<https://blog.csdn.net/weiyuefei/article/details/78779047>

http://blog.csdn.net/ls5718/article/details/51757467

一、HTTP协议和TCP协议

HTTP的长连接和短连接本质上是TCP长连接和短连接。HTTP属于应用层协议，在传输层使用TCP协议，在网络层使用IP协议。IP协议主要解决网络路由和寻址问题，TCP协议主要解决如何在IP层之上可靠的传递数据包，使在网络上的另一端收到发端发出的所有包，并且顺序与发出顺序一致。TCP有可靠，面向连接的特点。

二、HTTP协议的长连接和短连接

在HTTP/1.0中，默认使用的是短连接。也就是说，浏览器和服务器每进行一次HTTP操作，就建立一次连接，但任务结束就中断连接。如果客户端浏览器访问的某个HTML或其他类型的 Web页中包含有其他的Web资源，如JavaScript文件、图像文件、CSS文件等；当浏览器每遇到这样一个Web资源，就会建立一个HTTP会话。

但从 HTTP/1.1起，默认使用长连接，用以保持连接特性。使用长连接的HTTP协议，会在响应头有加入这行代码：

|  |
| --- |
| Connection:keep-alive |

在使用长连接的情况下，当一个网页打开完成后，客户端和服务器之间用于传输HTTP数据的 TCP连接不会关闭，如果客户端再次访问这个服务器上的网页，会继续使用这一条已经建立的连接。Keep-Alive不会永久保持连接，它有一个保持时间，可以在不同的服务器软件（如Apache）中设定这个时间。实现长连接要客户端和服务端都支持长连接。

HTTP协议的长连接和短连接，实质上是TCP协议的长连接和短连接。

三、TCP长连接和短连接：

我们模拟一下TCP短连接的情况，client向server发起连接请求，server接到请求，然后双方建立连接。client向server 发送消息，server回应client，然后一次读写就完成了，这时候双方任何一个都可以发起close操作，不过一般都是client先发起 close操作。为什么呢，一般的server不会回复完client后立即关闭连接的，当然不排除有特殊的情况。从上面的描述看，短连接一般只会在 client/server间传递一次读写操作

短连接的优点是：管理起来比较简单，存在的连接都是有用的连接，不需要额外的控制手段

接下来我们再模拟一下长连接的情况，client向server发起连接，server接受client连接，双方建立连接。Client与server完成一次读写之后，它们之间的连接并不会主动关闭，后续的读写操作会继续使用这个连接。

首先说一下TCP/IP详解上讲到的TCP保活功能，保活功能主要为服务器应用提供，服务器应用希望知道客户主机是否崩溃，从而可以代表客户使用资源。如果客户已经消失，使得服务器上保留一个半开放的连接，而服务器又在等待来自客户端的数据，则服务器将应远等待客户端的数据，保活功能就是试图在服务 器端检测到这种半开放的连接。

四、长连接和短连接的生命周期

短连接在建立连接后，完成一次读写就会自动关闭了。

正常情况下，一条TCP长连接建立后，只要双不提出关闭请求并且不出现异常情况，这条连接是一直存在的，操作系统不会自动去关闭它，甚至经过物理网络拓扑的改变之后仍然可以使用。所以一条连接保持几天、几个月、几年或者更长时间都有可能，只要不出现异常情况或由用户（应用层）主动关闭。

在编程中，往往需要建立一条TCP连接，并且长时间处于连接状态。所谓的TCP长连接并没有确切的时间限制，而是说这条连接需要的时间比较长。

五、怎样维护长连接或者检测中断

1、在应用层使用heartbeat来主动检测。

对于实时性要求较高的网络通信程序，往往需要更加及时的获取已经中断的连接，从而进行及时的处理。但如果对方的连接异常中断，往往是不能及时的得到对方连接已经中断的信息，操作系统检测连接是否中断的时间间隔默认是比较长的，即便它能够检测到，但却不符合我们的实时性需求，所以需要我们进行手工去不断探测。

探测的方式有两种：

2、改变socket的keepalive选项，以使socket检测连接是否中断的时间间隔更小，以满足我们的及时性需求。有关的几个选项使用和解析如下：

A、我们在检测对端以一种非优雅的方式断开连接的时候，可以设置*SO\_KEEPALIVE*属性使得我们在*2*小时以后发现对方的*TCP*连接是否依然存在。用法如下：

*keepAlive = 1*；

*setsockopt(listenfd, SOL\_SOCKET, SO\_KEEPALIVE, (void\*)&keepAlive, sizeof(keepAlive));*

*B、*如果我们不想使用这么长的等待时间，可以修改内核关于网络方面的配置参数，也可设置*SOCKET*的*TCP*层（*SOL\_TCP*）选项*TCP\_KEEPIDLE*、*TCP\_KEEPINTVL*和*TCP\_KEEPCNT*。

*TCP\_KEEPIDLE*：开始首次*KeepAlive*探测前的*TCP*空闭时间

*The tcp\_keepidle parameter specifies the interval of inactivity that causes TCP to generate a KEEPALIVE transmission for an application that requests them. tcp\_keepidle defaults to 14400 (two hours).*

*TCP\_KEEPINTVL*：两次*KeepAlive*探测间的时间间隔

*The tcp\_keepintvl parameter specifies the interval between the nine retries that are attempted if a KEEPALIVE transmission is not acknowledged. tcp\_keepintvl defaults to 150 (75 seconds).*

*TCP\_KEEPCNT*：断开前的*KeepAlive*探测次数

*The TCP\_KEEPCNT option specifies the maximum number of keepalive probes to be sent. The value of TCP\_KEEPCNT is an integer value between 1 and n, where n is the value of the systemwide tcp\_keepcnt parameter.*

如果心搏函数要维护客户端的存活，即服务器必须每隔一段时间必须向客户段发送一定的数据，那么使用*SO\_KEEPALIVE*是有很大的不足的。因为*SO\_KEEPALIVE*选项指*"*此套接口的任一方向都没有数据交换*"*。在*Linux 2.6*系列上，上面话的理解是只要打开*SO\_KEEPALIVE*选项的套接口端检测到数据发送或者数据接受就认为是数据交换。因此在这种情况下使用*SO\_KEEPALIVE*选项 检测对方是否非正常连接是完全没有作用的，在每隔一段时间发包的情况，*keep-alive*的包是不可能被发送的。上层程序在非正常断开的情况下是可以正常发送包到缓冲区的。非正常端开的情况是指服务器没有收到*"FIN"*或者*"RST"*包。