TCP/IP

参考资料：内网与外网

IP 地址格式：ipv4——32 位共 4 字节，每一字节使用十进制表示

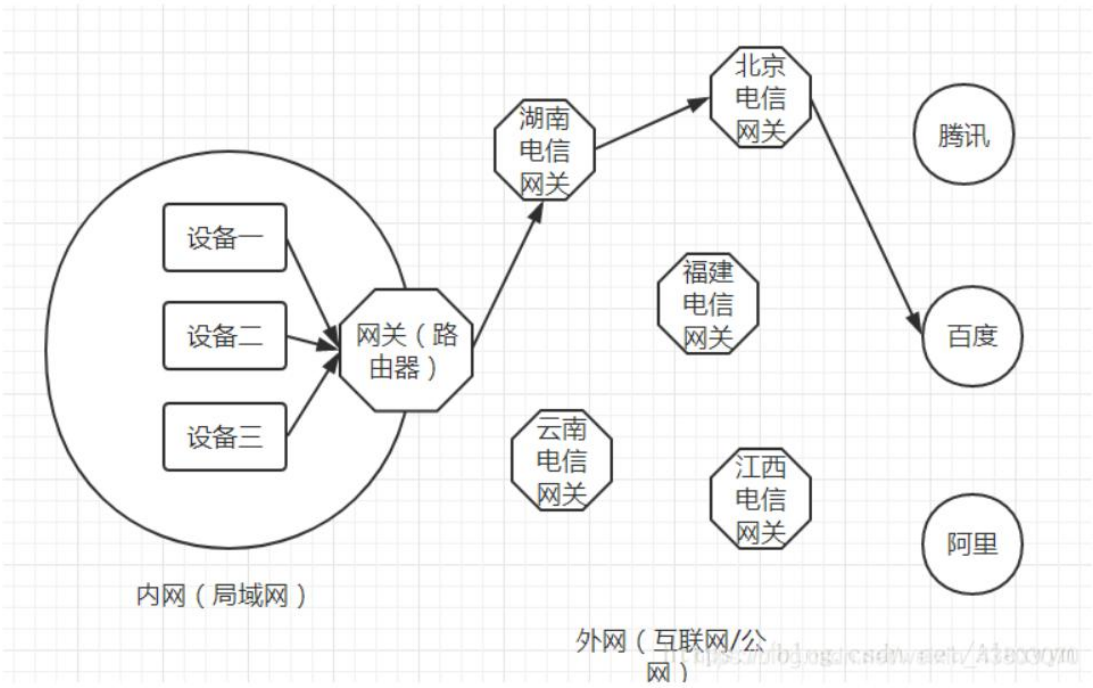
内网与外网的概念是相对的，内网 IP 经过 NAT 地址转换为外网 IP——解决了计算机爆发式增长的问题。内网可能是一个独立的局域网，通过其中的**网关**（网关就是连接两个网络的节点，说白了，就是有双重身份的电脑，既有局域网的 IP 地址，又有 Internet 的 IP 地址，两个 IP 地址分别捆绑在不同的网卡上）**的代理访问外部网络**

（注：所谓代理，就是你提要求，他来办事，类似于代购火车票。局域网的电脑想和外面联络，就把对方地址告诉服务器，也就是网关，网关以自己的身份和对方联络，同时把对方发回来的消息转送给局域网内的电脑。因此，**对方看不见局域网内电脑的 IP，只会以为是网关那台电脑在与自己交流。**网吧内的所有 QQ 都显示同样的 IP，现在你能理解为什么了吗？）

不同的计算机在不同的局域网（内网）下内网 IP 可以是相同的，但只要经过网关地址转换为外网 IP 后，每台计算机都有唯一的 IP 地址。

网络上的通信通过网关中转，发送方将数据发送至网关而不是具体的 PC，数据再由网关进行分配，因此**发送方角度的接收方无法分辨接收方是 PC 还是代理（网关）。**

内网ip和外网ip的联系以及连接过程



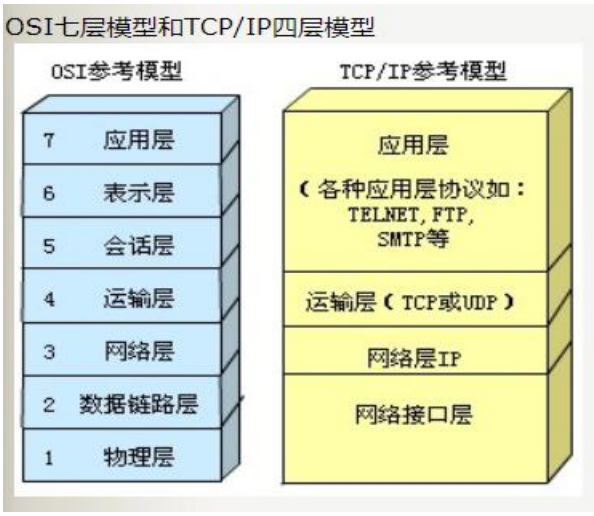
整个网络呈网状结构，它会自动找到一条通往目的（图中的百度）的路径——**基于深度优先搜索或者广度优先搜索**

网络模型

OSI 七层模型：OSI（Open System Interconnection）开放系统互连参考模型是国际标准化组织（ISO）制定的一个用于计算机或通信系统间互联的标准体系。

TCP/IP 四层模型：TCP/IP 参考模型是计算机网络的祖父 ARPANET 和其后继的因特网使用的参考模型。

分层作用：方便管理



七层模型优点：

1、把复杂的网络划分成为更容易管理的层（将整个庞大而复杂的问题划分为若干个容易处理的小问题）

2、没有一个厂家能完整的提供整套解决方案和所有的设备，协议.

3、独立完成各自该做的任务，互不影响，分工明确，上层不关心下层具体细节，分层同样有益于网络排错

功能与代表设备

分层	名字	功能	工作在该层的设备
7	应用层	提供用户界面	QQ、IE 浏览器应用程序
6	表示层	表示数据，进行加密等处理	
5	会话层	将不同应用程序的数据分离	
4	传输层	提供可靠或不可靠的传输，在重传前执行纠错	防火墙
3	网络层	提供逻辑地址，路由器使用它们来选择路径	三层交换机、路由器
2	数据链路层	将分组拆分为字节，并讲字节组合成帧，使用 MAC 地址提供介质访问，执行错误检测，但不纠错	二层交换机，网卡
1	物理层	在设备之间传输比特，指定电平，电缆速度和电缆针脚	集线器

问题：为什么现代网络通信过程中用 TCP/IP 四层模型，而不是用 OSI 七层模型呢?

OSI 七层模型是理论模型，一般用于理论研究，他的分层有些冗余，实际应用，选择 TCP/IP 的四层模型。而且 OSI 自身也有缺陷，大多数人都认为 OSI 模型的层次数量与内容可能是最佳的选择，其实并非如此，其中会话层和表示层几乎是空的，而数据链路层和网络层包含内容太多，有很多的子层插入，每个子层都有不同的功能。

网络连接

TCP/IP 意味着 TCP 和 IP 在一起协同工作。

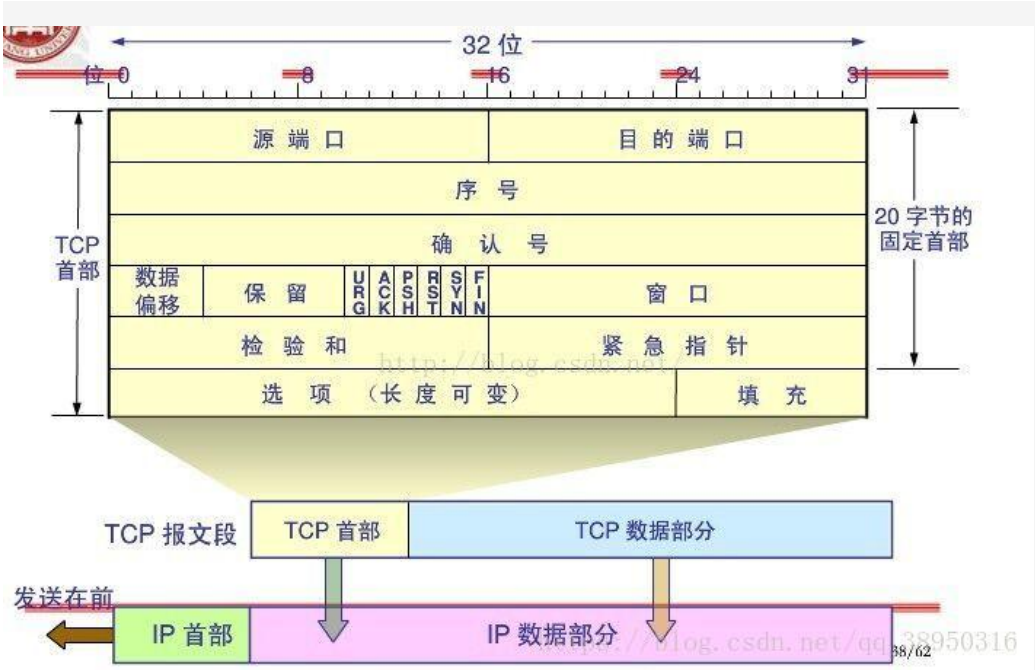
TCP 负责应用软件（比如你的浏览器）和网络软件之间的通信。

IP 负责计算机之间的通信。

TCP 负责将数据分割并装入 IP 包，然后在它们到达的时候重新组合它们。

IP 负责将包发送至接受者。

- **TCP 报文格式**



序列号 seq：占 4 个字节，用来标记数据段的顺序，TCP 把连接中发送的所有数据字节都编上一个序号，第一个字节的编号由本地随机产生；给字节编上序号后，就给每一个报文段指派一个序号；序列号 seq 就是这个报文段中的第一个字节的数据编号。

确认号 ack：占 4 个字节，期待收到对方下一个报文段的第一个数据字节的序号；序列号表示报文段携带数据的第一个字节的编号；而确认号指的是期望接收到下一个字节的编号；因此当前报文段最后一个字节的编号+1 即为确认号。

确认 ACK：占 1 位，仅当 ACK=1 时，确认号字段才有效。ACK=0 时，确认号无效

同步 SYN：连接建立时用于同步序号。当 SYN=1，ACK=0 时表示：这是一个连接请求报文段。若同意连接，则在响应报文段中使得 SYN=1，ACK=1。因此，SYN=1 表示这是一个连接请求，或连接接受报文。SYN 这个标志位只有在 TCP 建产连接时才会被置 1，握手完成后 SYN 标志位被置 0。

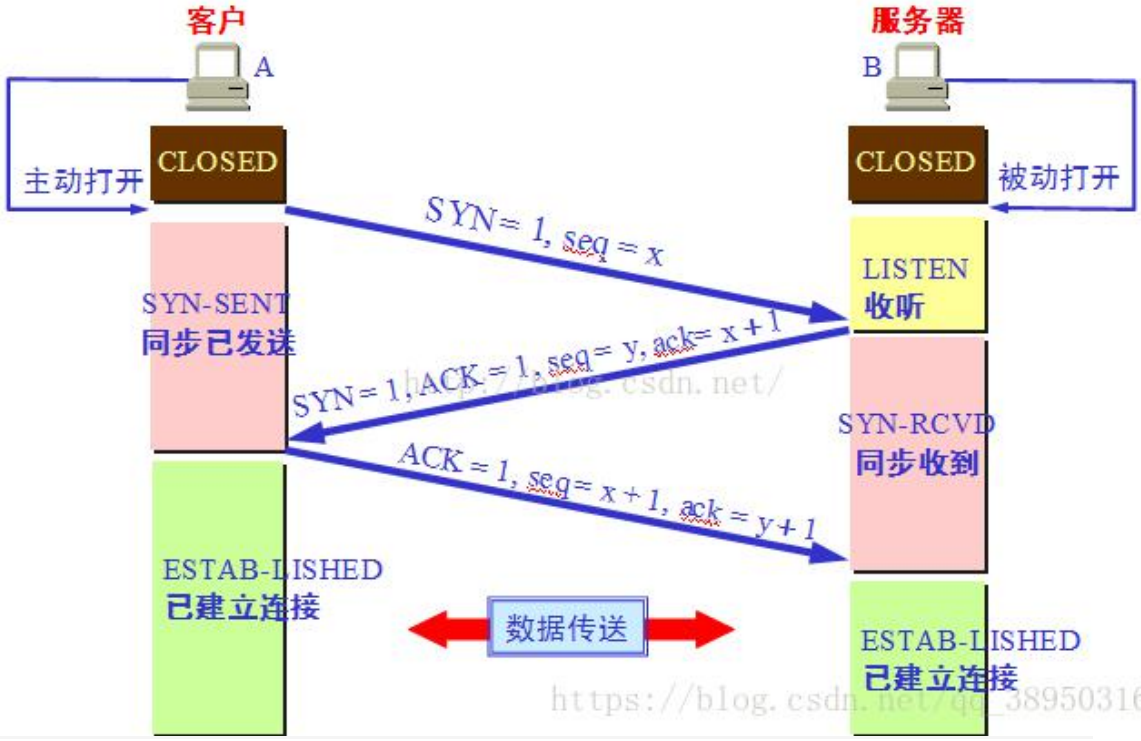
终止 FIN：用来释放一个连接。FIN=1 表示：此报文段的发送方的数据已经发送完毕，并要求释放运输连接

PS：ACK、SYN 和 FIN 这些大写的单词表示标志位，其值要么是 1，要么是 0；ack、seq 小写的单词表示序号。

字段	含义
URG	紧急指针是否有效。为1，表示某一位需要被优先处理
ACK	确认号是否有效，一般置为1。
PSH	提示接收端应用程序立即从TCP缓冲区把数据读走。
RST	对方要求重新建立连接，复位。
SYN	请求建立连接，并在其序列号的字段进行序列号的初始值设定。建立连接，设置为1
FIN	希望断开连接。

三次握手过程理解

所谓三次握手（Three-Way Handshake）即建立 TCP 连接，就是指建立一个 TCP 连接时，**需要客户端和服务端总共发送 3 个包以确认连接的建立。**在 socket 编程中，这一过程由**客户端执行 connect 来触发**，整个流程如下图所示：



第一次握手：建立连接时，客户端发送 syn 包（syn=j）到服务器，并进入 SYN_SENT 状态，等待服务器确认；SYN：同步序列编号（Synchronize Sequence Numbers）。

第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN_RECV 状态；

第三次握手：客户端收到服务器的 SYN+ACK 包，向服务器发送确认包 ACK(ack=k+1），此包发送完毕，客户端和服务器进入 ESTABLISHED（TCP 连接成功）状态，完成三次握手。

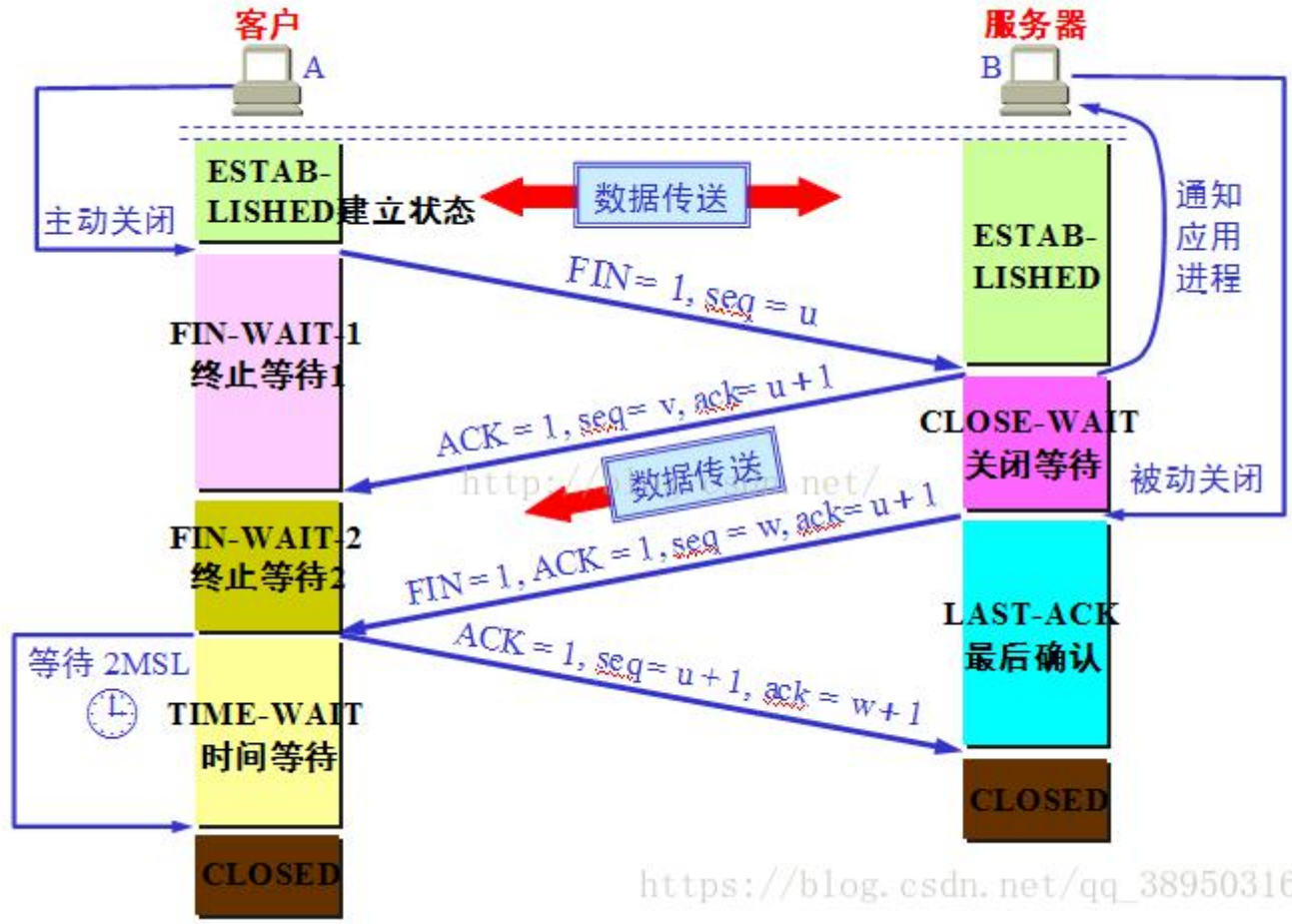
SYN攻击：

在三次握手过程中，Server发送SYN-ACK之后，收到Client的ACK之前的TCP连接称为半连接（half-open connect），此时Server处于SYN_RCVD状态，当收到ACK后，Server转入ESTABLISHED状态。SYN攻击就是Client在短时间内伪造大量不存在的IP地址，并向Server不断地发送SYN包，Server回复确认包，并等待Client的确认，由于源地址是不存在的，因此，Server需要不断重发直至超时，这些伪造的SYN包将产时间占用未连接队列，导致正常的SYN请求因为队列满而被丢弃，从而引起网络堵塞甚至系统瘫痪。SYN攻击时一种典型的DDOS攻击，检测SYN攻击的方式非常简单，即当Server上有大量半连接状态且源IP地址是随机的，则可以断定遭到SYN攻击了，使用如下命令可以让之现行：

```
#netstat -nap | grep SYN_RECV
```

- **四次挥手过程理解**

所谓四次挥手（Four-Way Wavehand）即终止 TCP 连接，就是指断开一个 TCP 连接时，**需要客户端和服务端总共发送 4 个包以确认连接的断开。**在 socket 编程中，这一过程由客户端或服务端任一方**执行 close 来触发**，整个流程如下图所示：



注：在 HTTP 工作中，关闭连接由服务器主动关闭

1）客户端进程发出连接释放报文，并且停止发送数据。释放数据报文首部，FIN=1，其序列号为 seq=u（等于前面已经传送过来的数据的最后一个字节的序号加 1），此时，客户端进入 FIN-WAIT-1（终止等待 1）状态。 TCP 规定，FIN 报文段即使不携带数据，也要消耗一个序号。

2）服务器收到连接释放报文，发出确认报文，ACK=1，ack=u+1，并且带上自己的序列号 seq=v，此时，服务端就进入了 CLOSE-WAIT（关闭等待）状态。TCP 服务器通知高层的应用进程，客户端向服务器的方向就释放了，这时候处于半关闭状态，即客户端已经没有数据要发送了，但是服务器若发送数据，客户端依然要接受。这个状态还要持续一段时间，也就是整个 CLOSE-WAIT 状态持续的时间。

3）客户端收到服务器的确认请求后，此时，客户端就进入 FIN-WAIT-2（终止等待 2）状态，等待服务器发送连接释放报文（在这之前还需要接受服务器发送的最后的数据）。

4）服务器将最后的数据发送完毕后，就向客户端发送连接释放报文，FIN=1，ack=u+1，由于在半关闭状态，服务器很可能又发送了一些数据，假定此时的序列号为 seq=w，此时，服务器就进入了 LAST-ACK（最后确认）状态，等待客户端的确认。

5)客户端收到服务器的连接释放报文后,必须发出确认,ACK=1,ack=w+1,而自己的序列号是 seq=u+1，此时，客户端就进入了 TIME-WAIT（时间等待）状态。**注意此时 TCP 连接还没有释放，必须经过 2* *MSL（最长报文段寿命）的时间后，当客户端撤销相应的 TCB 后，才进入 CLOSED 状态。**

6）**服务器只要收到了客户端发出的确认，立即进入 CLOSED 状态。**同样，撤销 TCB 后，就结束了这次的 TCP 连接。可以看到，服务器结束 TCP 连接的时间要比客户端早一些。

常见面试题

【问题 1】为什么连接的时候是三次握手，关闭的时候却是四次握手?

答：因为当 Server 端收到 Client 端的 SYN 连接请求报文后，可以直接发送 SYN+ACK 报文。其中 ACK 报文是用来应答的，SYN 报文是用来同步的。但是关闭连接时，当 Server 端收到 FIN 报文时，很可能并不会立即关闭 SOCKET，**所以只能先回复一个 ACK 报文，告诉 Client 端，"你发的 FIN 报文我收到了"。只有等到我 Server 端所有的报文都发送完了，我才能发送 FIN 报文，因此不能一起发送。故需要四步握手。**

【问题 2】为什么 TIME_WAIT 状态需要经过 2MSL(最大报文段生存时间)才能返回到 CLOSE 状态?

答：虽然按道理，四个报文都发送完毕，我们可以直接进入 CLOSE 状态了，但是我们必须假象网络是不可靠的，有可以最后一个 ACK 丢失。所以 **TIME_WAIT 状态就是用来重发可能丢失的 ACK 报文。**在 Client 发送出最后的 ACK 回复，但该 ACK 可能丢失。**Server 如果没有收到 ACK，将不断重复发送 FIN 片段。所以 Client 不能立即关闭，它必须确认 Server 接收到了该 ACK。**Client 会在发送出 ACK 之后进入到 TIME_WAIT 状态。Client 会设置一个计时器，等待 2MSL 的时间。**如果在该时间内再次收到 FIN，那么 Client 会重发 ACK 并再次等待 2MSL。**所谓的 2MSL 是两倍的 MSL(Maximum Segment Lifetime)。MSL 指一个片段在网络中最大的存活时间，**2MSL 就是一个发送和一个回复所需的最大时间。如果直到 2MSL，Client 都没有再次收到 FIN，那么 Client 推断 ACK 已经被成功接收，则结束 TCP 连接。**

【问题 3】为什么不能用两次握手进行连接?

答：**3 次握手完成两个重要的功能，既要双方做好发送数据的准备工作(双方都知道彼此已准备好)，也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。**

现在把三次握手改成仅需要两次握手，死锁是可能发生的。作为例子，考虑计算机 S 和 C 之间的通信，假定 C 给 S 发送一个连接请求分组，S 收到了这个分组，并发 送了确认应答分组。按照两次握手的协定，S 认为连接已经成功地建立了，可以开始发送数据分组。可是，C 在 S 的应答分组在传输中被丢失的情况下，将不知道 S 是否已准备好，不知道 S 建立什么样的序列号，C 甚至怀疑 S 是否收到自己的连接请求分组。在这种情况下，C 认为连接还未建立成功，将忽略 S 发来的任何数据分 组，只等待连接确认应答分组。而 S 在发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

【问题 4】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 还设有一个保活计时器，显然，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器每收到一次客户端的请求后都会重新复位这个计时器，时间通常是设置为 2 小时，若两小时还没有收到客户端的任何数据，服务器就会发送一个探测报文段，以后每隔 75 秒钟发送一次。若一连发送 10 个探测报文仍然没反应，服务器就认为客户端出了故障，接着就关闭连接。

参考资料——[TCP 的三次握手与四次挥手理解及面试题（很全面） - 李卓航 - 博客园 (cnblogs.com)]

为什么建立连接是三次握手，而关闭连接却是四次挥手呢?

这是因为服务端在 LISTEN 状态下，**收到建立连接请求的 SYN 报文后，把 ACK 和 SYN 放在一个报文里发送给客户端。**而关闭连接时，当收到对方的 FIN 报文时，仅仅表示对方不再发送数据了但是还能接收数据，**己方也未必全部数据都发送给对方了，所以己方可以立即 close，也可以发送一些数据给对方后，再发送 FIN 报文给对方来表示同意现在关闭连接**，因此，己方 ACK 和 FIN 一般都会分开发送。

为什么 TIME_WAIT 状态需要经过 2MSL(最大报文段生存时间)才能返回到 CLOSE 状态?

原因有二:

一、保证 TCP 协议的全双工连接能够可靠关闭

二、保证这次连接的重复数据段从网络中消失

先说第一点,如果 Client 直接 CLOSED 了,那么由于 IP 协议的不可靠性或者是其它网络原因,**导致 Server 没有收到 Client 最后回复的 ACK。**那么 Server 就会在超时之后继续发送 FIN，此时由于 Client 已经 CLOSED 了，就**找不到与重发的 FIN 对应的连接**，最后 Server 就会收到 RST 而不是 ACK，Server 就会以为是连接错误把问题报告给高层。这样的情况虽然不会造成数据丢失，但是却导致 TCP 协议不符合可靠连接的要求。所以，Client 不是直接进入 CLOSED，而是要保持 TIME_WAIT，当再次收到 FIN 的时候，能够保证对方收到 ACK，最后正确的关闭连接。

再说第二点，如果 Client 直接 CLOSED，然后又再向 Server 发起一个新连接，我们不能保证这个新连接与刚关闭的连接的端口号是不同的。也就是说有可能新连接和老连接的端口号是相同的。一般来说不会发生什么问题，但是还是有特殊情况出现：**假设新连接和已经关闭的老连接端口号是一样的，如果前一次连接的某些数据仍然滞留在网络中，这些延迟数据在建立新连接之后才到达 Server，由于新连接和老连接的端口号是一样的，又因为 TCP 协议判断不同连接的依据是 socket pair，于是，TCP 协议就认为那个延迟的数据是属于新连接的，这样就和真正的新连接的数据包发生混淆了。**所以 TCP 连接还要在 TIME_WAIT 状态等待 2 倍 MSL，这样可以保证本次连接的所有数据都从网络中消失。

参考资料——(1 条消息) TCP/IP 协议详解_王佳斌-CSDN 博客_tcp/ip 协议