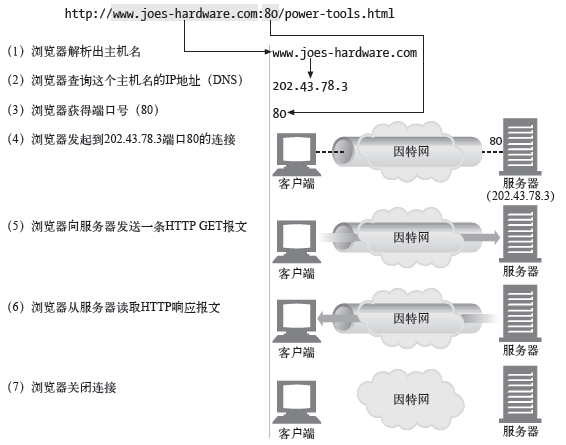
**一、HTTP如何使用TCP连接**

      世界上几乎所有的HTTP通信都是有TCP/IP承载的，它是一种常用的分组交换网络分层协议集。

HTTP连接实际就是TCP连接及其使用规则。web浏览器与服务器通过TCP连接的交互如下图：



TCP流是通过分段、由IP分组传送

**TCP数据是通过IP分组（或IP数据报）的小数据块来发送的。HTTP就是“HTTP OVER TCP OVER IP”这个协议栈中的最顶层。其安全版本HTTPS就是在HTTP和TCP之间插入了一个密码加密层（称为TLS或SSL）。HTTP要传送一条报文的时候，会以流的形式将报文数据的内容通过一条打开的TCP连接按照顺序传输。TCP收到数据流后，会将数据流分成被称之为段的小数据块，并将段封装在ip分组中，通过因特网进行传输。**

每个IP分组都包括：

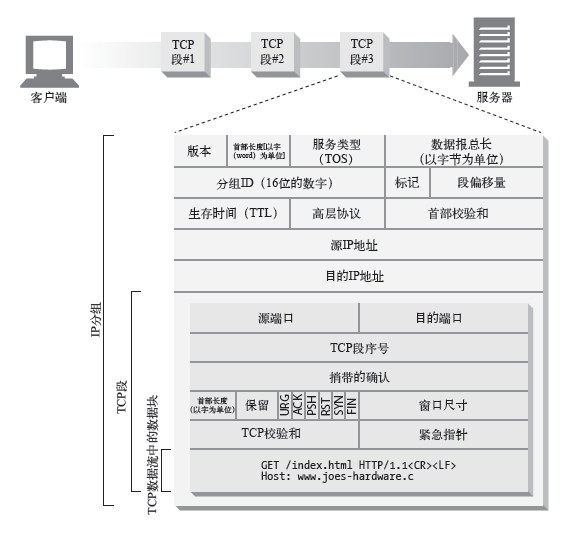
l      一个IP分组收首部（包含源和目的IP地址、长度等）

l      一个TCP段首部（（包含TCP端口号、控制标志等）

l      一个TCP数据块

保持TCP连接持续不断地运行

 一个TCP连接是由4个值来识别的，<源ip地址：源端口    目的ip地址：目的端口> ，这4个值唯一地定义了一条连接。两条不同的TCP连接不能拥有4个完全相同的地址组件值（但不同连接的部分组件可以拥有相同的值），TCP是通过端口号来保持所有这些连接的正确运行的。



 操作系统提供了一套操作TCP连接的工具，TCP客户端和服务器是通过tcp套接字接口来进行通讯的，如下图所示

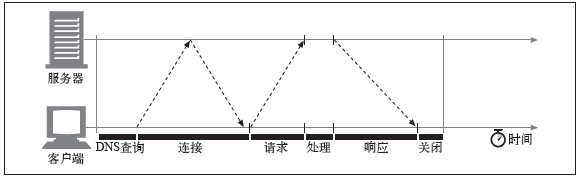
如上图，web服务器等待连接（S4），客户端根据URL判定出IP地址和端口号，并建立一条到服务器的TCP连接C3，（建立连接要花费一些时间，时间长短取决于服务器距离的远近、服务器的负载情况，以及因特网的拥挤程度），一旦建立了连接，客户端发送http请求C5，服务器读取请求S6。一旦服务器获取了整条请求报文，就会对请求进行处理，执行请求的动作S7，并将数据写回客户端，客户端读取数据C6，客户端读取数据，并对响应数据进行处理。

**二、对TCP性能的考虑**

     HTTP紧挨着TCP，位于其上层，所以HTTP事务的性能在很大程度上取决于底层TCP通道的性能。

**1.HTTP事务处理时的延时**

下图描绘了HTTP事务主要的连接、传输以及处理时延：



从上图可以看出整个HTTP事务的延时主要有以下：

1）.解析时延   DNS解析与DNS缓存

客户端首先需要根据URL确定Web服务器的IP地址和端口号，如果最近没有对URL中的主机名进行访问，通过DNS将URL中的主机名转换为IP地址可能会花费数十秒的时间。如果是近期访问过的主机名，那么在HTTP客户端的DNS缓存中，就会保存该主机名对应的IP地址。

2）.连接时延   TCP连接的建立

接下来，客户端会向服务器发送一条TCP连接请求，并等待服务器回送一个请求接受应答。每条新的TCP连接都会有连接新建时延，这个时间通常最多只有一两秒钟，但是如果数百个HTTP事务的话，这个值会快速叠加上去。

3）.传输时延   HTTP请求发送    HTTP响应返回

一旦连接建立起来之后，客户端就会通过新建立的TCP管道来发送HTTP请求，数据到达时，web服务器会从TCP链接中读取请求报文，并对其进行处理。因特网传输请求报文，以及服务器处理请求报文都需要时间。

4）.处理时延   HTTP报文处理

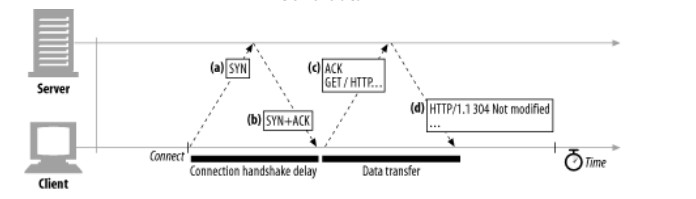
服务器会回送HTTP响应，这也需要花费时间。

**2.TCP网络时延、瓶颈**

**1）.TCP连接的握手时延**

建立一条新的TCP连接时，甚至是在发送任意数据前，TCP软件之间会交换一系列的IP分组，对连接的有关参数进行沟通。如果连接只用来传送少量的数据，这些交换过程就会严重降低HTTP的性能。

在发送数据之前，TCP要传送两个分组来建立连接：



 TCP连接握手需要经过以下几个步骤：

1） 请求新的TCP连接时，客户端要服务器发送一个小的TCP分组，这个分组中设置了一个特殊的SYN标记，说明这是一个连接请求。

2） 如果服务器接收了连接，就会对一些连接参数进行计算，并向客户端回送一个TCP分组，这个分组中的SYN和ACK标记都被置位，说明连接请求已被接受。

3） 最后，客户端向服务器回送一条确认信息，通知它连接已成功建立。现在的TCP栈都允许客户端在这个确认分组中发送数据

通常HTTP事务的交换数据量都不会太多，所以SYN/SYN+ACK握手会产生一个可测量的时延。最后可能的结果是：小的HTTP事务可能会在TCP建立上花费50%，或更多的时间。后面会介绍如何通过重用现存连接来减小这种TCP建立时延所造成的影响。

**2）.延迟确认机制：保证数据传输的成功**

由于因特网自身无法确保可靠的分组传输（路由器超负荷的话，可以随意丢弃分组），所以TCP实现了自己的确认机制来确保数据的成功传输。

**每个TCP段都有一个序列号和一个数据完整性校验和。每个段的接收者收到完好的段时，都会向发送者回送一个小的确认分组。如果发送者没有在指定的窗口时间内收到确认信息，发送者就会认为分组已被破坏或损毁，并重发数据。**

由于确认报文很小，所以TCP允许服务器在发往客户端的或者是客户端发往服务器的数据分组中队其进行“**捎带**”，将返回的确认信息和输出的数据分组结合在一起，更有效地利用网络。 HTTP的双峰特征-请求应答行为降低了捎带信息的可能性，通常，延迟确认算法会引入相当大的时延，根据操作系统的不同，可以调整或禁止延迟确认算法。

为了增加确认报文找到同向传输数据分组的可能性，很多TCP栈都实现了一种“延迟确认”的算法。**延迟确认算法会在一个特定的窗口时间(通常是100~200ms)内将输出确认放在缓冲区中，以寻找能够捎带它的输出分组。如果在时间段内没有输出分组符合条件，那么确认信息就放到单独的分组中进行传送。**

**3）.TCP慢启动**

**TCP连接会随着时间进行自我调谐，起初会限制连接的最大速度，如果数据成功传输，会随着时间的推移提高传输的速度。这种调谐被称为TCP慢启动，用于防止因特网的突然过载和拥塞。**

TCP慢启动限制了一个TCP端点在任意时刻可以传输的分组数。简单来说，每成功接收一个分组，发送端就有了发送另外两个分组的权限。当一个HTTP事务由大量数据要发送的时候，是不能一次性将所有分组都发送出去的，必须先发送一个分组，等待确认，然后可以发送两个分组，每个分组都必须被确认，这样就可以发送4个分组了，以此类推。这种方法被称为“打开拥塞窗口”。由于已调谐的连接要更快一些，可以通过“持久连接”重用现存的连接。

4）.数据聚集的Nagle算法

TCP发送大量包含数据的分组，会严重影响网络性能。Nagle算法试图在发送分组之前，绑定大量TCP数据，鼓励发送全尺寸的段，将数据缓存直至其他分组都被确认或者缓存中已足够全尺寸的段才会发送。这样就引入了一些性能问题。

 1 小的HTTP报文可能无法填满一个分组，可能会因为等待不会到来的数据产生时延。

 2 与延迟确认算法交互存在问题。Nagle算法阻止数据发送，直到有确认分组抵达，但确认分组自身会被延迟确认算法延迟，因为它在等待捎带它的数据包。

5）．TIME\_WAIT累积与端口耗尽

**当某个TCP端点关闭TCP连接时，会在内存中维护一个小的控制块，用来记录所关闭的连接的IP地址和端口号。这个信息通常只能存在一个小时间段。这个算法可以防止在短时间内创建、关闭具有相同IP和端口号的连接。**

TIME\_WAIT的作用：允许老的重复分组在网络中消失，防止最后ACK的丢失，可靠地实现TCP全双工通信的终止。

即使没有遇到端口耗尽问题，也要特别小心有大量连接处于打开状态的情况，在有大量打开连接或控制块的情况下，有些操作系统的速度会严重减缓。

**二、持久连接（keep-alive）**

Web客户端经常会打开到同一个站点的连接，而且相当一部分指向其他对象的超链通常都指向同一个站点。因此，初始化对某服务器HTTP请求的应用程序很可能会在不久的将来对那台服务器发起更多的请求。

因此，允许HTTP设备在事务处理结束之后将TCP连接保持在打开状态，以便为将来的HTTP请求重用现存的连接。如果服务器愿意为下一条请求保持打开状态，就在响应中包含相同的首部，如果响应中没有Connection:Keep-alive首部，客户端就认为服务器不支持Keep-alive，会在发回响应报文后关闭连接。

同时可以在响应首部中指定timeout（服务器希望将连接保持在活跃状态的时间）与max（服务器还希望为多少个事务保持此连接的活跃状态）.Keep-alive首部是可选的，但只有在提供了Connection:Keep-alive时才能使用它，以下是一个Keep-alive响应首部的例子。

Connection:Keep-Alive

**Keep-Alive:max=5,timeout=120**

**注意：**

**在HTTP/1.0中，keep-alive并不是默认使用的。客户端必须发送一个Connection:Keep-alive请求首部来激活Keep-alive连接。**

**通过检测响应中是否包含Connection:Keep-alive响应首部，客户端可以判断服务器是否会在发出响应之后关闭连接。**