第3章 应 用 层

TCP/IP 体系结构中的应用层是面向用户的。网络应用是互联网存在和发展的驱动力，是利用计算机网络为满足用户在不同领域、不同问题的需求而提供的软件。应用层协议与网络应用不同，它是网络应用的组成部分，是为实现网络应用的功能提供服务的。应用层中定义了很多协议，称为应用层协议。每个应用层协议都能通过不同主机中多个应用进程之间的通信来解决特定的应用问题。应用层研究的主要内容就是应用进程在通信时应遵循的协议。

本章主要介绍应用层协议的原理及相关知识，主要内容包括应用层协议的基本概念和原理，互联网上进程之间的通信方式——客户-服务器方式和对等计算模式，互联网上著名的网络应用万维网(World Wide Web，WWW)，域名系统(domain name system，DNS)，动态主机配置协议(dynamic host configuration protocol，DHCP)和电子邮件系统等。

应用层是学习网络协议非常好的起点，也最被人们熟悉。通过对应用层相关知识的学习，有助于更好地理解计算机网络背后诸多网络协议的运行原理，引发对很多计算机网络问题的思考，这些问题在学习传输层、网络层及数据链路层协议时也同样会碰到。本章从人们熟悉的万维网应用开始，探索计算机网络的应用层协议。

3.1 应用层协议原理

异彩纷呈的网络应用是计算机网络存在和蓬勃发展的根本原因，如果没有广泛应用于人们工作、生活和娱乐的网络应用，也就没有服务于这些应用的网络协议。

20 世纪 70 年代和 80 年代出现了文本电子邮件、远程访问计算机、文件传输和新闻组等基于文本的网络应用；20 世纪 90 年代中期出现了 Web 冲浪、搜索和电子商务等万维网应用；20 世纪末出现了即时通信(instant message，IM)和对等计算模式(peer to peer，P2P)的文件共享；现在，出现了基于互联网的语音电话、流媒体视频平台以及用户生成的视频发布(如抖音)等流行的语音和视频应用。此外，还有极具吸引力的多方在线游戏和在线教育平台，以及微信和微博等新一代社交网络应用，它们在互联网的应用层构建了引人入胜的社交网络。由此可见，网络的互连促进了人与人、人与物、物与物的联系，人们也越来越依赖于网络来进行活动与交流。

在 TCP/IP 体系结构中，应用层的主要功能是通过网络边缘的大量主机进行进程之间的交互来实现特定网络应用，网络应用程序的核心是能够运行在不同的端系统上并通过网络相互通信的程序。例如，在万维网应用中，有两个互相通信的应用程序：一个是运行在用户主机(笔记本计算机、平板计算机、智能手机等)上的浏览器程序，另一个是运行在万维网服务器主机上的 Web 服务器程序。又例如，使用了对等连接方式的 P2P 文件共享程序，在每个结点上安装的文件共享程序都具有类似的功能，每个结点既可以从其他结点获得文件，也可以向其他结点共享文件，从而可以充分利用庞大的终端资源。

从第 1 章已经知道，“主机 A 和主机 B 进行网络通信”实际上是指“运行在主机 A 上的某个网络应用程序和运行在主机 B 上的某个网络应用程序进行通信”。从操作系统的角度看，进行网络通信的实际上是进程(process)而不是程序，而进程是正在运行的应用程序的实例。当多个应用进程在同一个主机运行时，使用进程间通信(inter-process communication，IPC)机制相互通信，具体的通信规则由主机的操作系统确定。在计算机网络中，人们并不特别关注同一台主机上进程之间的通信，而更加关注运行在不同端系统(可能具有不同的操作系统)上的应用进程之间是如何通过计算机网络进行通信的，从而使得互联网上出现了大量异彩纷呈的应用。

在互联网的应用层中，应用进程之间的通信方式分为两种：客户-服务器(client-server，C/S)方式和对等连接方式，本节将分别进行介绍。

应用进程之间的通信必须遵循严格的规则，即遵循应用层协议。应用层协议应当规定如下内容。

(1) 交换的报文类型。例如，是请求报文还是响应报文。

(2) 各种报文类型的语法。例如报文中的各个字段及这些字段是如何描述的。

(3) 字段的语义。字段的语义就是这些字段中信息的含义。

(4) 确定一个进程何时以及如何发送报文，以及如何对报文进行响应的规则。

以上内容与第 1 章中介绍的协议三要素一致。本章会介绍几种重要的应用层协议，包括超文本传送协议(hypertext transfer protocol，HTTP)、域名系统(domain name system，DNS)、简单邮件传送协议(simple mail transfer protocol，SMTP)、动态主机配置协议(dynamic host configuration protocol，DHCP)等。

3.1.1 客户-服务器方式

客户-服务器方式是互联网上应用层进程之间最常用的通信方式。客户(client)和服务器(server)都是指通信中涉及的应用进程，客户-服务器方式描述的是进程之间服务和被服务的关系。例如，人们常用的万维网应用就是基于客户-服务器方式运作的。浏览器软件进程是客户进程，而在浏览器访问的远程主机上则驻留着万维网服务器进程。浏览器进程是服务请求方，它通过统一资源定位符(uniform resource locator，URL)主动向万维网服务器发出服务请求。万维网服务器进程是服务提供方，负责根据 URL 中对资源的描述寻找相应资源，并按照服务请求的要求将找到的资源发送给浏览器客户进程。客户与服务器的通信关系一旦建立，就可以进行双向通信，即客户和服务器都可以发送和接收信息，如图 3.1 所示。

在实际应用中，客户和服务器进程通常具有如下特点。

1. 客户进程的特点

(1) 客户进程被用户调用后，需要通信时主动向服务器进程发起通信并请求对方提供服务，因此它必须知道服务器进程的地址。这里的地址由主机的 IP 地址和进程绑定的端口(port)组成。在第 2 章中已经介绍了 IP 地址，关于端口的知识详见第 4 章。

(2) 客户进程不需要特殊的硬件和复杂的操作系统支持。

2. 服务器进程的特点

(1) 服务器进程是一种专门用来提供某种服务的进程，可以同时处理一个或多个远程

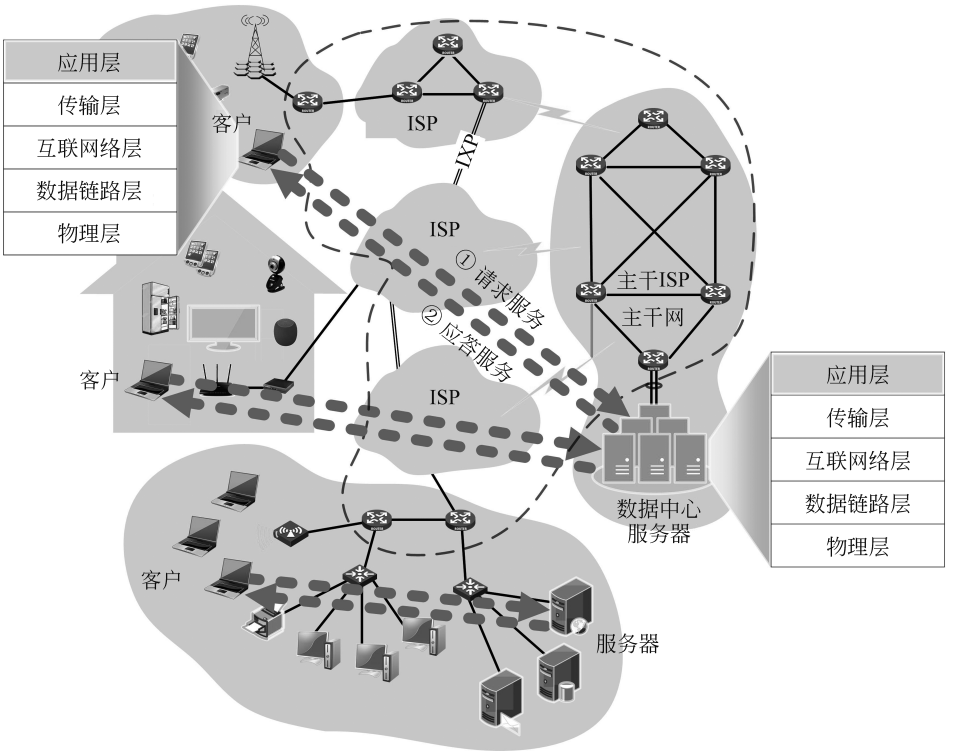


图 3.1　客户-服务器方式

或本地客户的请求。

(2) 当服务器进程被启动后,会立即自动调用并一直不断地运行,被动地等待并接受来自多个客户进程的通信请求,因此服务器进程不需要知道客户进程的地址。

(3) 服务器进程一般需要性能好的硬件和功能强大的操作系统支持。

值得注意的是,在客户-服务器方式下,客户之间不能直接通信。例如,在万维网应用中,两个浏览器之间并不能直接通信。使用客户-服务器方式进行通信的互联网应用有万维网、域名系统(DNS)、电子邮件等。在客户-服务器方式下,常常会出现一台单独的服务器主机无法满足所有客户请求的情况。为此,配备大量主机的数据中心常被用于创建强大的虚拟服务器,以支持数以百万计的并发用户请求。

虽然客户和服务器这两个词一般都是指通信中所涉及的应用进程,但在某些英文文献中,也会把运行客户进程的计算机称为 client,把运行服务器进程的计算机称为 server。因此需要根据上下文来判断 client 和 server 的具体含义。

3.1.2　对等计算模式

在客户-服务器方式中,服务器性能的好坏决定了整个系统的性能,当大量用户并发请求服务时,服务器就必然成为系统的瓶颈。对等计算模式的思想是整个网络中共享的内容不再被保存在中心服务器上,各个主机没有固定的客户和服务器角色的划分,即不明确区分哪个是服务请求方哪个是服务提供方,只要任意一对主机运行了对等连接软件(如 P2P 软件),就可以进行对等连接通信。例如,在 P2P 文件共享应用中,每个主机都同时具有下载、

上传的功能，通信双方都可以下载对方存储的共享资源。

在图 3.2 所示的对等计算模式中，主机 A、B、C 和 D 都运行了 P2P 软件，因此这几台主机都可进行对等通信（如 A 和 D，以及 B 和 C）。实际上，对等计算模式从本质上看仍然是客户-服务器方式，只是对等连接中的每台主机既作为客户访问其他主机的资源，也作为服务器提供资源给其他主机访问。例如，当主机 A 请求主机 D 的服务时，主机 A 是客户，主机 D 是服务器；但是若主机 A 又同时向主机 C 提供服务，则主机 A 又同时起着服务器的作用。

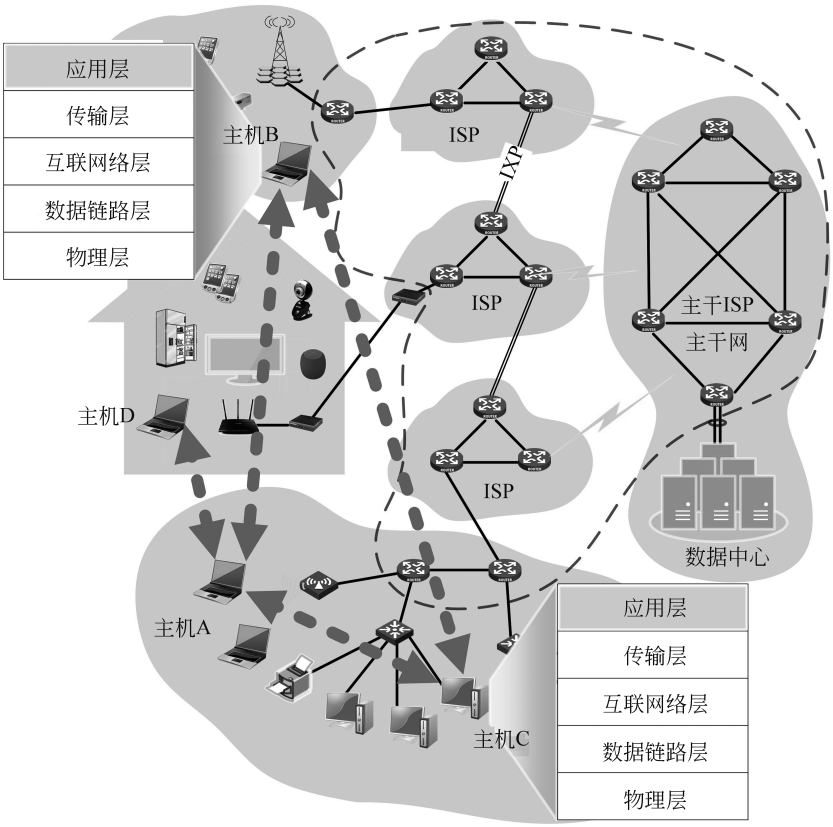


图 3.2　对等计算模式

对等计算模式最大的优点在于具有自扩展性（self-scalability）。例如，在一个 P2P 文件共享应用中，虽然每台对等的主机都由于请求文件而产生工作负载，但是也会因向其他对等主机分发文件而提高整个系统的总服务能力。这种情况下，通常不需要庞大的服务器基础设施和服务器带宽。这种高度的非集中式结构对于基于计算模式的应用会面临安全性、性能和可靠性等挑战。

3.1.3　进程通信

无论是客户-服务器方式，还是对等计算模式，人们总能将正在通信的一对进程中的一个标识为客户进程，另一个标识为服务器进程。例如，在万维网应用中，浏览器为客户进程，万维网服务器为服务器进程；再如，在 P2P 文件共享应用中，下载文件的进程为客户进程，

上传文件的进程为服务器进程。

客户进程和服务器进程的定义如下：当一对进程之间进行通信时，发起通信的进程称为客户进程，等待通信请求的进程称为服务器进程。

在两个不同的端系统上运行的应用进程是通过计算机网络交换报文来相互通信，以完成相应功能的。客户进程生成并向网络发送请求报文；服务器进程在接收到这些报文后通过响应报文进行回应。互联网 5 层协议栈中的应用层进程之间的互相通信情况如图 3.3 所示。

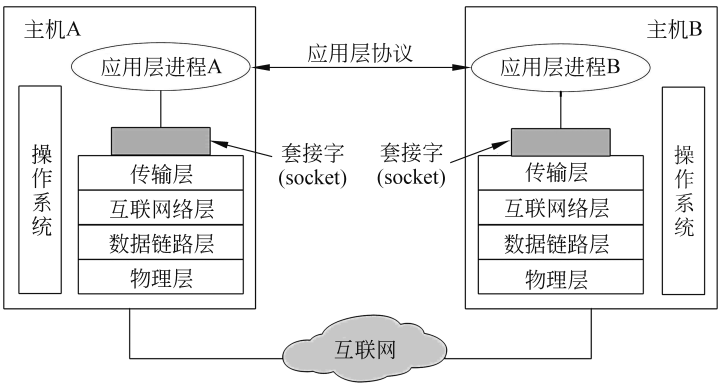


图 3.3　应用层进程之间的互相通信

如图 3.3 所示，顶层的部分是网络应用程序，也就是面向用户的各种网络应用，如浏览器、电子邮件客户端、万维网服务器、电子邮件服务器等应用进程。从一个应用进程向另一个应用进程发送报文时，需要通过下面的网络。应用进程通过一个名为套接字[①]的软件接口发送和接收报文。套接字就是应用进程和传输层协议之间的接口。发送方应用进程的套接字到接收方应用进程的套接字之间有传输层的逻辑信道。当一台主机上的应用进程 A 想向另外一台主机上的应用进程 B 发送报文时，就把报文推送到对应的套接字接口中。传输层逻辑信道可以把报文传送到目的进程的套接字内。一旦该报文抵达接收进程的套接字，应用进程 B 就可以读取报文并对其进行处理。

目前可供应用程序使用的 TCP/IP 应用程序接口（API）中，最著名的一种是美国加利福尼亚大学伯克利分校为 Berkeley UNIX 操作系统定义的，被称为套接字接口（socket interface）。套接字（socket）库中一般都包括地址解析器，用来向域名服务器发起查询，并将域名解析为 IP 地址，具体的工作过程将在 3.3 节进行介绍。当应用进程（客户进程或服务器进程）需要使用网络进行通信时，必须首先发出 socket 系统调用，请求操作系统会为其创建一个“套接字”。这个调用的实际效果是请求操作系统把网络通信所需要的存储器空间、CPU 时间、网络带宽等系统资源分配给该应用进程。操作系统会为这些资源分配一个套接字描述符（socket descriptor），然后把这个套接字描述符返回给应用进程。套接字描述符是一个小的整数。此后，应用进程所进行的网络操作都必须使用这个套接字描述符。几乎所有的网络系统调用都会把这个套接字描述符作为众多参数中的第一个参数。通过套接字描

① 套接字也称为插口。

述符，操作系统就可以识别出应该使用哪些资源来完成应用进程所请求的操作。通信完毕后，应用进程通过系统调用 close(关闭)套接字，通知操作系统回收与该套接字描述符相关的所有资源。由此可见，套接字也是应用进程为了获得网络通信服务而与操作系统进行交互时使用的一种机制。

套接字是一台主机内应用层与传输层之间的接口。应用程序开发者仅可以控制套接字在应用层的一切，而对传输层几乎没有控制权。应用程序开发者对于传输层的控制仅限于选择传输层协议和设定传输层参数。

一旦应用程序开发者选择了一种传输层协议，则该应用程序就建立在由该协议提供的传输层服务之上。最常用的传输层协议有提供面向连接和可靠传输服务的传输控制协议(TCP)，以及提供无连接、尽力而为服务的用户数据报协议(UDP)，在第 4 章会介绍这两种传输层协议。最重要的传输层参数是端口，端口可以唯一地标识本主机上的一个进程，端口与 IP 地址结合，可以唯一地标识互联网上一个进程。术语套接字(socket)的另外一种含义就是“IP 地址＋端口”，它也可以理解为进程地址。端口分为系统端口、用户端口和动态端口，其中系统端口和用户端口用于服务器进程，而动态端口用于客户进程。例如，万维网服务器所用 TCP 的 80 端口就属于系统端口。关于端口的概念和作用，会在第 4 章详细介绍。

3.2 万 维 网

3.2.1 万维网概述

20 世纪 90 年代之前，互联网的主要使用者还是研究人员、学者和大学生，他们登录远程主机，在本地主机和远程主机之间传输文件、收发新闻、收发电子邮件等。此时的互联网基本上不为学术界和研究界之外的人所知。直到 1989 年，欧洲原子核研究组织(CERN)的蒂姆·伯纳斯·李(Tim Berners Lee)发明了万维网，这种情况才得到改变。万维网极大地方便了非专业人员对网络的使用，它的出现将互联网带入了千家万户。万维网的影响力远远超出了专业技术范畴，已经进入电子商务、远程教育、远程医疗与信息服务等领域。目前，万维网应用是互联网上使用最方便、最受用户欢迎的网络应用。

万维网(World Wide Web，WWW)简称为 Web。它并非某种特殊的计算机网络，而是一个大规模的、联机式的信息存储空间，是运行在互联网上的一个超大规模的分布式应用。万维网是一个分布式的超媒体(hypermedia)系统，它是超文本(hypertext)系统的扩展。一个超文本由多个信息源链接而成，用户利用链接可以找到其他文档，而这些文档又可以包含其他链接。链接到其他文档的字符串称为超链接(hyperlink)。超文本与超媒体的区别是文档内容不同，超文本文档仅包括文本信息，而超媒体文档还包括图形、图像、声音、动画、活动视频等信息。带有超链接的超媒体可以非常方便地从互联网上的一个站点访问另一个站点。

万维网应用有很多组成部分，包括文档格式的标准，如超文本标记语言(HTML)、万维网浏览器、万维网服务器以及应用层协议——超文本传送协议(HTTP)。其中，HTTP 作为应用层协议是万维网应用中非常重要的组成部分，万维网应用就是建立在 HTTP 之上进行客户-服务器通信的。

万维网浏览器是万维网应用的客户程序，世界上第一个图形界面的浏览器是于 1993 年 2 月诞生的 Mosaic。目前比较流行的浏览器有 Chrome 浏览器、火狐（Firefox）浏览器、Microsoft Edge 浏览器、Safari 浏览器、Opera 浏览器等。浏览器最重要的部分是渲染引擎，也就是浏览器内核，负责对网页内容进行解析和显示。各种浏览器采用了不同的内核，例如，Chrome 浏览器使用的内核是 Blink，火狐浏览器使用的内核是 Gecko，Safari 浏览器使用的内核是 Webkit 等。不同的浏览器内核对网页内容的渲染效果大体一致，但细节有所不同。

万维网应用通过统一资源定位符（uniform resource locator，URL）定位信息资源，通过超文本标记语言（HTML）描述信息资源，通过超文本传送协议（HTTP）传递信息资源。URL、HTML 和 HTTP 这 3 个规范构成了万维网的核心构建技术，是支撑着万维网运行的基石。用通俗一点的语言来说，浏览器将 URL（例如 http://www.zzu.edu.cn）封装入 HTTP 请求报文，发给万维网服务器；万维网服务器收到该请求报文后，利用 URL 找到资源，将该资源封装入 HTTP 响应报文发回给浏览器；浏览器解析并渲染后展示给用户。浏览器与万维网服务器之间使用 HTTP 的交互过程如图 3.4 所示。

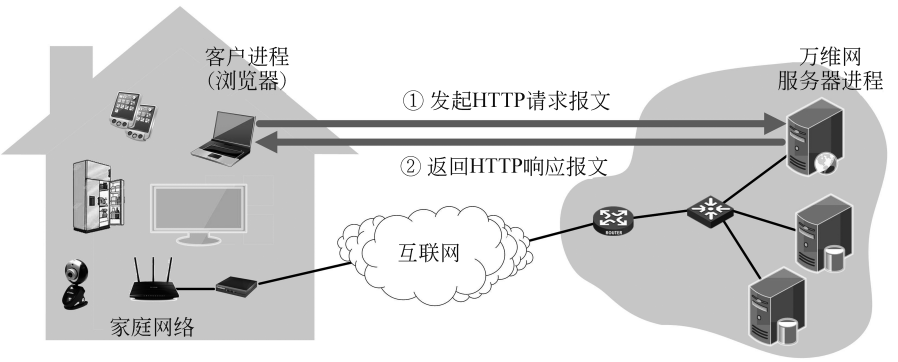


图 3.4　HTTP 的请求和响应过程

综上所述，URL、HTML 和 HTTP 这 3 个规范解决了万维网应用面对的 3 个关键问题。

（1）用 URL 解决了如何标识分布在整个互联网上的资源的问题，本节后面会加以介绍。

（2）用 HTML 解决了万维网文档以及超链接的标准化问题，使不同人员创作的不同风格的万维网文档，都能以统一的形式在各种主机上显示出来，同时使跨越站点的资源访问更加方便，本节后面会加以介绍。

（3）用 HTTP 解决了万维网上的信息资源的传递问题，会在 3.2.2～3.2.5 节加以介绍。

1. 统一资源定位符

统一资源定位符（URL）用来表示互联网上资源的位置和访问这些资源的方法。URL 给资源的位置提供一种抽象的识别方法，并用这种方法给资源定位。只要能够对资源定位，系统就可以对资源进行各种操作，如存取、更新、替换和查找其属性。这里所说的“资源”是指在互联网上可以被访问的任何对象，包括文件目录、文件、文档、图片、声音、视频等，以及与互联网相连的任何形式的数据。

由此可见，URL 实际上就是资源在互联网上的地址，相当于一个文件名在互联网范围的扩展。因此，URL 是指向互联网上的主机中的任何可访问对象的指针。显然，互联网上的所有可访问资源，都必须有一个唯一的 URL。

由于访问不同资源所使用的协议不同，所以 URL 还必须指出访问某个资源时所使用的协议。URL 的一般形式由以下 3 部分组成：

<协议>://<主机>:<端口号>/<路径>

URL 的第一部分是最左边的“<协议>://”，它指出访问该资源所使用的协议，或称为服务方式。目前，最常用的协议就是 HTTP 和 HTTPS(HTTP over secure socket layer)。<协议>后面紧跟着的“://”是规定的格式。

URL 的第二部分是“<主机>:<端口>”，它指出保存该资源的主机和处理该 URL 的服务器进程。其中“<主机>”指明保存该资源的主机的域名或者 IP 地址，“<端口>”指明处理该 URL 的进程。如果服务器上采用的端口是已经在 IANA 注册过的熟知端口，则“:<端口>”可以省略。如 HTTP 在 IANA 注册的熟知端口是 80 端口，当 Web 服务器采用 80 端口时，就可以省略“:<端口>”部分。

URL 的第三部分是“/<路径>”，它指出资源在该主机中的具体位置，如目录和文件名等。其中“/”代表根目录，根目录是一个逻辑目录，它可以映射到主机上的某个物理目录，映射关系由服务器程序指定。“<路径>”是一个相对于根目录的相对路径。如果在服务器上设置了某目录下的默认资源，则“<路径>”可以仅指明资源保存的目录，而省略文件名，这表示访问该目录下的默认资源；如果服务器上设置了根目录下的默认资源，则“/<路径>”部分可以省略，这表示访问根目录下的默认资源。

例如，郑州大学主页的 URL 为“http://www.zzu.edu.cn”，省略了 HTTP 的熟知 80 端口，也省略了“/<路径>”部分，它代表使用 HTTP 访问主机 www.zzu.edu.cn 上的根目录下的默认资源，与目的主机的 80 端口绑定的服务器进程负责处理该 URL 指定的资源的访问。当点击郑州大学主页中的“学校概况”超链接时，将访问另一个页面，该页面的 URL 为“http://www.zzu.edu.cn/xxgk/xxjj.htm”，这个 URL 与网站主页 URL 中的协议、主机以及端口都相同，不同的仅是路径。在路径中，“/xxgk/”是目录名，“xxjj.htm”是要访问的资源的文档名。

2. HTML 文档

万维网引入了超文本标记语言(HTML)作为制作万维网页面的标准语言，以消除不同计算机之间信息交流的障碍。由于 HTML 易于掌握且实施简单，因此它很快就成为万维网的重要基础。官方的 HTML 标准由万维网联盟(WWW consortium，W3C)负责制定。从 1993 年 HTML 问世开始，W3C 就不断地对其版本进行更新，直到 1997 年 HTML 4.0 推出后，在相当长的时间内，HTML 没有大的版本更新，非常稳定。HTML 4.0 版本成为应用最广泛的 HTML 版本。RFC2854 对 HTML 4.0 版本之前的历史进行了综述。2014 年，W3C 发布了 HTML 5.0 版本，它是目前最新的版本。HTML 5.0 中增加了在网页内嵌入音频、视频以及交互式文档等功能，随着时间推移，HTML 5.0 的应用越来越广泛，目前主流的浏览器都支持 HTML 5.0。

HTML 是一种标记语言，或者说是一种描述如何格式化文档的语言。HTML 使用标

记标签(markup tag)来描述网页文档,HTML 标记标签通常简称为 HTML 标签。HTML 标签是由“＜＞”包围的关键词,例如＜html＞。HTML 标签通常是成对出现的,例如＜body＞和＜/body＞。标签对中的第一个标签是开始标签,第二个标签是结束标签。开始和结束标签也被称为开放标签和闭合标签。开始标签和结束标签之间可以为空,也可以包含文本,还可以嵌套其他标签。标签之间的文本通常是可显示的内容,而标签可以带有参数,这些参数称为属性,用来指明内容显示的格式等。HTML 标签的组成如下:

```
<tag-name[[attribute-name[=attribute-value]]…]>(文本内容)</tag-name>
```

从开始标签到结束标签的所有代码称为 HTML 元素。HTML 文档由一组嵌套的元素组成。完整的 HTML 文档的结构如图 3.5 所示。

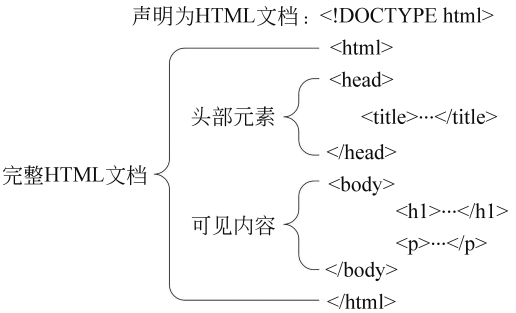


图 3.5　HTML 文档的结构

HTML 定义了几十种元素,用来定义不同的对象,如＜img＞元素用来定义图像、＜p＞元素用来定义段落、＜a＞元素用来定义超链接等。

HTML 的目标是指定文档的结构,而不是文档的外观。为了控制文档的呈现方式,通常会使用层叠样式表(cascading style sheets,CSS)语言为 HTML 文档定义布局,描述如何显示 HTML 元素。在浏览器上显示的字体、颜色、背景颜色或图片、边距、高度、宽度等方面,都可以通过 CSS 能够给出精确的规定。

HTML 文档分为静态文档、动态文档和活动文档 3 种。

(1) 静态 HTML 文档。静态 HTML 文档在创作完毕后就存放在万维网服务器中,它的内容不会根据浏览器发来的数据而改变。

(2) 动态 HTML 文档。动态 HTML 文档在浏览器访问服务器时才得以创建。当浏览器的请求到达时,服务器将 URL 映射到一个应用程序,由应用程序根据请求中的数据创建一个 HTML 文档。因此,每一个请求所得到的动态文档的内容也不一样。

(3) 活动 HTML 文档。活动 HTML 文档把创建文档的工作移到浏览器进行。服务器发回给浏览器的文档中包含脚本程序,浏览器执行脚本后,得到完整的活动 HTML 文档。虽然,活动 HTML 文档中的脚本程序可与用户直接交互,但活动文档一旦建立,它所包含的内容也就被固定了下来。

3.2.2　超文本传送协议概述

万维网的应用层协议是超文本传送协议(HTTP),它是万维网的核心。HTTP 有多个

版本，RFC1945 中定义了 HTTP/1.0，RFC7230～RFC7235 中定义了 HTTP/1.1，RFC7540 和 RFC7541 中定义了 HTTP/2。目前，HTTP/1.1 和 HTTP/2 是互联网建议标准。

目前互联网上应用广泛的 HTTP/1.1 最初由 RFC2068 定义，2014 年，IETF 更新了 HTTP/1.1，这是 HTTP/1.1 的一次重大更新。组织者将原来的 RFC2068 拆分为 6 个单独的 RFC 文档说明，并重点对原来语义模糊的部分进行了解释，新的文档说明更易懂、易读。新的 RFC 文档包括以下 6 部分：

(1) RFC7230(HTTP/1.1：message syntax and routing)。

(2) RFC7231(HTTP/1.1：semantics and content)。

(3) RFC7232(HTTP/1.1：conditional requests)。

(4) RFC7233(HTTP/1.1：range requests)。

(5) RFC7234(HTTP/1.1：caching)。

(6) RFC7235(HTTP/1.1：authentication)。

HTTP 由两个程序实现：一个客户程序(通常是浏览器)和一个服务器程序(通常是万维网服务器)。浏览器和万维网服务器一般运行在不同的端系统中，通过交换 HTTP 报文进行会话。HTTP 定义了这些报文的结构以及客户和服务器进行报文交换的方式。HTTP 的目的是实现浏览器从万维网服务器获取资源。这里的资源包括互联网上可以被访问的任何对象，例如文本、声音、图像等各种多媒体文件。

HTTP 的工作过程大致如图 3.6 所示。当用户在浏览器的地址栏中输入 URL 或者在某一个页面中单击一个超链接后，浏览器会自动在互联网上访问指定的资源。

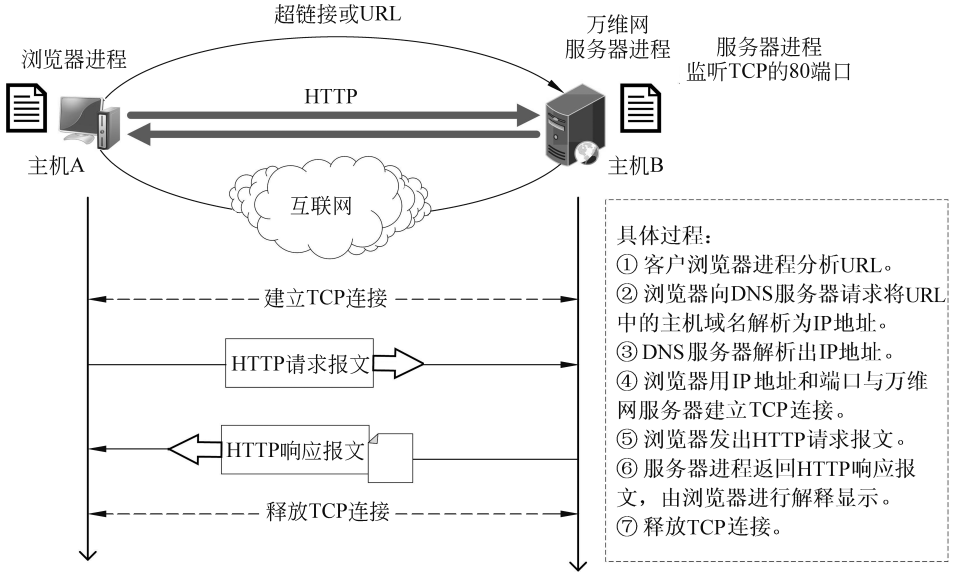


图 3.6　HTTP 工作过程

HTTP 使用面向连接的 TCP 作为传输层协议来保证数据的可靠传输。万维网服务器进程总是打开的，具有固定的 IP 地址，服务于大量来自不同浏览器的请求。每个万维网服务器进程不断地监听 TCP 的 80 端口，以便发现是否有浏览器向它发出 TCP 连接建立请求。当用户在浏览器的地址栏中输入 URL 或者在某一个页面中触发了超链接后，浏览器

首先解析 URL，如果 URL 中的主机使用的是域名而不是 IP 地址，则主机将域名发给 DNS 服务器，请求将域名解析为 IP 地址。关于 DNS，将在 3.3 节介绍。当 DNS 解析出 IP 地址后，浏览器利用该 IP 地址和 80 端口向万维网服务器发出 TCP 连接建立请求。在万维网服务器接受了连接建立请求并建立了 TCP 连接后，浏览器就可以向万维网服务器发出获取某个资源的请求，服务器便会以返回所请求的资源作为响应。最后，TCP 连接被释放。浏览器和万维网服务器之间的请求报文和响应报文的交互，必须遵循 HTTP 的规定。

HTTP 规定了在 HTTP 客户与 HTTP 服务器之间的每次交互，都由一个 ASCII 码串构成的请求报文和一个类似多用途互联网邮件扩展（multipurpose internet mail extensions，MIME），即“类 MIME（MIME-like）”的响应报文组成。MIME 最初是为了将纯文本格式的电子邮件扩展到可以支持多种信息格式而设计的，后来被应用到多种协议里。MIME 的常见形式是一个主类型加一个子类型，用“/”分隔。例如 text/html、application/JavaScript、image/jpg 等。在访问网页时，MIME 类型帮助浏览器识别一个 HTTP 响应报文中返回的是什么内容的数据，应该如何打开、如何显示。关于 MIME，会在 3.5 节介绍。

HTTP 属于无状态协议（stateless protocol）。也就是说，服务器不存储任何关于客户的状态信息，它既不记录曾经访问过的某客户，也不记录为某客户曾经服务过多少次。当同一个客户第二次访问同一个万维网服务器上的页面时，服务器的响应与第一次被访问时的相同。HTTP 的无状态特性简化了服务器的设计，使服务器更容易支持大量并发的 HTTP 请求。

在许多互联网应用中，客户和服务器会在一个相当长的时间范围内多次通信，其中客户发出一系列请求，而服务器对每个请求进行响应。如果该客户-服务器应用选用 TCP 作为传输层协议，则应用程序的设计者就需要做一个重要决定：每个请求/响应对需经过一个单独的 TCP 连接发送，还是一系列的请求及其响应都经过相同的 TCP 连接发送呢？前一种设计方法被称为非持续连接（non-persistent connection）；后一种设计方法被称为持续连接（persistent connection）。

为了深入地理解该设计问题，以万维网应用中 HTTP 的设计为例，研究持续连接的优点和缺点。HTTP/1.0 仅支持非持续连接，而 HTTP/1.1 既支持非持续连接，也支持持续连接。HTTP/1.1 在默认方式下使用持续连接，如果经过配置，也可以使用非持续连接。

当采用非持续连接时，浏览器每次向万维网服务器请求一个文件，都需要先建立 TCP 连接，然后经过一次 HTTP 请求和响应的交互后，才能获得文件。在非持续连接情况下，浏览器请求一个文件的时间估算如图 3.7 所示。

浏览器与服务器建立 TCP 连接需要使用三报文握手，关于 TCP 的三报文握手的细节，会在 4.5 节介绍。总之，三报文握手的前两部分所耗费的时间占用约一个 RTT。浏览器发送了三报文握手的最后一个报文段之后，紧跟着就可以发送 HTTP 请求，服务器收到 HTTP 请求报文后，就把所请求的文件封装到响应报文中返回给客户，该 HTTP 请求/响应占用了另一个 RTT 时间。因此，粗略地讲，总的响应时间就是两倍 RTT 加上服务器发送文件的延迟。文件发送完毕后，TCP 连接就被释放了。如果需要向服务器请求另一个文件，需要再次建立 TCP 连接。

采用非持续连接工作方式的主要缺点有两个。第一，为每次 HTTP 请求建立一个全新的 TCP 连接，需要服务器为该 TCP 连接分配缓存等资源，这给服务器带来了严重的负担。

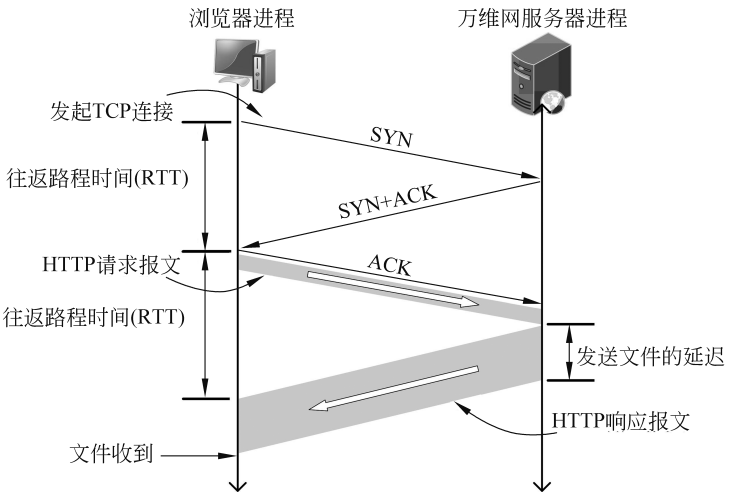


图 3.7　非持续连接情况下，获取一个文件所需时间

第二，每请求一个文件就要有两倍 RTT 的开销，效率很低。

持续连接的工作方式较好地解决了以上问题，万维网服务器在发送 HTTP 响应报文之后，仍然在一段时间内保持这条 TCP 连接，使同一个浏览器和该万维网服务器可以继续在这条 TCP 连接上传送后续的 HTTP 请求报文和响应报文。这并不局限于传送同一个页面上链接的文档，只要这些文档都在同一个服务器上就行。

HTTP/1.1 的持续连接有两种工作方式，即非流水线(without pipelining)方式和流水线(with pipelining)方式。在非流水线工作方式下，客户在收到前一个 HTTP 响应报文后才能发出下一个 HTTP 请求报文。因此，在 TCP 连接已建立后，客户每访问一次对象都要用去一个往返路程时间(RTT)，与非持续连接相比，节约了一个 RTT。但非流水线方式还是有缺点的，因为服务器在发送完一个对象后，其 TCP 连接就处于空闲状态，浪费了服务器资源。而如果采用流水线工作方式，客户在收到服务器发回的 HTTP 的响应报文之前就能够接着发送新的 HTTP 请求报文。于是一个接一个的请求报文到达服务器后，服务器就可连续发回响应报文。因此，使用流水线方式时，TCP 连接中的空闲时间减少，提高了万维网应用的效率。

HTTP/2 是在 HTTP/1.1 的基础上构建的，它允许在相同连接中交错发送多个 HTTP 请求报文和 HTTP 应答报文，并增加了在 TCP 连接中优化 HTTP 报文请求和应答的机制，因此效率更高。

3.2.3　HTTP 报文格式

HTTP 有两类报文：请求报文和响应报文。如图 3.8 所示，由于 HTTP 是面向文本的(text-oriented)，因此在报文中的每一个字段都是一些 ASCII 码串，因而各个字段的长度都是不确定的。

HTTP 请求报文和响应报文都是由 3 部分组成的。从图 3.8 中可以看出，这两种报文格式的区别就是开始行不同。

(1) 开始行。用于在请求报文中的开始行称为请求行(request-line)，而在响应报文中

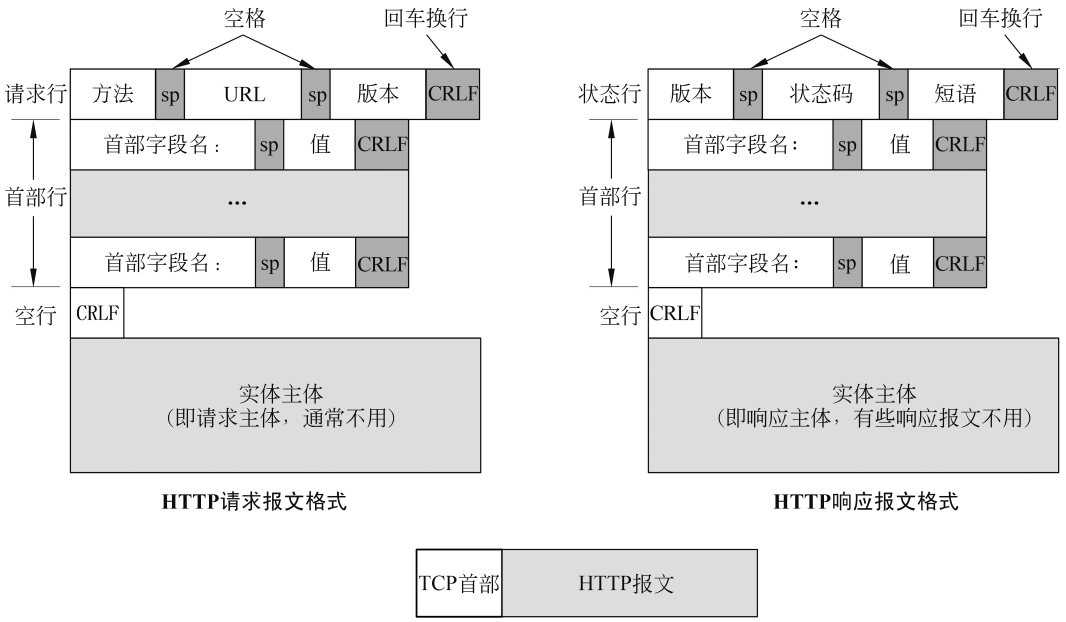


图 3.8　HTTP 报文的格式

的开始行称为状态行(status-line)。在开始行的 3 个字段之间都以空格分隔开,最后的“CR”和“LF”分别代表“回车”和“换行”。

(2) 首部行。用来说明浏览器、服务器或报文主体的一些信息。首部行可以包含多行,也可以不使用。每一个首部行中都包含首部字段名和它的值,每一个首部行都以“回车”和“换行”结束。所有首部行结束时,需要一个空行将首部行和后面的实体主体分开。在请求报文中的首部行也称为请求头,在响应报文中的首部行也称为响应头。

(3) 实体主体(entity body)。在请求报文中称为请求主体,HTTP 请求中一般不使用这个字段。在响应报文中称为响应主体,最常见的 HTTP 响应报文中包含该字段,但某些 HTTP 响应报文中没有这个字段。

1. HTTP 请求报文

如图 3.9 是用 Wireshark 截获的一段 HTTP 请求报文,下面以此为例进行说明。

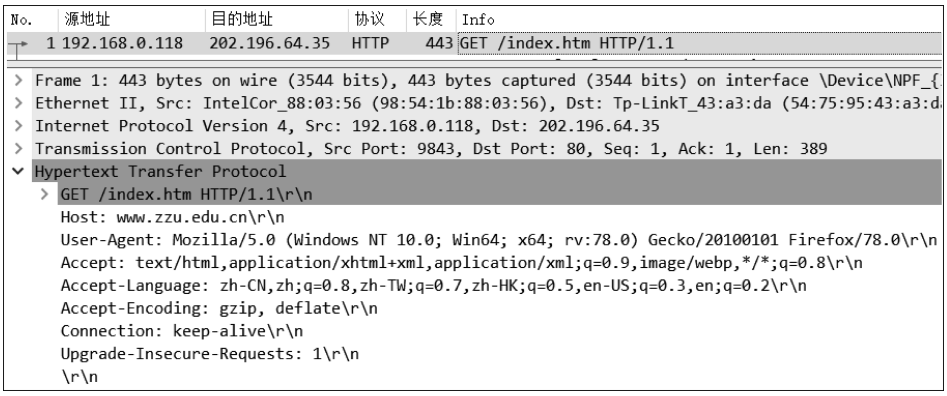


图 3.9　用 Wireshark 截获的 HTTP 请求报文

1）请求行

HTTP 请求报文的第一行是“请求行”，包含方法、URL 以及 HTTP 的版本号 3 项内容。

HTTP 不仅支持获取一个资源，也支持其他操作。在 HTTP 中，将操作称为方法（method）。所谓方法，是面向对象中的术语，在 HTTP 中代表一些命令。常见的方法有 GET、POST 等。表 3.1 列出了请求报文中的几种方法。方法名区分大小写。

表 3.1　HTTP 请求报文中的方法

方法（操作）	意　义
GET	获取 URL 指定的资源
POST	用于提交信息或数据
HEAD	请求读取由 URL 所指定的资源的首部
PUT	向指定 URL 位置上传内容
DELETE	请求服务器删除 URL 指定的资源
TRACE	回显服务器收到的请求，用于测试或诊断
CONNECT	用于代理服务器
OPTION	请求服务器告知选项功能

HTTP/1.0 定义了 GET、POST 和 HEAD 这 3 种请求方法。HTTP/1.1 增加了 PUT、DELETE、TRACE、CONNECT 和 OPTIONS 这 5 种请求方法。其用法如下。

（1）GET 方法。该方法用于请求服务器发送 URL 指定的资源。GET 方式是最简单的也是最常用的方法，它不使用请求主体。使用 GET 方法时，允许将请求参数和对应的值附加在 URL 后面，利用一个“?”将 URL 和请求参数分隔开，多个参数之间使用“&”隔开。因为 URL 长度受限，所以利用 GET 方法提交数据时，所有参数的总长度也受到了限制。

（2）POST 方法。该方法用于向指定 URL 提交信息或数据，请求服务器进行处理请求（例如提交表单或者上传文件），待提交数据被包含在请求主体中。相对于利用 GET 方法提交数据，POST 方法提交的数据不受 URL 长度限制。

（3）HEAD 方法。该方法与 GET 类似，但仅请求消息头，而不获取真正的资源。使用 HEAD 方法，可以在不获取资源的情况下了解资源的情况。例如，下载文件前使用 HEAD 发送请求，通过响应头中的 ContentLength 字段了解网络资源的大小；或者通过响应头中的 LastModified 字段判断本地缓存资源是否要更新。

（4）PUT 方法。与 GET 方法相反，该方法不用于读取资源，而是用于请求服务器将请求主体的内容写入 URL 指定的位置。如果 URL 指定的资源不存在，则新建该资源；如果 URL 指定的资源已经存在，就用请求主体的内容覆盖它。因为 PUT 方法允许用户对内容进行修改，所以很多万维网服务器要求执行 PUT 方法之前进行身份认证。

（5）TRACE 方法。该方法允许客户发起一次回送测试，要求服务器在响应主体中携带它收到的原始请求报文，便于客户软件查看和比较。

（6）DELETE 方法。该方法用于请求服务器删除 URL 所指定的资源。使用 DELETE

方法，客户进程无法确保删除操作一定会被执行，因为 HTTP 允许服务器在不通知客户进程的情况下撤销该请求。

(7) CONNECT 方法。该方法使客户进程能够通过一个中间设备(比如代理服务器)与服务器建立连接。

(8) OPTIONS 方法。该方法用于请求服务器告知其支持的哪些其他的功能和方法，或者针对某些特定的资源支持哪些特定的操作。

在图 3.9 的例子中，请求行内容如下：

```
GET  /index.htm  HTTP/1.1\r\n
```

该请求行的 URL 中省略了主机的域名，这是因为下面首部行的 host 字段告知了所要访问的主机的域名。该请求行含义为请求服务器发送/index.htm 文件。

2) 首部行

HTTP 请求中，常用的首部行字段含义如下。

(1) Host 字段。该字段用于指定请求资源的主机名和端口号，其中端口号可选，默认为 80。在图 3.9 的例子中，Host 字段值为 www.zzu.edu.cn。

(2) User-Agent 字段。该字段指明了用户使用的代理软件(包括用户的操作系统、浏览器等)的相关属性。在图 3.9 的例子中，User-Agent 字段指明用户操作系统为 WindowsNT 10，浏览器为 Firefox。

(3) Accept 字段。该字段用于告知服务器客户进程能够处理的媒体类型及其相对优先级顺序，其中媒体类型用 MIME 表示。如果一次性指定多种媒体类型，可以采用权重值 q 表示相对优先级。本字段可使用“*”作为通配符，指定任意媒体类型。在图 3.9 所示的例子中，客户进程接受 text/html 等类型的数据。

(4) Accept-Encoding 字段。该字段用于告知服务器客户进程支持的内容编码及内容编码的优先级顺序。如果一次性指定多种内容编码，可以采用权重值 q 表示相对优先级。本字段可使用“*”作为通配符，指定任意编码格式。在图 3.9 所示的例子中，客户进程接受 gzip 等编码格式。

(5) Accept-Language 字段。该字段用于告知服务器所用的客户进程能够处理的自然语言集。在图 3.9 的例子中，客户进程能够处理中文等。

(6) Accept-Charset 字段。该字段用于告知服务器所用的客户进程能够处理的字符集。在图 3.9 的例子中，未包含该字段。

(7) Connection 字段。该字段用于告知服务器是否使用持续连接。在图 3.9 的例子中，客户进程使用持续连接。

(8) Cookie 字段。该字段用于表示请求者身份。在图 3.9 的例子中，未包含该字段。详细内容会在 3.2.4 节介绍。

(9) If-Modified-Since 和 If-Match 字段。这两个字段都可以用来构造条件 GET 请求。条件 GET 请求用来询问服务器本地缓存的副本是否仍然有效。浏览器在获得 HTTP 响应时，可以将获得的资源缓存起来，并记录响应报文中 Last-Modified 和 Etag 字段的值并在浏览器再次请求相同 URL 时，将 Last-Modified 值写入 If-Modified-Since 字段或者将 Etag 值写入 If-Match 字段，询问服务器所请求的资源是否发生过改变。如果相应的资源未被修

改，则万维网服务器返回一个状态码为 304、实体为空的响应报文。浏览器收到 304 响应后，可以直接应用缓存的资源。

3）请求实体

与其他方法不同，在 HTTP 请求中只有 POST 和 PUT 方法才需要包含请求实体。在图 3.9 的例子中使用的是 GET 方法，未包含请求实体。

2. HTTP 响应报文

万维网服务器收到 HTTP 请求报文后，应该发回一个 HTTP 响应报文。

万维网服务器对如图 3.9 所示 HTTP 请求的响应报文会被分为多个 TCP 报文段发送，由 Wireshark 软件重组后的 HTTP 响应报文如图 3.10 所示。

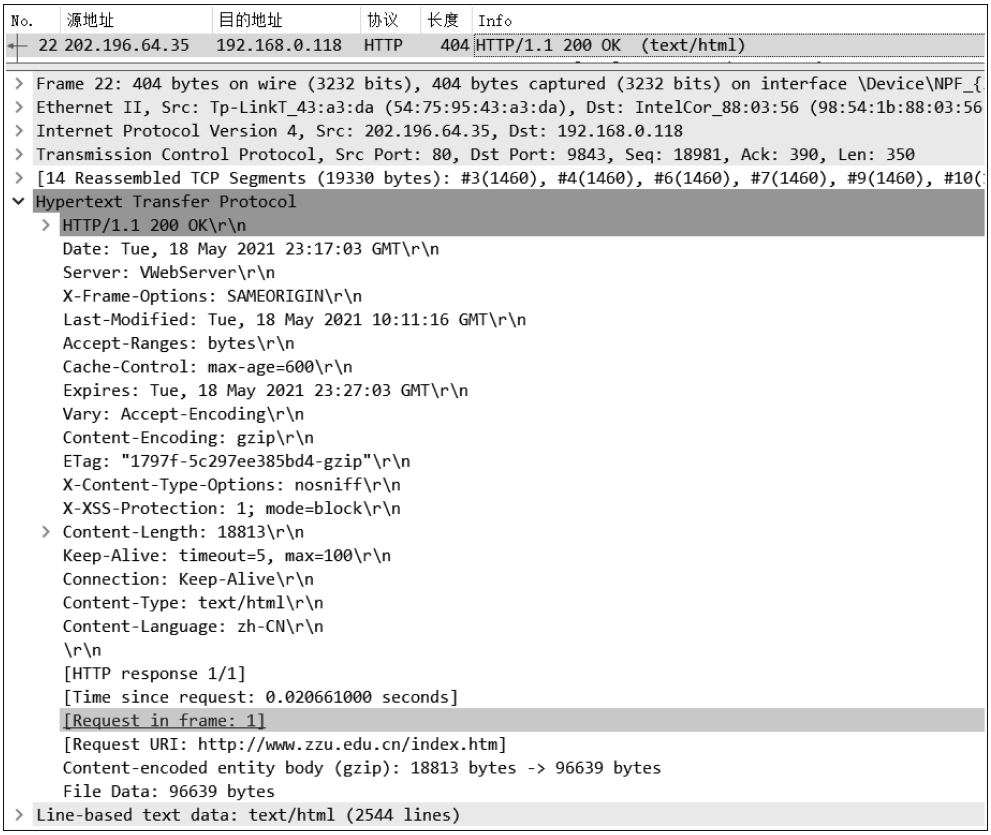


图 3.10　HTTP 响应报文的例子

1）状态行

HTTP 响应报文的第一行是状态行，它包括 3 项内容：HTTP 的版本、状态码（status-code）以及解释状态码的简单短语。

HTTP 的状态码都是 3 位数字形式，第一个数字把状态码分为 5 类，如表 3.2 所示。1xx 表示信息提示，例如请求已被服务器接受，但需要继续处理。实际上很少被使用。2xx 表示请求成功，例如处理成功，并且返回了相应的内容。3xx 表示重定向，例如访问的资源已被移动。重定向响应中包含新资源的参考 URL，如果需要完成请求，客户端收到重定向会重新对新资源发起请求。4xx 表示客户端的错误而导致请求失败，例如资源不存在或者

请求中有错误的语法等。5xx 表示服务器自身错误导致服务无法完成，例如服务器程序异常或者临时负载过重等。

表 3.2　HTTP 响应报文中的状态码

状　态　码	含　　义	举　　例	解　　释
1xx	信息提示	100	表示服务器同意处理客户请求
2xx	请求成功	200	表示请求成功
		204	表示请求成功，无须内容
3xx	重定向	301	表示资源被永久性移动
		304	表示缓存的资源依然有效
4xx	客户端错误	403	表示禁止访问
		404	表示资源没有找到
5xx	服务器端错误	500	表示服务器内部错误
		503	表示服务器临时过载

例如，图 3.10 中的状态行

```
HTTP/1.1  200  OK
```

表示之前发送的 HTTP 请求报文已被服务器成功处理。

2）首部行

HTTP 响应中，常用的首部行字段含义如下。

（1）Server 字段。该字段用于服务器所使用的 Web 服务器名称。攻击者可以通过查看该信息，来探测 Web 服务器名称。所以一般服务器端会对该信息进行修改。图 3.10 所示例子中，该字段值为 VWebServer。

（2）Last-Modified 字段。该字段用于指定被请求资源最后被修改的日期和时间。图 3.10所示例子中，被请求资源的最后修改日期是 18 May 2021。

（3）Etag 字段。该字段是一个与资源相关联的唯一标识。服务器为发送的每个资源分派一个唯一的字符串形式的标识符。浏览器将资源缓存并与 Etag 建立关联，当再次发起请求时，可以使用 Etag 作为标识，向服务器发送条件 GET 请求。若资源未改变，则服务器返回不包含响应实体的 304 响应即可。图 3.10 所示例子中，Etag 字段的值为 1797f-5c297ee385bd4-gzip。

（4）Location 字段。对于一个已经移动的资源，Location 字段用于指向新的资源位置。该字段通常与状态码 302（暂时移动）或者 301（永久性移动）配合使用。图 3.10 所示例子中，状态码为 200，未包含该字段。

（5）Cache-Control 字段。该字段用于指定客户端对资源的缓存策略。例如指定是否进行缓存、缓存资源的有效期等。图 3.10 所示例子中，Cache-Control 字段指定 max-age＝600，代表该资源缓存的有效期为 600s。

（6）X-Frame-Options 字段。该字段用于指明本页面的内容是否允许嵌套在其他万维网网站的 frame 标签内显示。该字段值为 DENY，代表不允许在任何 frame 标签内显示；值

为 SAMEORIGIN，代表允许在相同域名网站的 frame 标签内显示；值为 ALLOW-FROM url，代表仅允许在指定 URL 的 frame 标签内显示。该字段的主要目的是为了防止点击劫持（click jacking）攻击。图 3.10 所示例子中，X-Frame-Options 字段的值为 SAMEORIGIN，这是最常见的一种设置。

（7）X-XSS-Protection 字段。该字段用于指明针对跨站脚本攻击（XSS）的响应策略，是用于控制浏览器 XSS 防护机制的开关。该字段值为“0”代表禁用 XSS 保护，值为“1”代表启用 XSS 保护。图 3.10 所示例子中，X-XSS-Protection 字段的值为“1；mode＝block”，代表启用 XSS 保护，并在检查到 XSS 攻击时，停止渲染页面。

（8）Content-Type 字段。该字段用于指明响应实体的内容类型。图 3.10 所示例子中，Content-Type 字段的值为 text/html，说明返回的响应实体为 HTML 格式的文本文件。

（9）Content-Encoding 字段。该字段用于指明响应实体的编码格式。图 3.10 所示例子中，Content-Encoding 字段的值为 gzip。

（10）Content-Length 字段。该字段用于指明响应实体的内容长度。图 3.10 所示例子中，Content-Length 字段的值为 18813。

（11）Set-Cookie 字段。该字段用于向客户端设置 Cookie，与 Cookie 请求头相互对应。

3）响应实体

在 HTTP 响应中，最常见的是 200 响应，服务器返回给浏览器的资源就放在 200 响应的响应实体中。在图 3.10 所示的例子中，200 响应的响应实体内容是一个 2544 行的 HTML 文件。

3. Cookie

HTTP 是一种无状态协议，每次请求之间是相互独立的，当前请求不会记录它的上一次请求信息，这样设计的好处是可以简化服务器的设计。但是对于一套完整的业务逻辑，需要多次发送请求的情况数不胜数，使用 HTTP 如何关联上下文请求呢？Cookie 就是这样的一种机制，它可以弥补 HTTP 无状态的不足，利用 Cookie 可以实现跟踪会话的功能。RFC6265 中规定了利用 Cookie 跟踪用户状态的方法，目前 RFC6265 是互联网建议标准。

Cookie 包含 4 个组件。

（1）在 HTTP 响应报文中的 Set-Cookie 首部行。

（2）在 HTTP 请求报文中的 Cookie 首部行。

（3）在用户系统中保存的 Cookie。该 Cookie 可以保存在内存或磁盘中，由用户浏览器进行管理。

（4）位于万维网服务器的后端数据库。

目前主流的浏览器都支持 Cookie，除非用户禁用了 Cookie 功能。

Cookie 实际上是一小段的文本信息。浏览器向万维网服务器发送 HTTP 请求后，如果服务器需要记录该用户状态，就在 HTTP 响应中用 Set-Cookie 首部行发送一个 Cookie 给浏览器。用户的浏览器会把 Cookie 保存起来。当浏览器再次向该万维网服务器发送 HTTP 请求时，就在 HTTP 请求中用 Cookie 首部行将该 Cookie 信息发送给服务器。万维网服务器利用后端数据库检查该 Cookie，以此来辨别用户状态。

图 3.11 所示为一个利用 Cookie 跟踪用户状态的例子。该例中，服务器发送给浏览器的响应报文中的 Set-Cookie 首部行如下：

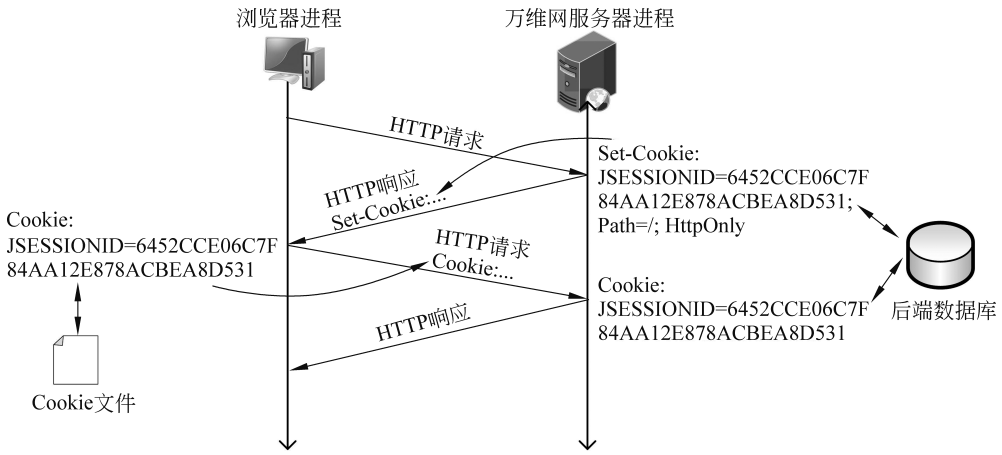


图 3.11　利用 Cookie 跟踪用户状态的例子

```
Set-Cookie: JSESSIONID=6452CCE06C7F84AA12E878ACBEA8D531; path=/; HttpOnly
```

收到 Cookie 后，浏览器再次请求时发送的请求报文中的 Cookie 首部行如下：

```
Cookie: JSESSIONID=6452CCE06C7F84AA12E878ACBEA8D531
```

RFC6265 中规定，一个 Cookie 可以包含至多 7 个字段。

(1) 内容字段。由一个或多个键值对组成，内容字段是 Cookie 的主体。在图 3.11 所示的例子中，内容字段为 JSESSIONID＝6452CCE06C7F84AA12E878ACBEA8D531。

(2) 过期时间(Expire)字段。指定了该 Cookie 何时过期。如果这个字段不存在，则浏览器在退出时将丢弃该 Cookie，这样的 Cookie 称为非持续 Cookie，也称为临时 Cookie；如果针对某个 Cookie 提供了时间和日期，那么这样的 Cookie 称为持续 Cookie，它被一直保存直至过期为止。通常临时 Cookie 保存在内存中，而持续 Cookie 保存在磁盘中。

(3) 最大生存期(Max-Age)字段。新版本的生存期字段，以秒为单位指定 Cookie 的过期时间。如果 Max-Age 为正数，则 Cookie 为持续 Cookie；如果 Max-Age 为负数，则 Cookie 为临时 Cookie；如果 Max-Age 为 0，则代表服务器要求删除 Cookie。

(4) 域(Domain)字段。该字段用于指出 Cookie 的来源。临时 Cookie 不能带有 Domain 字段。图 3.11 所示的例子中，Cookie 不包含 Expire 字段和 Max-Age 字段，属于临时 Cookie，因此也不包含 Domain 字段。

(5) 路径(Path)字段。该字段用于指明允许访问此 Cookie 的文档路径。例如 Path 是“/test”，那么只有“/test”路径下的文档可以读取此 Cookie。图 3.11 所示的例子中，Path 是“/”，代表本网站所有文档都可以读取此 Cookie。

(6) 安全(Secure)字段。该字段是个标志字段，如果 Cookie 中包含 Secure，则该 Cookie 仅能用于安全连接，安全连接由浏览器规定，通常指 HTTPS 连接。

(7) 仅 HTTP(HttpOnly)字段。该字段也是个标志字段，如果 Cookie 中包含 HttpOnly，则浏览器会忽略那些通过“非 HTTP”方式对 Cookie 的访问，例如浏览器开放给 JavaScript 脚本的接口将不能读取 Cookie。图 3.11 所示例子中包含了 HttpOnly，因此该 Cookie 不能被客户端脚本获取到。

利用 Cookie，万维网服务器就能够根据 Cookie 的内容（如图 3.11 中的 JSESSIONID）跟踪用户在该网站的活动，记录用户的状态。虽然 Cookie 能够简化用户浏览网站的过程，但 Cookie 的使用一直引起很多争议。这些争议主要集中在用户隐私的保护方面。为了使用户具有拒绝接受 Cookie 的自由，目前主流的浏览器都支持用户自主设置接受 Cookie 的条件。

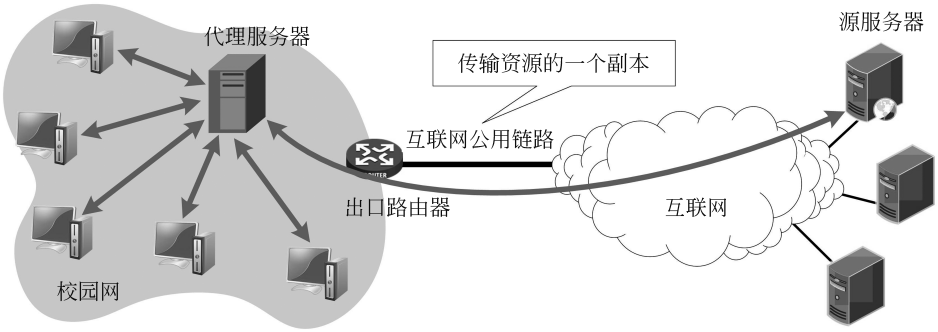
3.2.4 代理服务器和内容分发网络

代理服务器（proxy server）和内容分发网络（content delivery network，CDN）都是提高万维网访问效率的技术。

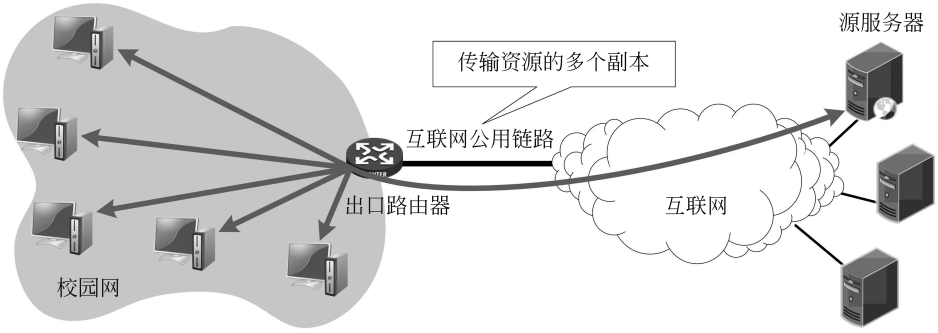
1. 代理服务器

代理服务器也称为万维网缓存（Web cache）。代理服务器把最近请求过的资源的副本保存在自己的存储空间，当收到新请求时，如果代理服务器发现新请求的资源与缓存的资源相同，就返回缓存的资源，而不需要根据 URL 再次访问该资源。代理服务器可在客户端或服务器端工作，也可在中间系统上工作。下面，以一个校园网中的代理服务器为例说明它的作用。

假设校园网中的某用户使用浏览器请求视频资源 http://www.abc.cn/xyz.mp4，图 3.12(a)所示为校园网用户使用代理服务器访问互联网的情况。过程如下。



（a）使用代理服务器



（b）未使用代理服务器

图 3.12　代理服务器的例子

（1）浏览器首先与代理服务器建立 TCP 连接，并以上述 URL 为参数，向代理服务器发送 HTTP 请求。