链滴

tcp 和 http 理解学习

作者：Jiafeimao

原文链接：https://ld246.com/article/1519916016508

来源网站：链滴

在 C# 编写代码，很多时候会遇到 Http 协议或者 TCP 协议，这里做一个简单的理解。
TCP 协议对应于传输层，而 HTTP 协议对应于应用层，从本质上来说，二者没有可比性。Http 议是建立在 TCP 协议基础之上的，当浏览器需要从服务器获取网页数据的时候，会发出一次 Http 请 。Http 会通过 TCP 建立起一个到服务器的连接通道，当本次请求需要的数据完毕后，Http 会立即将 CP 连接断开，这个过程是很短的。所以 Http 连接是一种短连接，是一种无状态的连接。所谓的无状 ，是指浏览器每次向服务器发起请求的时候，不是通过一个连接，而是每次都建立一个新的连接。如 是一个连接的话，服务器进程中就能保持住这个连接并且在内存中记住一些信息状态。而每次请求结 后，连接就关闭，相关的内容就释放了，所以记不住任何状态，成为无状态连接。
随着时间的推移，html 页面变得复杂了，里面可能嵌入了很多图片，这时候每次访问图片都需 建立一次 tcp 连接就显得低效了。因此 Keep-Alive 被提出用来解决效率低的问题。从 HTTP/1.1 起 默认都开启了 Keep-Alive，保持连接特性，简单地说，当一个网页打开完成后，客户端和服务器之 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 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。虽然这里使用 TCP 连接保持了一段时间，但是这个时间是有限范围的 到了时间点依然是会关闭的，所以我们还把其看做是每次连接完成后就会关闭。后来，通过 Session, ookie 等相关技术，也能保持一些用户的状态。但是还是每次都使用一个连接，依然是无状态连接。< p>
以前有个概念很容忍搞不清楚。就是为什么 Http 是无状态的短连接，而 TCP 是有状态的长连接 Http 不是建立在 TCP 的基础上吗，为什么还能是短连接？现在明白了，Http 就是在每次请求完成后 把 TCP 连接关了，所以是短连接。而我们直接通过 Socket 编程使用 TCP 协议的时候，因为我们自 可以通过代码区控制什么时候打开连接什么时候关闭连接，只要我们不通过代码把连接关闭，这个连 就会在客户端和服务端的进程中一直存在，相关状态数据会一直保存着。
在 C# 中会有 Socket，实际上 socket 是对 TCP/IP 协议的封装，Socket 本身并不是协议，而 一个调用接口 (API)。Socket 的出现只是使得程序员更方便地使用 TCP/IP 协议栈而已，是对 TCP/IP 协议的抽象，从而形成了我们知道的一些最基本的函数接口，比如 create、listen、connect、accep 、send、read 和 write 等等。
比较形象的描述：HTTP 是轿车，提供了封装或者显示数据的具体形式; Socket 是发动机，提供 网络通信的能力。对于从 C# 编程的角度来讲，为了方便，你可以直接选择已经制造好的轿车 Http 与服务器交互。但是有时候往往因为环境因素或者其他的一些定制的请求，必须要使用 TCP 协议， 时就需要使用 Socket 编程，然后自己去处理获取的数据。就像是你用已有的发动机，自己造了一辆 车，去从服务器交互。
HTTP/1.0 和 HTTP/1.1 都把 TCP 作为底层的传输协议。HTTP 客户首先发起建立与服务器 TCP 连接。一旦建立连接，浏览器进程和服务器进程就可以通过各自的套接字来访问 TCP。如前所述，客 端套接字是客户进程和 TCP 连接之间的 “门”，服务器端套接字是服务器进程和同一 TCP 连接之 的 “门”。客户往自己的套接字发送 HTTP 请求消息，也从自己的套接字接收 HTTP 响应消息。类 地，服务器从自己的套接字接收 HTTP 请求消息，也往自己的套接字发送 HTTP 响应消息。客户或服 器一旦把某个消息送入各自的套接字，这个消息就完全落入 TCP 的控制之中。TCP 给 HTTP 提供一 可靠的数据传输服务；这意味着由客户发出的每个 HTTP 请求消息最终将无损地到达服务器，由服务 发出的每个 HTTP 响应消息最终也将无损地到达客户。
C# 代码连接远程数据库用的是 TCP 协议。每次 new 一个 connection 的时候，connection.op n 就打开了这个 TCP 连接。connection.Close 的时候就关闭了这个连接。FTP 的底层也是 TCP， 过是长连接的。传输大文件比较快。 需要看具体场景。在服务器端，如果程序是采取的长连接的方 ，那么就能控制同时连接到这个服务器的连接个数，防止同时有多个连接。但是采取短连接的方式， 么就不能控制同时连接到这个服务器上的连接的个数，这也是一个优点，可以同时处理大量连接请求 但是如果连接请求量太大的话，可能造成服务器停止工作。
WebService 不需要连接，一秒中至少可以支持上万 / 十万的请求，每次请求然后释放，没有空 的内存消耗。一般不会限制同时连接的个数，这是优势。Message Queue 需要建立连接， 支持上 的连接就很吃力了。因为每个连接即使没有在请求数据，也会在内存中占用一定的空间存储。会限制 比如 SQL Server 数据库服务器，一般最多同时连接 16 个。
Http 协议一定通过指定的端口，80，所以一般计算机上不会限制这个端口，所以 Http 协议能 顺利通过所有机器上的防火墙。而使用 Socket 编程的话，就需要自己指定特定的端口，那么很可能 个端口是在某个环境中禁用的，那么就无法穿透防火墙。IIS 使用的是 80 端口，也就是这个程序一直 监听着这个端口。一旦发现有人要建立到这个端口的连接，他就会响应，然后建立连接。这里说的连 都是短连接。所以你对服务器上的网址的请求，都是通过 80 端口送到网站程序的。然后通过这个端

发送的客户端浏览器。</p>