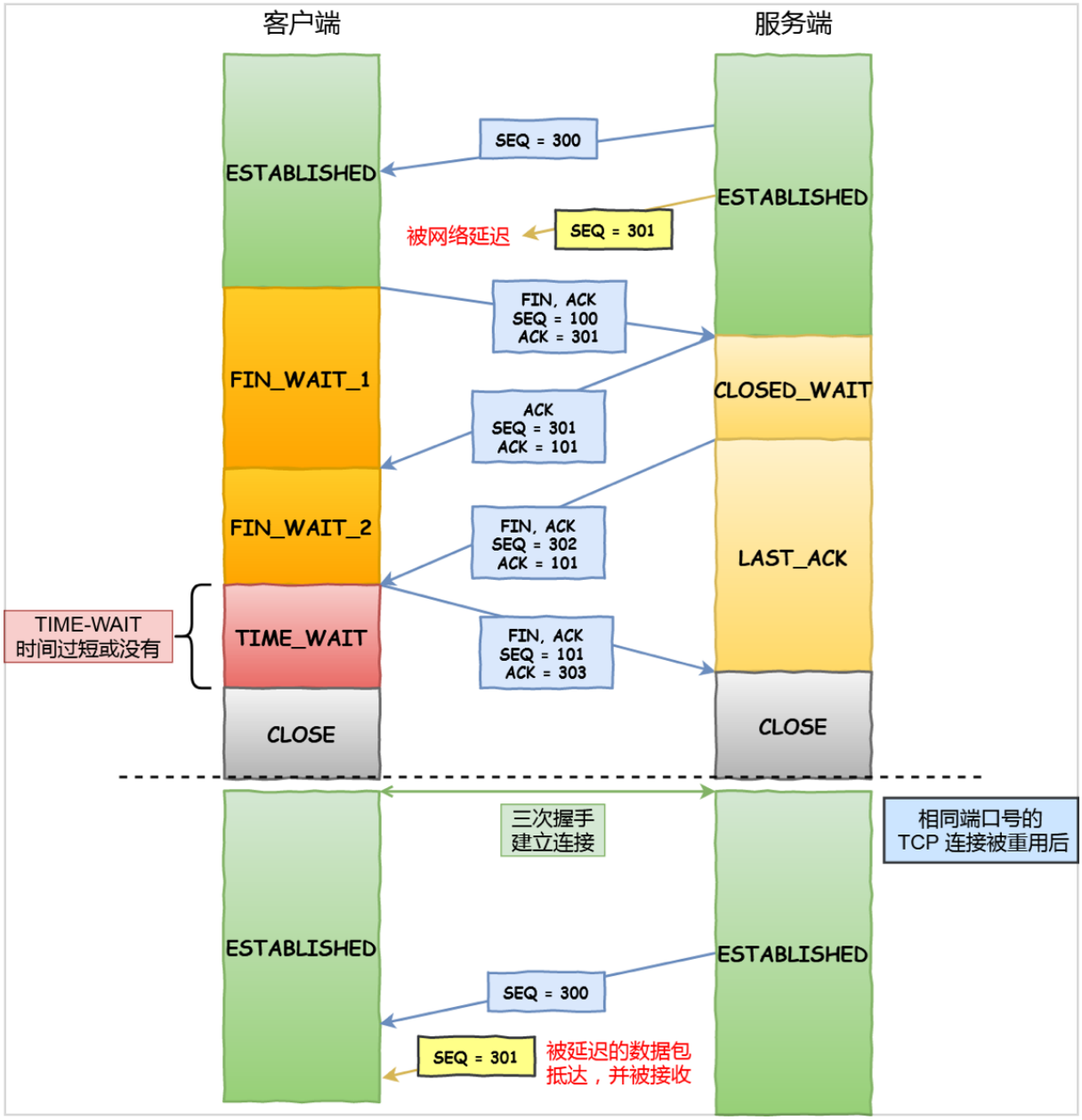
为什么需要TIME_WAIT状态?

主动发起关闭连接的一方，才会有 `TIME-WAIT` 状态
需要 TIME-WAIT 状态，主要是两个原因:

1. 防止具有相同「四元组」的「旧」数据包被收到;
2. 保证「被动关闭连接」的一方能被正确的关闭，即保证最后的 ACK 能让被动关闭方接收，从而帮 助其正常关闭;

- 原因一：防止旧连接的数据包被新连接接收

假设 TIME-WAIT 没有等待时间或时间过短，被延迟的数据包抵达后会发生什么呢



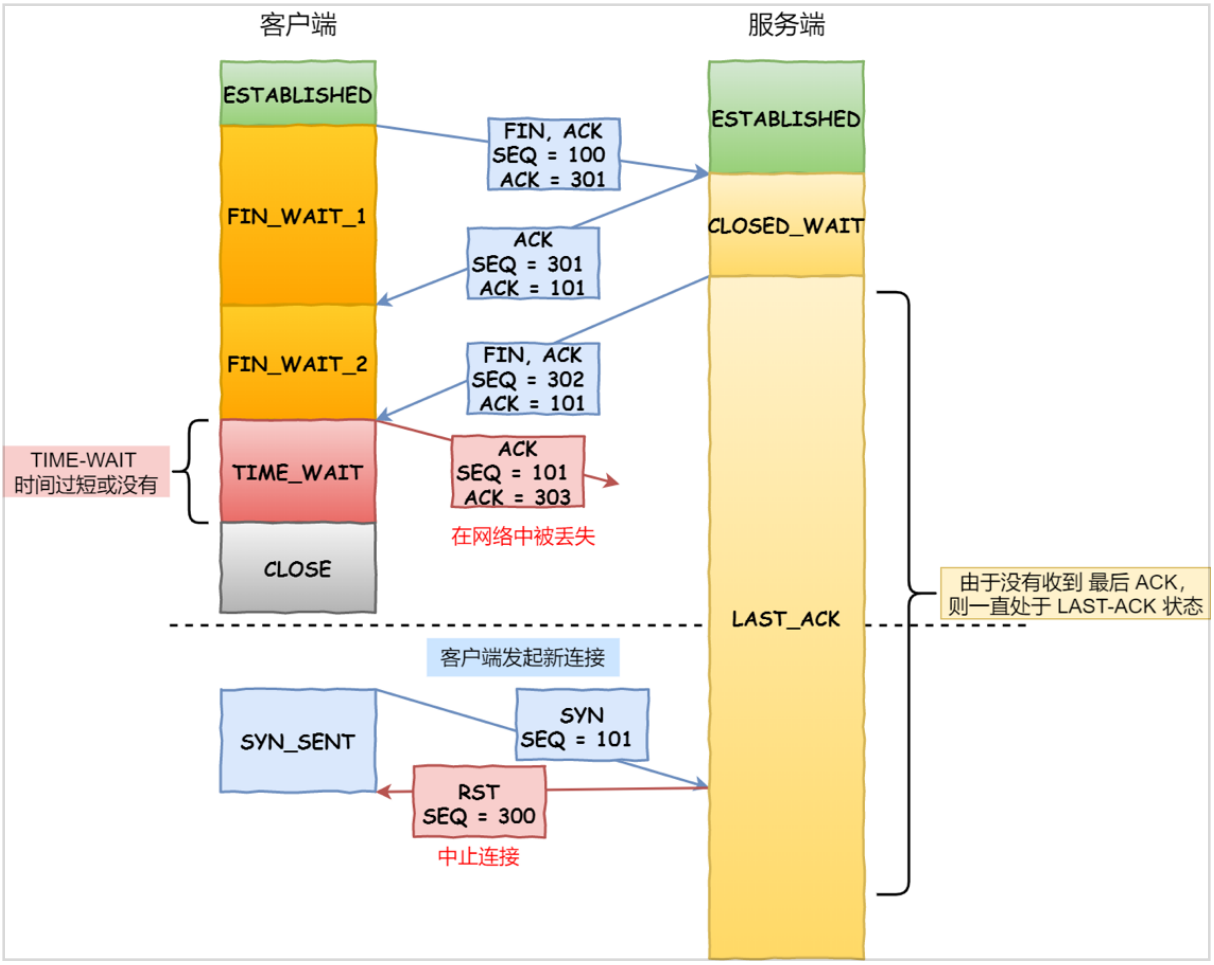
- 如上图黄色框框服务端在关闭连接之前发送的 `SEQ=301` 报文，被网络延迟了
- 这时有相同端口的 TCP 连接被复用后，被延迟的 抵达了客户端，那么客户端是有可 能正常接收这个过期的报文，这就会产生数据错乱等严重的问题

所以，TCP 就设计出了这么一个机制，经过 `2MSL` 这个时间，**足以让两个方向上的数据包都被丢弃，使得原来连接的数据包在网络中都自然消失，再出现的数据包一定都是新建立连接所产生的。**

- 原因二：保证连接正确关闭

`TIME-WAIT` 作用是**等待足够的时间以确保最后的ACK能让被动关闭方接收，从而帮助其正常关闭**

假设 TIME-WAIT 没有等待时间或时间过短，断开连接会造成什么问题呢?



- 如上图红色框框客户端四次挥手的最后一个 ACK 报文如果在网络中被丢失了，此时如果客户端 `TIME_WAIT` 过短或没有，则就直接进入了 `CLOSED` 状态了，那么服务端则会一直处在 `LASE_ACK` 状态

- 当客户端发起建立连接的 `SYN` 请求报文后，服务端会发送 `RST` 报文给客户端，连接建立的过程就会被终止

如果 TIME-WAIT 等待足够长的情况就会遇到两种情况：

- 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
- 服务端没有收到四次挥手的最后一个 ACK 报文时，则会重发 FIN 关闭连接报文并等待新的 ACK 报文

所以客户端在 TIME-WAIT 状态等待 2MSL 时间后，就可以**保证双方的连接都可以正常的关闭。**

TIME_WAIT过多有什么危害?

如果服务器有处于 TIME-WAIT 状态的 TCP，则说明是由服务器方主动发起的断开请求。 过多

的 TIME-WAIT 状态主要的危害有两种:

- 第一是**内存资源占用**;
- 第二是对**端口资源的占用**，一个 TCP 连接至少消耗一个本地端口;

第二个危害是会造成严重的后果的，要知道，端口资源也是有限的，一般可以开启的端口为 `32768 ~ 61000` ，也可以通过如下参数设置指定

```
net.ipv4.ip_local_port_range
```

如果发起连接一方的 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。

客户端受端口资源限制:

- 客户端TIME_WAIT过多，就会导致端口资源被占用，因为端口就65536个，被占满就会导致无法创建新的连接。

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。

如何优化TIME_WAIT?

这里给出优化 TIME-WAIT 的几个方式，都是有利有弊:

- 打开 `net.ipv4.tcp_tw_reuse` 和 `net.ipv4.tcp_timestamps` 选项;
- `net.ipv4.tcp_max_tw_buckets`
- 程序中使用 `SO_LINGER` ，应用强制使用 `RST` 关闭

> 小林网络page137

如果已经建立了连接，但是客户端突然出现故障了怎么办?

TCP 有一个机制是**保活机制**。这个机制的原理是这样的:

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