第二部分　TCP的其他应用

第13章　HTTP：超文本传送协议

13.1　概述

超文本传送协议(Hypertext Transfer Protocol，HTTP)是万维网(World Wide Web，WWW，也简称为Web)的基础。本章我们介绍HTTP协议，在下一章讨论一个实际的Web服务器的运作，它综合运用了卷1和卷2中的许多有关实际应用的内容。但本章并不介绍Web或如何使用Web浏览器。

NFSnet骨干网提供的统计数据(见图13-1)表明，自1994年1月以来，使用HTTP协议的增长速度令人吃惊。

月份	HTTP	NNTP	FTP 数据	Telnet	SMTP	DNS	分组数 $\times 10^9$
1994.01	1.5 %	8.8 %	21.4 %	15.4 %	7.4 %	5.8 %	55
1994.04	2.8	9.0	20.0	13.2	8.4	5.0	71
1994.07	4.5	10.6	19.8	13.9	7.5	5.3	74
1994.10	7.0	9.8	19.7	12.6	8.1	5.3	100
1995.01	13.1	10.0	18.8	10.4	7.4	5.4	87
1995.04	21.4	8.1	14.0	7.5	6.4	5.4	59

图13-1　NFSnet骨干网上各种协议的分组数量百分比

以上这些百分数是基于分组数量统计得来的，而不是基于字节数的 (这些统计数据均可从`ftp://ftp.merit.edu/statistics`获得)。随着HTTP协议所占百分比的上升，FTP和Telnet的比例在下降。同时我们注意到，分组的总数量在整个 1994年都在上升，而1995年初开始下降。这是因为1994年12月有其他的骨干网开始取代NFSnet骨干网。不过，分组数的百分比仍然有效，它表明使用HTTP协议的通信量在增加。

Web的简单结构如图13-2所示。

如上图示，Web客户(通常称为浏览器)与Web服务器使用一个或多个TCP连接进行通信。知名的Web服务器端口是TCP的80号端口。Web浏览时客户端与服务器在TCP连接上进行通信，所采用的协议就是本章描述的HTTP，即超文本传送协议。我们也可看出，一个Web服务器可以通过超文本链接“指向”另一Web服务器。Web服务器上的这些链接并不是只可以指向Web服务器，还可以是其他类型的服务器，例如：一台FTP或是Telnet服务器。

尽管HTTP协议从1990就开始使用，但第一个可用的文档出现在 1993年([Berners-Lee 1993] 大致描述了HTTP协议的1.0版本)，但是该Internet草案早就过期了。虽然有新的可用的文档([Berners-Lee, Fielding和Nielsen 1995])出现，但是仍旧只是一个Internet草案。

[Berners-Lee, Connolly 1995]中描述了一种从Web服务器返回给客户进程的文档，称为HTML(超文本标记语言)文档。Web服务器还返回其他类型的文档 (图象，PostScript文件，无格式文本文件，等等)，我们将在本章的后面举例说明这些文档。

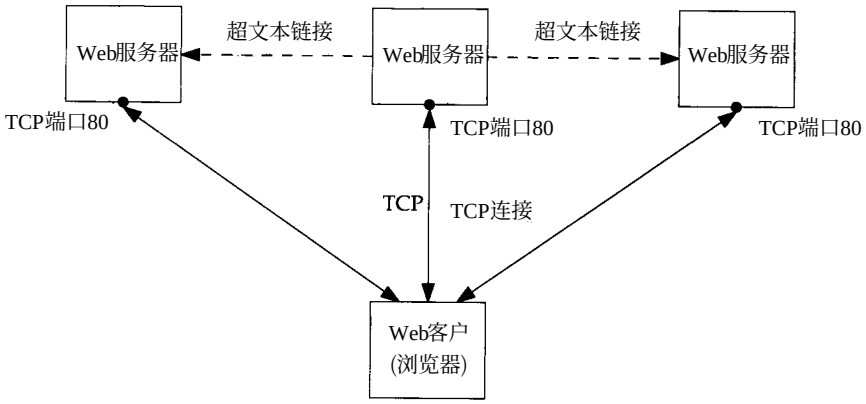

图13-2 Web客户 - 服务器结构

下一节我们将简要介绍HTTP协议和HTML文档，随后对协议进行详细描述。然后我们讨论一个流行的浏览器 (Netscape)是怎样使用该协议的， HTTP协议使用TCP的一些统计数据和HTTP协议的一些性能问题。[Stein 1995]讨论了运作一个Web站点的许多有关细节。

13.2 HTTP和HTML概述

HTTP是一个简单的协议。客户进程建立一条同服务器进程的 TCP连接，然后发出请求并读取服务器进程的响应。服务器进程关闭连接表示本次响应结束。服务器进程返回的文件通常含有指向其他服务器上文件的指针 (超文本链接)。用户显然可以很轻松地沿着这些链接从一个服务器到下一个服务器。

客户进程(浏览器)提供简单、漂亮的图形界面。 HTTP服务器进程只是简单返回客户进程所请求的文档，因此 HTTP服务器软件比HTTP客户软件要小得多。例如，NCSA版本1.3的Unix服务器由大约6 500行C代码写成，而X Window环境下的Unix Mosaic 2.5浏览器有约80 000行C代码。

我们可以用一个简单的方法来了解许多Internet协议是怎么工作的：那就是运行一个 Telnet的客户程序与相应的服务器程序通信。这种方法对 HTTP协议也是可行的，这是因为客户进程发送给服务器进程的语句包含有 ASCII命令(以回车和紧跟的换行符表示结束，称为 CR/LF)，服务器进程返回的内容也是以 ACSII字符行开始。HTTP协议使用的是8 bit的ISO Latin 1字符集，该字符集由ASCII字符及一些西欧语言中的字符组成 (以下网站可以找到各种字符集的信息：`http://unicode.drg`)。

下面是我们获取Addison-Wesley主页的例子。

```
sun % telnet www.aw.com 80        连接到服务器的80号端口
Trying 192.207.117.2...           由Telnet客户输出
Connected to aw.com.              由Telnet客户输出
Escape character is '^]'.         由Telnet客户输出
GET /                             我们只输入了这一行
<HTML>                            Web服务器输出的第一行
<HEAD>
<TITLE>AW's HomePage</TITLE>
</HEAD>
<BODY>
```

```
<CENTER><IMG SRC  = "awplogob.gif" ALT=" "><BR></CENTER>
<P><CENTER><H1>Addison-Wesley Longman</H1></CENTER>
Welcome to our Web server.
 ...                                    这里我们省略了33行输出
<DD><IMG ALIGN=bottom SRC="ball_whi.gif" ALT=" ">
Information Resource
<A HREF = "http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/MetaIndex.html">
Meta-Index</A>
 ...                                    这里我们省略了4行输出
</BODY>
</HTML>
Connection closed by foreign host.  由Telnet客户输出
```

我们只输入了GET /，服务器却返回了51行，共3611字节。这样，从该Web服务器的根目录下取得了它的主页。Telnet的客户进程输出的最后一行信息表示服务器进程在输出最后一行后关闭了TCP连接。

一个完整的HTML文档以<HTML>开始，以</HTML>结束。大部分的HTML命令都像这样成对出现。HTML文档含有以<HEAD>开始、以</HEAD>结束的首部和以<BODY>开始、以</BODY>结束的主体部分。标题通常由客户程序显示在窗口的顶部。 [Raggett, Lam, and Alexander 1996]中详细讨论了HTML。

下面这一行指定了一张图片(本例中为公司的标识)。

```
<CENTER><IMG SRC = "awplogob.gif" ALT=" "><BR></CENTER>
```

<CENTER>标志告诉客户程序将该图片放在屏幕中央，<IMG>标志含有该图片的相关信息。客户程序要取得该图片的文件名由SRC指示，ALT给出当使用纯文本客户程序时要显示的字符串(本例中是一个空字符串)。
实现强制换行。Web服务器程序返回这个主页时并不返回图片文件本身，它只返回图片文件的文件名，客户程序必须打开另一条 TCP连接来取得该文件(在本章的后面我们将看到为每一个指定的图像申请不同的连接将增加 Web的负载)。

下面这一行表示开始新的一段(<P>)。

```
<P><CENTER><H1>Addison-Wesley Longman</H1></CENTER>
```

这一段位于窗口的中央，它是第一级标题(<H1>)。客户程序可以选择怎样显示第一级标题(相对应的有2~7级标题)，通常采用比正常更大更粗的字体显示。

从上面可以看出，标记语言 (如HTML)与其他格式化语言 (如Troff, TeX, PostScript)之间的区别。HTML起源于SMGL，即标准通用标记语言 (Standard Generalized Markup Language) (http://www.sgmlopen.org包含更多的有关SGML的信息)。HTML指定了文档的数据和结构(本例中为一个1级标题)，但是没有指定浏览器怎样对文档进行格式化。

接着我们先忽略主页中跟在“ Welcome ”后面的很多问候语，看下面几行：

```
<DD><IMG ALIGN=bottom SRC="ball_whi.gif" ALT=" ">
Information Resource
<A HREF = "http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/MetaIndex.html">
Meta-Index</A>
```

其中<DD>指明一张定义表的入口。该入口以一张图片 (一个白球)开始，后面跟着文字"Information Resource Meta-Index"，最后一部分指明一个超文本链接 (<A>标志)和一个以"http://www.ncsa.uiuc.edu"开头的超文本引用(HREF属性)。像这样的超文本链接通

常在客户程序中被加上下划线或以不同的颜色来显示。当遇到上面所示的图象(公司的标志)时，服务器不会返回超链接引用的该图象或 HTML文档。客户程序通常会立即下载图象并显示在主页上，但对于超文本链接，除非用户点击(也就是说，把鼠标移到该链接上并单击)，否则客户程序通常不加处理。当用户点击了该链接，客户程序将打开一个到 www.ncsa.uiuc.edu 站点的HTTP连接，并执行GET，得到指明的文档。

类似http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/MetaIndex.html这样的表示被称为URL：统一资源定位符(Uniform Resource Locator)。URL的详细说明和意义在RFC 1738 [Berners-Lee, Masinter and McCahill 1994], 和RFC 1808 [Fielding 1995]中给出。URL是另一个重要的机制：统一资源标识符 URI(Uniform Resource Identifier)的一部分，URI还包括通用资源名称URN(Universal Resource Name)。RFC 1630 [Berners-Lee 1994]中描述了URI。URN试图比URL做得更好，但还没有制定出来。

大多数浏览器都提供查看 Web页面HTML源文件的功能，例如， Netscape和Mosaic都提供“ View Source ”的特性。

13.3 HTTP

上节的例子中，客户程序发出的GET /命令是HTTP版本0.9的命令，大多数服务器均支持这个版本(为了提供向后兼容性)。但目前HTTP的版本是1.0。因为1.0版本HTTP协议的客户程序在请求命令行中指出版本号，例如：

```
GET  /  HTTP/1.0
```

因此服务器能得知客户程序所采用的 HTTP协议的版本。本节我们将更详细地了解HTTP/1.0。

13.3.1 报文类型：请求与响应

HTTP/1.0报文有两种类型：请求和响应。 HTTP/1.0请求的格式是：

request-line
headers (0或有多个)
<blank line>
body (只对POST请求有效)

*request-line*的格式是：

*request request-URI HTTP*版本号

支持以下三种请求：

1) GET请求，返回*request-URI*所指出的任意信息。
2) HEAD请求，类似于GET请求，但服务器程序只返回指定文档的首部信息，而不包含实际的文档内容。该请求通常被用来测试超文本链接的正确性、可访问性和最近的修改。
3) POST请求用来发送电子邮件、新闻或发送能由交互用户填写的表格。这是唯一需要在请求中发送body的请求。使用POST请求时需要在报文首部Content-Length字段中指出body的长度。

对一个繁忙的 Web服务器进行采样，统计结果表明： 500 000个客户程序的请求中有

99.68%是GET请求，0.25%是HEAD请求，0.07%是POST请求。当然，如果是在一个接受比萨饼定购的站点上，POST请求的百分比将会更高。

HTTP/1.0响应的格式是：

status-line
headers (0个或有多个)
<blank line>
body

*status-line*的格式是：

*HTTP*版本号 *response-code response-phrase*

下面我们就要讨论这几个字段。

13.3.2 首部字段

HTTP/1.0的请求和响应报文的首部均可包含可变数量的字段。用一个空行将所有首部字段与报文主体分隔开来。一个首部字段由字段名 (如图13-3所示)和随后的冒号、一个空格和字段值组成，字段名不区分大小写。

报文头可分为三类：一类应用于请求，一类应用于响应，还有一类描述主体。有一些报文头(例如：Date)既可用于请求又可用于响应。描述主体的报文头可以出现在 POST请求和所有响应报文中。图13-3列出了17种不同的报文头，在[Berners-Lee, Fielding, and Nielsen 1995]中均有详细的描述。未知的报文头字段将被接收者忽略。我们讨论完响应代码后将回过头来看几个通用的报文头例子。

首部名称	请求？	响应？	主体？
Allow			•
Authorization	•		
Content-Encoding			•
Content-Length			•
Content-Type			•
Date	•	•	
Expires			•
From	•		
If-Modified-Since	•		
Last-Modified			•
Location		•	
MIME-Version	•	•	
Pragma	•	•	
Referer	•		
Server		•	
User-Agent	•		
WWW-Authenticate		•	

图13-3 HTTP报文首部的名称

13.3.3 响应代码

服务器程序响应的第一行叫状态行。状态行以 HTTP版本号开始，后面跟着 3位数字表示响应代码，最后是易读的响应短语。图 13-4列出了3位数字的响应代码的含义。根据第一位可以把响应分成5类。

使用这种3位的响应代码并不是任意的选择。我们将看到 NTTP(见图15-2)及其他的Internet应用如FTP、SMTP也使用这些类型的响应代码。

响　应	说　明
1yz	信息型，当前不用
	成功
200	OK，请求成功
201	OK，新的资源建立(post命令)
202	请求被接受，但处理未完成
204	OK，但没有内容返回
	重定向；需要用户代理执行更多的动作
301	所请求的资源已被指派为新的固定URL
302	所请求的资源临时位于另外的URL
304	文档没有修改(条件GET)
	客户差错
400	错误的请求
401	未被授权；该请求要求用户认证
403	不明原因的禁止
404	没有找到
	服务器差错
500	内部服务器差错
501	没有实现
502	错误的网关；网关或上游服务器来的无效响应
503	服务暂时失效

图13-4　HTTP 3位响应码

13.3.4　各种报文头举例

如果我们使用HTTP/1.0来获取上节列出的主页中所引用的标识图，则需要执行以下一些操作：

```
sun % telnet www.aw.com 80
Trying 192.207.117.2...
Connected to aw.com.
Escape character is '^]'.
GET /awplogob.gif HTTP/1.0                  我们输入了这一行
From: rstevens@noao.edu                     以及这一行
                                            然后输入一个空行表示请求结束
HTTP/1.0 200 OK                             服务器响应的第一行
Date: Saturday, 19-Aug-95 20:23:52 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: image/gif
Last-modified: Monday, 13-Mar-95 01:47:51 GMT
Content-length: 2859
                                            空行表示服务器响应头部的结束
                                        ←   这里收到了2859字节的二进制GIF图象
Connection closed by foreign host.          由Telnet客户输出
```

- 在GET请求中指出版本1.0。
- 发送一个可以被服务器记录的简单的报文头：From。

- 服务器返回的状态行给出了版本号、响应代码200和响应短语“OK”。
- Date报文头给出服务器上的时间和日期，通常是格林尼治时间。上例中服务器返回一个老式时间串。推荐的格式应是：缩写的天，日期中不含连字符， 4位数的年，如：

```
Date: Sat, 19 Aug 1995 20:23:52 GMT
```

- 服务器程序的类型和版本号分别是： NCSA Server版本1.3。
- MIME版本是1.0。在卷1的28.4节和 [Rose 1993]中有较多关于MIME的内容。
- 报文体的数据类型由Content-Type和Content-Encoding字段指出。Content-Type指出的是类型，类型后跟一‘ / ’，然后是子类型。本例中类型是 image，子类型是gif。HTTP使用的Internet媒体类型在最近的Assigned Number RFC文档中定义(本书写作时最新的文档是：[Reynolds and Postel 1994])。

其他的典型值是：

```
Content-Type: text/html
Content-Type: text/plain
Content-Type: application/postscript
```

如果报文主体是经过编码的，则 Content-Encoding报文头也会出现。例如：如果返回的报文中含有经过Unix的compress程序压缩的PostScript文件(通常带有.ps.Z后缀)，下面的两种报文头会同时出现：

```
Content-Type: application/postscript
Content-Encoding: x-compress
```

- Last-Modified指出了最后一次修改资源的时间。
- 图象文件的长度(2 859字节)在Content-Length报文头中指出。

在最后一个响应报文首部的后面，服务器程序紧跟着图象后发送了一个空行 (一个回车/换行对)。因为HTTP协议交换8 bit字节的数据，所以可以通过TCP连接发送二进制数据。这点不同于其他的Internet应用，特别是SMTP协议(卷1的第28章)，它通过TCP连接传输7 bit的ASCII字符，显式地将每字节的高位设置为0，阻止了二进制数据的交换。

User-Agent是公用的客户程序报文头，它用来标识客户程序的类型。下面是一些公用报文头的例子：

```
User-Agent: Mozilla/1.1N (Windows; I; 16bit)
User-Agent: NCSA Mosaic/2.6b1 (X11;SunOS 5.4 sun4m)  libwww/2.12 modified
```

13.3.5 例子：客户程序缓存

许多客户程序根据获取文件中的日期和时间在硬盘上缓存 HTTP文档。如果客户程序要获取的文档已存储在客户程序的缓存中，则客户程序将发送 If-Modified-Since 报文首部。这样，如果服务器程序发现该文档没有发生任何变化，就无需再发送一次该文档了。这称为条件GET请求。

```
sun % telnet www.aw.com 80
Trying 192.207.117.2...
Connected to aw.com.
Escape character is '^]'.
GET /awplogob.gif HTTP/1.0
If-Modified-Since: Saturday, 08-Aug-95 20:20:14 GMT
                                                    用空行结束客户请求
HTTP/1.0 304 Not modified
```

```
Date: Saturday, 19-Aug-95 20:25:26 GMT
Server: NCSA/1.3
MIME-version: 1.0
```

空行表示服务器响应头部的结束

```
Connection closed by foreign host.
```

上例中响应报文的响应代码为304，它表示文档没有变化。从TCP协议来看，这样做避免了将文档的主体(上例中是一个2 859字节的GIF图象)从服务器程序传送给客户程序。但是余下的TCP连接的开销(三次握手、终止连接的四个分组)还是必须的。

13.3.6 例子：服务器重定向

下面是一个服务器重定向的例子。我们试着去获取作者的主页，但是故意省略最后的“ / ”(这是用来指定目录的URL所必需的一部分)。

```
sun % telnet www.noao.edu 80
Trying 140.252.1.11...
Connected to gemini.tuc.noao.edu.
Escape character is '^]'.
GET /~rstevens HTTP/1.0
```

用空行结束客户请求

```
HTTP/1.0 302 Found
Date: Wed, 18 Oct 1995 16:37:23 GMT
Server: NCSA/1.4
Location: http://www.noao.edu/~rstevens/
Content-type: text/html
```

空行表示服务器响应头部的结束

```
<HEAD><TITLE>Document moved</TITLE></HEAD>
<BODY><H1>Document moved</H1>
This document has moved <A HREF="http://www.noao.edu/~rstevens/">here</A>.<P>
</BODY>
Connection closed by foreign host.
```

例子中响应报文的响应代码为302，表示所请求的URL已经被移动。Location报文首部指出了以“ / ”结尾的新位置。许多浏览器能自动去连接这个新的URL。但如果浏览器不愿意自动去访问这个新的URL，服务器程序也将返回一个可供浏览器显示的HTML文件。

13.4 一个例子

下面有一个使用流行的Web客户程序(Netscape 1.1N)的详细例子，通过它我们来逐一查看HTTP和TCP的使用。我们从Addison-Wesley的主页(http://www.aw.com)开始，然后到它指向的三个链接(都在www.aw.com上)，最后到卷1中描述过的页面上结束。共使用17条TCP连接，客户主机发送3 132字节，服务器主机返回47 483字节。在17条连接中，4条是为了传输HTML文档(共28 159字节)，还有13条是传输GIF图象(共19 324字节)。在进行这个会话前，先清除硬盘上的Netscape使用的缓存，迫使客户程序从服务器重新取得所有文件。我们还在客户主机上运行Tcpdump软件，记录客户程序发送和接收的所有报文段。

如我们所预期的，第一条TCP连接是访问主页(GET /)，主页的HTML文档共涉及了7个GIF图象。客户程序收到这个主页后，马上并行地打开4条TCP连接去获取前4个GIF图象。这是Netscape程序为了减少打开主页总时间的一种方法(大多数Web客户程序并不像这样，而是只能一次下载一个图象)。并行连接数量可由用户来配置，默认是4个。当这些连接中有一条结束时，客户程序会立即打开一条新的连接来获取下一个图象，直到客户程序取得全部7个图

象。图13-5表示这8条TCP连接的时间线，*y*轴是时间，单位为秒。

这8条TCP连接都由客户程序发起，依次使用1 114~1 121的8个端口号。而8条连接均由服务器程序关闭。我们把客户程序发送最初的SYN(客户的connect)看作连接的开始，客户程序收到服务器程序的FIN后发送FIN(客户的close)认为是连接的结束。取得这个主页以及它所涉及的所有7个图象共需要约12秒的时间。

下一章的图14-22中给出了由客户程序发起的第一条连接(端口 1 114)的Tcpdump分组跟踪情况。

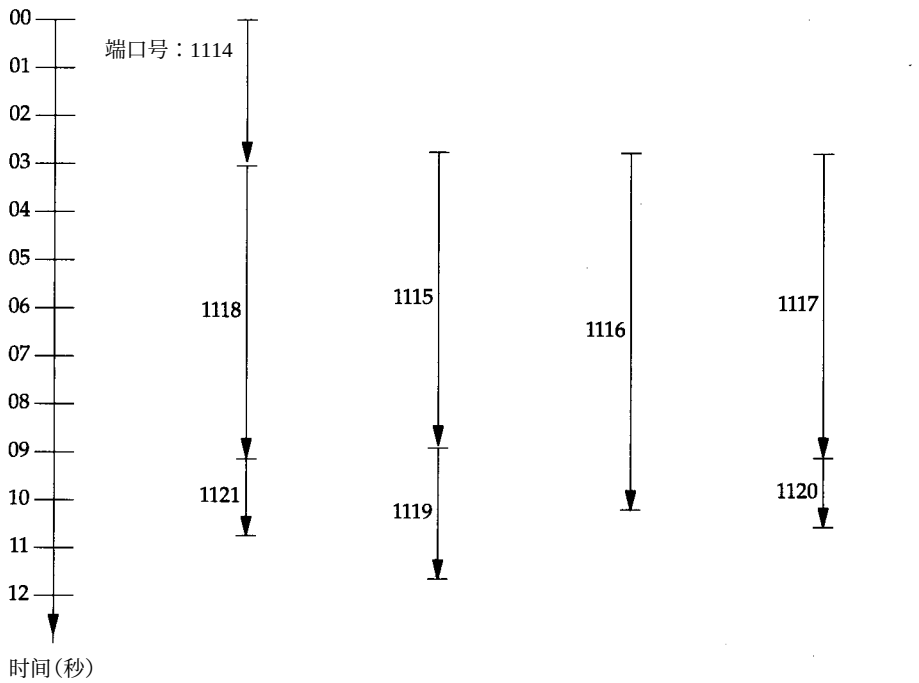

图13-5 一个主页和7个GIF图象的8条TCP连接的时间线

注意，端口号为1 115，1 116，1 117的三条连接是在第一条连接(端口号为1 114)结束之前建立的，这是因为Netscape的客户程序在读到第一条连接上的文件结束标志以后，并在关闭第一条连接之前发起三条无阻塞的连接。实际上，在图 14-22中我们可以注意到，客户程序在收到FIN标志后约半秒钟才发出FIN分组。

同时使用多条TCP连接是否真的能减少交互式用户所需的处理时间呢？为了测试这一点，我们在主机 sun上运行Netscape客户程序(图1-13)，还是来获取Addison-Wesley的主页。但这台主机是采用如今常用的方式连接Internet，即通过拔号调制解调器以28.8 Kb/s的速度连接Internet。我们修改客户程序的首选文件，对客户程序最大的连接数从1至7都进行了测试。测试时关闭了客户程序的硬盘缓存功能。在每一种最大连接数下客户程序均运行三次，取结果的平均值。图13-6是测试结果。

同时存在的连接数	总时间(秒)
1	14.5
2	11.4
3	10.5
4	10.2
5	10.2
6	10.2
7	10.2

图13-6 Web客户程序并行连接数与总时间的比较

上图可以看出，从1到4，随着连接数增加，总时间在减少。

但是如果用Tcpdump来跟踪这种交换，我们会发现，虽然用户可能把连接数设成超过 4，但是程序的极限是4。不管怎么说，超过4条连接后增加连接数对总时间即便有影响也是很小，不如从1~2，2~3，3~4那么明显。

图13-5所示的总时间比图13-6所示的最短时间(10.2秒)要多约2秒，这是因为客户主机的显示硬件速度有差异。图13-6所示的测试是客户程序运行在一台工作站上，而图13-5所示的测试客户程序运行在一台显示速度和运行速度均较慢的个人计算机上。

[Padmanabhan 1995]指出了多连接方法的两个问题。首先，这样做对其他协议不公平。例如，FTP协议获取多个文件时每次只能使用一条连接 (不包括控制连接)。其次，当在一条连接上遇到拥塞并执行拥塞避免 (在卷1的第21.6节中有描述)时，拥塞避免信息不会传递到其他连接上去。

对客户程序来说，同时对同一主机使用多条连接实际上使用的可能是同一条路径。如果处于瓶颈的路由器因为拥塞而丢弃某条连接的分组，那么其他连接的分组通过该路由器时也同样可能会被丢弃。

客户程序同时使用多个连接带来的另一个问题是容易造成服务器程序未完成的连接队列溢出，这样会使得客户主机重传它的 SYN分组而造成较大的时延。下一章我们讨论 Web服务器时，将在14.5节中详细讨论服务器程序的未完成连接队列。

13.5 HTTP的统计资料

在下一章中我们将仔细讨论TCP/IP协议族的一些特性和怎样在一个繁忙的 HTTP服务器上使用(和误用)它们。本节我们感兴趣的是一个典型的 HTTP连接到底是怎么回事。我们将使用下一章一开始要讲到的24小时的Tcpdump数据集。

图13-7列出了对近130 000个独立的HTTP连接进行统计的结果。如果客户程序非正常关闭连接，例如电话掉线等，我们就无法通过 Tcpdump的输出计算图中字节计数的中间值 (Median)或均值(Mean)或两者的值。存在一些因服务器超时而结束的连接会使图中连接持续时间的均值比正常值偏高。

	中 值	均 值
客户发送的字节数/连接	224	266
服务器发送的字节/连接	3 093	7 900
连接持续时间(秒)	3.4	22.3

图13-7 独立的HTTP连接的统计

大多数关于HTTP连接的统计均采用中间值和均值。中间值能较好地体现“正常”连接的情况，而均值则会因为少数长文件而较高。 [Mogul 1995b]中统计了200 000个HTTP连接，发现服务器返回数据量的中间值和均值分别为 1770字节和12 925字节。此文还对另一个服务器上约150万个检索进行统计，结论是返回数据量的中间值和均值分别为 958字节和2 394字节。[Braun and Claffy 1994] 列出的对NCSA服务器进行统计的结果是中间值为3 000字节，均值为17 000字节。明显可以看出服务器返回数据量的大小取决于该服务器上所提供的文件，不同的服务器之间有很大的差别。

在本节中迄今为止我们讨论的都是单个使用 TCP的HTTP连接。大多数运行 Web浏览器的用户会在一个HTTP会话期间访问给定服务器的多个文件。因为服务器可利用的信息就是客户主机的IP地址，所以要测量 HTTP会话的特性比较困难。多个用户能在同一时间利用同一客户主机访问同一个服务器。此外，还有一些组织把所有的 HTTP客户请求集中起来通过少数几个服务器(有时结合防火墙网关使用)去访问外部网的Web服务器，这样在Web服务器端看起来很多用户都在使用少数几个 IP地址(这些少数的服务器通常称为代理服务器，在 [Stein 1995]的第4章中对它进行了讨论)。不管怎么说，[Kwan, McGrath, and Reed 1995] 还是试图在NCSA服务器上对会话的特性进行测定，并定义一个会话最长持续时间为 30分钟。在这30分钟的会话中平均每个客户执行6个HTTP请求，服务器共返回95 000字节数据。

本节中提到的统计都是在服务器端进行测量的，因此结论都受服务器所提供的 HTTP文档类型的影响，例如，一个提供庞大气象图的 Web服务器平均每个HTTP会话所返回的字节数要比一个主要提供文本信息的服务器大得多。通常在 Web上跟踪大量客户程序对不同服务器的HTTP请求能获得更好的统计结果。 [Cunha, Bestavros, and Crovella 1995]中提供了一组测量数据。他们对4700个HTTP会话进行了测试，其中包括 591个不同用户对575772个文件的访问。测量的结果表明这些文件的平均长度为 11 500字节, 同时他们也提供了不同类型文件 (如HTTP、图象、声音、视频、文本等)的平均长度。通过其他测试，他们发现文件长度的分布状态曲线有一个大尾巴，即有大量的大型文件，这些文件影响了文件的平均字节数。他们发现大量被访问的是小文件。

13.6 性能问题

随着HTTP协议使用的增长 (图13-1)，它对Internet产生了广泛而重要的影响。 [Kwan, McGrath, and Reed 1995]中给出了NCSA服务器上HTTP协议一般应用的特性。 1994年，上文的作者对服务器的日志文件进行了为期五个月的检查后得出了一些结论。例如，他们注意到58%的客户请求是由个人计算机发起的，这类请求的每月增长率在 11%~14%之间。他们在文中还提供了一周中各天的请求数量、平均连接时间等统计数据。 [Braun and Claffy 1994]中提供了对NCSA服务器的其他分析。在这篇论文中，作者还讨论了 HTTP服务器可以通过缓存经常被访问的文档来提高性能。

影响交互式用户响应时间的最大因素是 HTTP协议中使用的TCP连接。前面我们看到每个要传输的文档使用一个TCP连接。[Spero 1994a]中以“ HTTP / 1.0 与TCP交互不协调 ”为标题对这个问题进行了描述。客户与服务器之间的 RTT和服务器的负载是影响响应时间的其他因素。

[Spero 1994a]也提出连接建立较慢 (卷1的20.6节中有描述)增加了时延。连接建立时间主要取决于客户请求报文和服务器的 MSS通告报文(通过Internet的客户连接，典型长度为 512或536字节)的长度。设想如果客户的请求报文小于或等于 512字节，一个MSS报文的长度为512字节，那么连接建立时间就不会长了 (但是要注意，很多基于伯克利的实现中的对 mbuf(在14.11节中有描述)的访问会引起连接建立慢的问题)。当客户的请求报文超过服务器的 MSS时，较慢的建立还要加上额外的RTT。客户请求报文的长度取决于浏览器软件。 [Spero 1994a]中提到当Xmoasic浏览器请求三个TCP报文段时发起了一个1 130字节的请求报文(这个请求报文共有42行，其中41行是Accept报文首部)。在13.4节的例子中， Netscape 1.1N浏览器共发起17

个请求，报文长度的范围是150~197字节，因此没有发生长时延的情况。图13-7列出了客户程序请求报文长度的中间值和平均值，从中可以看出大多数客户发起向服务器的请求不会引起长时延，但服务器的应答报文则会引起长时延。

我们刚刚提到，Mosaic客户程序会发出许多Accept报文首部，但这些报文首部并没有在图13-3中列出来(因为它们没有在[Berners-Lee, Fielding, and Nielsen 1995]中出现)。因为少数服务器不对这些报文首部作任何处理，所以在这个Internet草案中没有提到它们。这些报文首部的作用是告诉服务器，客户程序能接受哪些数据格式，如GIF图象、PostScript文件等。但也有少数服务器提供一个文档的不同格式的副本，而且目前还没有提供客户程序与服务器协商文档内容的方法。

另外重要的一点是：HTTP连接通常由服务器关闭，服务器经过TIME_WAIT时延后关闭连接，导致在繁忙的服务器上许多控制块停留在该状态。

[Padmanabhan 1995]和[Mogul 1995b]中建议客户与服务器保持一个打开的TCP连接，而不是服务器在发出响应后关闭连接。当服务器知道生成的响应报文的长度时才可以这样做，回想前面13.3.4节中我们提到的例子，Content-Length报文首部中指出GIF图象的大小。否则，服务器必须通过关闭连接来为客户程序指出响应的结尾。对协议作这样的修改必须同时修改客户端和服务器端。客户端规定Pragma: hold-connection报文首部，提供向后兼容的能力。如果服务器不能识别这种Pragma，就会忽略它，然后在发送完响应后关闭连接。这种Pragma允许新客户程序在尽可能情况下保持连接，同时访问新的服务器，还允许现有所有客户和服务器交互操作。

HTPP协议的下一版本(版本1.1)中可能会支持持续的连接，虽然具体怎么做可能会有变化。

在这里我们实际上提到了当前定义的三种服务器结束响应的方法。最好的办法是使用Content-Length报文首部，其次是服务器发送一个带有boundary=属性的Content-Type报文首部([Rose 1993]的6.1.1节中给出了怎样使用这种属性的例子，但是并非所有的客户程序都支持这种特性)。最差的选择(但最广泛运用的)便是服务器关闭连接。

Padmanabhan和Mogul也提出两种新的客户请求报文，用来允许服务器流水线式的响应。这两种请求是GETALL(服务器将在单个响应内返回一个HTML文档和所有内嵌的图象)和GETLIST(类似客户程序执行一系列的GET请求)。当客户程序确认在它的缓存中没有所要请求的任何文件时，可以使用GETALL报文。当客户程序发起对一个HTML文档的GET请求后，用GETLIST命令可以取得该HTML文档所引用的、不在缓存中的所有文件。

HTTP协议的一个重要问题是：面向字节的TCP数据流与面向报文的HTTP服务不匹配。一种理想的解决方法是：在TCP协议之上制定一个在HTTP客户和服务器之间、单个TCP连接之上、提供面向报文接口的会话层协议。[Spero 1994b]中描述了这样一种称为HTTP-NG的解决方法。HTTP-NG在单个TCP连接上提供多个会话。其中一个会话携带控制信息(客户请求和服务器响应报文)，其他的会话从服务器返回所请求的文件。通过TCP连接交换的数据包括一个8字节的会话首部(包含一些标志位、一个会话ID和所跟数据的长度)，会话首部后跟着这个会话的数据。

13.7 小结

HTTP是一个简单的协议。客户程序与服务器建立一个 TCP连接，发送请求并读回服务器的响应。服务器通过关闭连接来指示它的响应结束。服务器所返回的文件通常含有指针 (超文本链接)指向一些位于其他服务器的文件。用户可以轻松地跟随这些链接从一个服务器到另一个服务器。

客户请求是简单的 ASCII文本，服务器的响应也是以 ASCII文本开始(首部)，后面跟着数据(可以是ASCII或二进制数据)。客户程序软件 (浏览器)分析服务器的响应，并把它格式化输出，同时以高亮显示指向其他文档的链接。

通过HTTP连接传输的数据量较小。客户请求报文长度，为几百字节，服务器响应报文的典型值也在几百字节至 10 000字节间。因为一些大文档 (如图象或大的 PostScript文件)会将服务器响应报文长度的平均值拉大，所以 HTTP统计通常报告中间值。许多研究表明，服务器响应报文长度的中间值小于3000字节。

HTTP带来的最大的性能问题是每个文件使用一条 TCP连接。我们看一下 13.4节中提到的例子，为了打开一个主页，客户程序建立了 8条TCP连接。当客户请求报文的长度超过服务器通告的MSS时，缓慢的建立使每一个 TCP连接增加了额外的时延。另一个问题是：服务器进程正常关闭连接将引起在服务器主机上产生 TIME_WAIT时延，在一个繁忙的服务器上可以看到很多这种待终止的连接。

我们比较一下几乎与 HTTP协议同时开发的 Gopher协议。Gopher协议的文档号是 RFC 1436[Anklesaria et al. 1993]。从网络的观点来看， HTTP与Gopher非常相似。客户程序打开一条与服务器的连接 (Gopher使用70号端口)，并发起请求。服务器返回带有应答的响应，并关闭连接。它们的主要区别在于服务器送回给客户的报文的内容。尽管 Gopher协议允许服务器返回非文本信息，如GIF文件，但大多数Gopher客户程序是为ASCII终端设计的。因此Gopher服务器返回的文档，大多数是 ASCII文本文件。因为HTTP协议有明显的优势，所以写作本书时Internet上的许多站点已关闭了它们的 Gopher服务程序。当URL为`gopher://hostname`时，也有很多Web浏览器能识别Gopher协议，并与这些Gopher服务器通信。

HTTP协议的下一个版本 (HTTP / 1.1)将在1995年12月作为一个Internet草案公布。届时，包括认证(MD5签名)、持续的TCP连接、连接协商等方面均将有所增强。