png](https://p6-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/558b8e4d4b224091829e2fffc32476fa~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=938&h=90&s=77462&e=png&b=253239)

根据上面的 Local Address 中的端口，找到对应的进程，发现此进程是我们自身的服务。

```
lsof -iTCP:19585
```

由此可见是我们的服务链接端口 1234 进程的 tcp 连接数过多，并且链接状态大多数是 ESTABLISHED。定位到链接消耗的范围已经确定，本地进程调用端口 1234 的进程也符合我们的业务逻辑。下面就需要分析为什么会有这么多连接是 ESTABLISHED，为什么没有被释放。

确认服务的链接状态

TCP 连接的状态

首先我们先来了解下 TCP 的链接状态，在 TCP（传输控制协议）连接的建立和关闭过程中，会经历一系列特定的状态，这些状态定义了连接在不同阶段的状态和行为。了解这些状态有助于诊断和解决网络问题。

TCP 连接建立（Three-Way Handshake）

TCP 连接的建立过程称为“三次握手”，包括以下状态：

1. CLOSED：初始状态，表示连接关闭或不存在。
2. SYN-SENT：客户端在发送连接请求（SYN）报文后进入该状态，等待服务器的回应。
3. SYN-RECEIVED：服务器在收到客户端的 SYN 报文并发送 SYN-ACK 报文后进入该状态，等待客户端的确认。

4. ESTABLISHED：客户端和服务器在完成三次握手后都进入该状态，表示连接已经建立，可以进行数据传输。

三次握手详细步骤

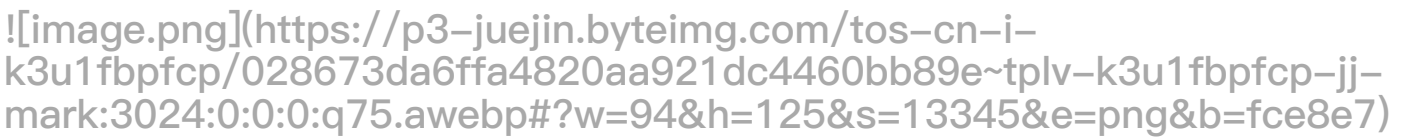
1. 客户端 → 服务器：SYN
2. 1. 客户端发送一个 SYN 报文，表示希望建立连接。
2. 进入 `SYN-SENT` 状态。
3. 服务器 → 客户端：SYN-ACK
4. 1. 服务器收到 SYN 报文后，发送一个 SYN-ACK 报文，表示同意建立连接，并要求确认。
2. 进入 `SYN-RECEIVED` 状态。
5. 客户端 → 服务器：ACK
6. 1. 客户端收到 SYN-ACK 报文后，发送一个 ACK 报文，确认连接建立。
2. 客户端和服务器都进入 `ESTABLISHED` 状态，连接建立完成。

TCP 连接关闭（Four-Way Handshake）

TCP 连接的关闭过程称为“四次挥手”，包括以下状态：

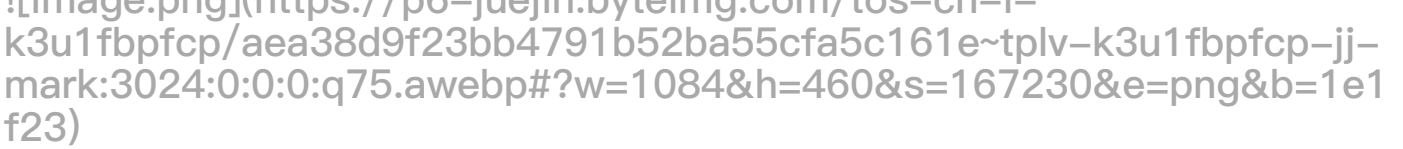
1. FIN-WAIT-1：主动关闭连接的一方（通常是客户端）发送 FIN 报文后进入该状态，等待对方的确认。
2. FIN-WAIT-2：在收到对方的 ACK 报文后进入该状态，等待对方的 FIN 报文。
3. CLOSE-WAIT：被动关闭连接的一方（通常是服务器）收到 FIN 报文后进入该状态，等待应用程序处理完成后关闭连接。
4. LAST-ACK：被动关 P 客户端能够通过代理服务器访问 Internet。
3. TLS 配置：`Transport` 结构体可以配置 TLS（Transport Layer Security）参数，包括根证书、客户端证书、客户端密钥等，用于建立安全的 HTTPS 连接。
4. 超时设置：`Transport` 结构体可以设置连接超时时间、读取超时时间、写入超时时间等，用于控制与服务器建立连接、发送请求和接收响应的超时行为。
5. 重定向策略：`Transport` 结构体可以设置重定向策略，包括是否允许重定向、最大重定向次数等，用于控制 HTTP 客户端的重定向行为。
6. 空闲连接管理：`Transport` 结构体可以设置空闲连接的超时时间，用于控制连接池中空闲连接的保持时间，超过该时间未被使用的连接会被关闭。
7. 基础认证和代理认证：`Transport` 结构体可以设置基础认证和代理认证的用户名和密码，用于在发送请求时进行身份验证。
8. HTTP/2 支持：`Transport` 结构体可以配置是否启用 HTTP/2 协议，以及 HTTP/2 相关的参数。

如果看过了解过或者阅读过 http 库代码的同学，一眼就能发现这个代码的问题。**根本原因主要就是在每次调用时都新建了 transport，这导致了链接不能被复用**。从而引发了每次调用都创建新的 tcp 连接，并且链接还是默认打开 keepalive 的。这样就会引发这个问题，链接累计的越来越多，且不能再使用完立刻关闭。



问题修复

修复这个问题比较简单，只需要将transport架构体定义为全局变量，为了更加优雅和安全又增加了最大空闲连接数和链接的空闲时间以及keealive的保持周期。参考如下：



上线运行后，观察了一段时间连接数不再持续增加，非常稳定，问题得到解决。

总结

通过本文我们可以了解到，在使用底层库时，必须小心谨慎，了解其实现机制，以避免因为不熟悉库的行为而引发问题。在处理网络连接问题时，了解 TCP 连接的状态、建立和关闭过程，以及熟悉相关工具的使用方法，能够更好帮助我们理解和解决问题。ps: 后续有精力会出一篇文章，详细介绍golang http client库的原理和设计。

原文链接: https://juejin.cn/post/7372404987702378536