**HTTP、TCP和Socket的概念和原理及其区别**

**一、HTTP**

**HTTP简介**

①HTTP协议是Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。  
②HTTP是一个基于TCP/IP通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。  
③HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于1990年提出，经过几年的使用与发展，得到不断地完善和扩展。目前在WWW中使用的是HTTP/1.0的第六版，HTTP/1.1的规范化工作正在进行之中，而且HTTP-NG(Next Generation of HTTP)的建议已经提出。  
④HTTP协议工作于客户端-服务端架构为上。浏览器作为HTTP客户端通过URL向HTTP服务端即WEB服务器发送所有请求。Web服务器根据接收到的请求后，向客户端发送响应信息。

**HTTP的主要特点**

主要特点  
**1、简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。**

**2、灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加以标记。**

**3.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。**

**4.无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。**

**5、支持B/S及C/S模式。**

**HTTP的理解（此段摘抄于**[**Http协议与TCP协议简单理解后续**](https://link.jianshu.com?t=http:/blog.csdn.net/sundacheng1989/article/details/52437128)**）**

**一、**  
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协议是建立在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请求。Http会通过TCP建立起一个到服务器的连接通道，当本次请求的数据完毕后，Http会立即将TCP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。所谓的无状态，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。如果是一个连接的话，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。而每次请求结束后，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。

**二、**  
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要建立一次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。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, Cookie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

**三、**  
为什么Http是无状态的短连接呢？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就把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，因为我们自己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，这个连接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。  
**四、**  
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，提供了网络通信的能力。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来与服务器交互。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，这时就需要使用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，自己造了一辆卡车，去从服务器交互。

HTTP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，客户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连接之间的“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。类似地，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。客户或服务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提供一个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，由服务器发出的每个HTTP响应消息最终也将无损地到达客户。

**HTTP的请求方法**

根据HTTP标准，HTTP请求可以使用多种请求方法。  
HTTP1.0定义了三种请求方法： GET, POST 和 HEAD方法。  
HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法。

GET 请求指定的页面信息，并返回实体主体。

HEAD 类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头

POST 向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。

PUT 从客户端向服务器传送的数据取代指定的文档的内容。

DELETE 请求服务器删除指定的页面。

CONNECT HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。

OPTIONS 允许客户端查看服务器的性能。

TRACE 回显服务器收到的请求，主要用于测试或诊断。

**HTTP的工作原理**

HTTP协议定义Web客户端如何从Web服务器请求Web页面，以及服务器如何把Web页面传送给客户端。HTTP协议采用了请求/响应模型。客户端向服务器发送一个请求报文，请求报文包含请求的方法、URL、协议版本、请求头部和请求数据。服务器以一个状态行作为响应，响应的内容包括协议的版本、成功或者错误代码、服务器信息、响应头部和响应数据。  
以下是 HTTP 请求/响应的步骤：

1、客户端连接到Web服务器

一个HTTP客户端，通常是浏览器，与Web服务器的HTTP端口（默认为80）建立一个TCP套接字连接。

2、发送HTTP请求

通过TCP套接字，客户端向Web服务器发送一个文本的请求报文，一个请求报文由请求行、请求头部、空行和请求数据

四部分组成。

3、服务器接受请求并返回HTTP响应

Web服务器解析请求，定位请求资源。服务器将资源复本写到TCP套接字，由客户端读取。一个响应由状态行、响应头部、

空行和响应数据4部分组成。

4、释放连接[TCP连接](http://www.jianshu.com/p/ef892323e68f)

若connection 模式为close，则服务器主动关闭[TCP连接](http://www.jianshu.com/p/ef892323e68f)，

客户端被动关闭连接，释放[TCP连接](http://www.jianshu.com/p/ef892323e68f);若connection 模式为

keepalive，则该连接会保持一段时间，在该时间内可以继续接收请求;

5、客户端浏览器解析HTML内容

客户端浏览器首先解析状态行，查看表明请求是否成功的状态代码。然后解析每一个响应头，响应头告知以下为若干字节的

HTML文档和文档的字符集。客户端浏览器读取响应数据HTML，根据HTML的语法对其进行格式化，并在浏览器窗口中显示。

例如：在浏览器地址栏键入URL，按下回车之后会经历以下流程：

1、浏览器向 DNS 服务器请求解析该 URL 中的域名所对应的 IP 地址;

2、解析出 IP 地址后，根据该 IP 地址和默认端口 80，和服务器建立[TCP连接] (http://www.jianshu.com/p/ef892323e68f);

3、浏览器发出读取文件(URL 中域名后面部分对应的文件)的HTTP 请求，该请求报文作为 [TCP 三次握手](http://www.jianshu.com/p/ef892323e68f)的第三个报文的数据发送给服务器;

4、服务器对浏览器请求作出响应，并把对应的 html 文本发送给浏览器;

5、释放 [TCP连接](http://www.jianshu.com/p/ef892323e68f);

6、浏览器将该 html 文本并显示内容;

**二、TCP协议**

**TCP协议主要是在传输层，三次握手四次挥手**

**①三次握手**

在TCP/IP协议中，TCP协议通过三次握手建立一个可靠的连接

三次握手

第一次握手：客户端尝试连接服务器，向服务器发送syn包（同步序列编号Synchronize Sequence Numbers），

syn=j，客户端进入SYN\_SEND状态等待服务器确认

第二次握手：服务器接收客户端syn包并确认（ack=j+1），同时向客户端发送一个SYN包（syn=k），即SYN+ACK包，

此时服务器进入SYN\_RECV状态

第三次握手：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状态，完成三次握手

**②四次挥手**

由于TCP连接时全双工的，因此，每个方向都必须要单独进行关闭，这一原则是当一方完成数据发送任务后，发送一个FIN来终止这一方向的连接，收到一个FIN只是意味着这一方向上没有数据流动了，即不会再收到数据了，但是在这个TCP连接上仍然能够发送数据，直到这一方向也发送了FIN。首先进行关闭的一方将执行主动关闭，而另一方则执行被动关闭，下图描述的即是如此。

第一次挥手：Client发送一个FIN，用来关闭Client到Server的数据传送，Client进入FIN\_WAIT\_1状态。

第二次挥手：Server收到FIN后，发送一个ACK给Client，确认序号为收到序号+1（与SYN相同，一个FIN占用一个序号），

Server进入CLOSE\_WAIT状态。

第三次挥手：Server发送一个FIN，用来关闭Server到Client的数据传送，Server进入LAST\_ACK状态。

第四次挥手：Client收到FIN后，Client进入TIME\_WAIT状态，接着发送一个ACK给Server，确认序号为收到序号+1，

Server进入CLOSED状态，完成四次挥手。

四次挥手

**三、Scoket**

**1、socket概念**

套接字（socket）是通信的基石，是支持TCP/IP协议的网络通信的基本操作单元。它是网络通信过程中端点的抽象表示，包含进行网络通信必须的五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，远地主机的IP地址，远地进程的协议端口。  
一个Socket是一对IP地址和端口。

Socket可以看成在两个程序进行通讯连接中的一个端点，一个程序将一段信息写入Socket中，该Socket将这段信息发送给另外一个Socket中，使这段信息能传送到其他程序中。

**2、socket的作用**

应用层通过传输层进行数据通信时，TCP和UDP会遇到同时为多个应用程序进程提供并发服务的问题。为了区别不同的应用程序进程和连接，许多计算机[操作系统](https://link.jianshu.com?t=http:/lib.csdn.net/base/operatingsystem" \t "_blank)为应用程序与TCP／IP协议交互提供了称为套接字(Socket)的接口，区分不同应用程序进程间的网络通信和连接。

**3、socket的原理**

**3.1 socket的实现方式**

生成套接字，主要有3个参数：通信的目的IP地址、使用的传输层协议(TCP或UDP)和使用的端口号。Socket原意是“插座”。通过将这3个参数结合起来，与一个“插座”Socket绑定，应用层就可以和传输层通过套接字接口，区分来自不同应用程序进程或网络连接的通信，实现数据传输的并发服务。

Host A上的程序A将一段信息写入Socket中，Socket的内容被Host A的网络管理软件访问，并将这段信息通过Host A的网络接口卡发送到Host B，Host B的网络接口卡接收到这段信息后，传送给Host B的网络管理软件，网络管理软件将这段信息保存在Host B的Socket中，然后程序B才能在Socket中阅读这段信息。

**3.2 Socket连接的实现方式**

要通过互联网进行通信，至少需要一对套接字，一个运行于客户机端，称之为ClientSocket，另一个运行于服务器端，称之为serverSocket。  
根据连接启动的方式以及本地套接字要连接的目标，套接字之间的连接过程可以分为三个步骤：服务器监听，客户端请求，连接确认。

服务器监听：是服务器端套接字并不定位具体的客户端套接字，而是处于等待连接的状态，实时监控网络状态。

客户端请求：是指由客户端的套接字提出连接请求，要连接的目标是服务器端的套接字。为此，客户端的套接字必须

首先描述它要连接的服务器的套接字，指出服务器端套接字的地址和端口号，然后就向服务器端套接字提出连接请求。

连接确认 ：是指当服务器端套接字监听到或者说接收到客户端套接字的连接请求，它就响应客户端套接字的请求，

建立一个新的线程，把服务器端套接字的描述发给客户端，一旦客户端确认了此描述，连接就建立好了。

而服务器端套接字继续处于监听状态，继续接收其他客户端套接字的连接请求。

**3.3 Socket与TCP/IP的关系**

创建Socket连接时，可以指定使用的传输层协议，Socket可以支持不同的传输层协议（TCP或UDP），当使用TCP协议进行连接时，该Socket连接就是一个TCP连接。  
socket则是对TCP/IP协议的封装和应用（程序员层面上）。也可以说，TPC/IP协议是传输层协议，主要解决数据 如何在网络中传输，而HTTP是应用层协议，主要解决如何包装数据。关于TCP/IP和HTTP协议的关系，网络有一段比较容易理解的介绍：  
“我们在传输数据时，可以只使用（传输层）TCP/IP协议，但是那样的话，如 果没有应用层，便无法识别数据内容，如果想要使传输的数据有意义，则必须使用到应用层协议，应用层协议有很多，比如HTTP、FTP、TELNET等，也 可以自己定义应用层协议。WEB使用HTTP协议作应用层协议，以封装HTTP文本信息，然后使用TCP/IP做传输层协议将它发到网络上。”  
我们平时说的最多的socket是什么呢，实际上socket是对TCP/IP协议的封装，Socket本身并不是协议，而是一个调用接口（API），通过Socket，我们才能使用TCP/IP协议。 实际上，Socket跟TCP/IP协议没有必然的联系。Socket[编程](https://link.jianshu.com?t=http:/www.2cto.com/kf)接口在设计的时候，就希望也能适应其他的网络协议。所以说，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、accept、send、read和write等等。网络有一段关于socket和TCP/IP协议关系的说法比较容易理解：  
“TCP/IP只是一个协议栈，就像操作系统的运行机制一样，必须要具体实现，同时还要提供对外的操作接口。这个就像操作系统会提供标准的编程接口，比如win32编程接口一样，TCP/IP也要提供可供程序员做网络开发所用的接口，这就是Socket编程接口。”  
实际上，传输层的TCP是基于网络层的IP协议的，而应用层的HTTP协议又是基于传输层的TCP协议的，而Socket本身不算是协议，就像上面所说，它只是提供了一个针对TCP或者UDP编程的接口。socket是对端口通信开发的工具,它要更底层一些.

**3.4 Socket与HTTP的关系**

由于通常情况下Socket连接就是TCP连接，因此Socket连接一旦建立，通信双方即可开始相互发送数据内容，直到双方连接断开。但在实际网络应用中，客户端到服务器之间的通信往往需要穿越多个中间节点，例如[路由器](https://link.jianshu.com?t=http:/www.2cto.com/net/router/)、网关、防火墙等，大部分防火墙默认会关闭长时间处于非活跃状态的连接而导致 Socket 连接断连，因此需要通过轮询告诉网络，该连接处于活跃状态。  
而HTTP连接使用的是“请求—响应”的方式，不仅在请求时需要先建立连接，而且需要客户端向服务器发出请求后，服务器端才能回复数据。

很多情况下，需要服务器端主动向客户端推送数据，保持客户端与服务器数据的实时与同步。此时若双方建立的是Socket连接，服务器就可以直接将数据传送给客户端；若双方建立的是HTTP连接，则服务器需要等到客户端发送一次请求后才能将数据传回给客户端，因此，客户端定时向服务器端发送连接请求，不仅可以保持在线，同时也是在“询问”服务器是否有新的数据，如果有就将数据传给客户端。

http协议是应用层的协义

有个比较形象的描述：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，提供了网络通信的能力。