

(续)

头字段类型	HTTP 版本		含 义
	1.0	1.1	
WWW-Authenticate	○	○	当请求的信息存在访问控制时，返回身份认证用的数据 (Challenge ^①)
Accept-Ranges		○	当希望仅请求部分数据 (使用 Range 来指定范围) 时，服务器会告知客户端是否支持这一功能
实体头：用于表示实体 (消息体) 的附加信息的头字段			
Allow	○	○	表示指定的 URI 支持的方法
Content-Encoding	○	○	当消息体经过压缩等编码处理时，表示其编码格式
Content-Length	○	○	表示消息体的长度
Content-Type	○	○	表示消息体的数据类型，以 MIME 规格定义的数据类型来表示
Expires	○	○	表示消息体的有效期
Last-Modified	○	○	数据的最后更新日期
Content-Language		○	表示消息体的语言。汉语为 zh，英语为 en
Content-Location		○	表示消息体在服务器上的位置 (URI)
Content-Range		○	当仅请求部分数据时，表示消息体包含的数据范围
Etag		○	在更新操作中，有时候需要基于上一次请求的响应数据来发送下一次请求。在这种情况下，这个字段可以用来提供上次响应与下次请求之间的关联信息。上次响应中，服务器会通过 Etag 向客户端发送一个唯一标识，在下次请求中客户端可以通过 If-Match、If-None-Match、If-Range 字段将这个标识告知服务器，这样服务器就知道该请求和上次的响应是相关的。这个字段的功能和 Cookie 是相同的，但 Cookie 是网景 (Netscape) 公司自行开发的规格，而 Etag 是将其进行标准化后的规格

○：在规格中定义的项目。
△：并非正式规格，而是在规格书附录 (Appendix) 中定义的附加功能。

① 这里的 Challenge 指的是 Challenge-Response 身份验证模型中的一环。简单来说，Challenge 相当于“天王盖地虎”，Response 相当于“宝塔镇河妖”。——译者注

当使用 POST 方法时，需要将表单中填写的信息写在消息体中。到此为止，请求消息的生成操作就全部完成了。

1.1.6 发送请求后会收到响应

当我们将上述请求消息发送出去之后，Web 服务器会返回响应消息。关于响应消息我们将在第 6 章详细介绍，这里先粗略地了解一下。响应消息的格式以及基本思路 and 请求消息是相同的（图 1.5 (b)），差别只在第一行上。在响应消息中，第一行的内容为状态码和响应短语，用来表示请求的执行结果是成功还是出错。状态码和响应短语表示的内容一致，但它们的用途不同。状态码是一个数字，它主要用来向程序告知执行的结果（表 1.3）；相对地，响应短语则是一段文字，用来向人们告知执行的结果。

表 1.3 HTTP 状态码概要

状态码的第一位数字表示状态类型，第二、三位数字表示具体的情况。下表列举了第一位数字的含义。

状态码	含 义
1xx	告知请求的处理进度和情况
2xx	成功
3xx	表示需要进一步操作
4xx	客户端错误
5xx	服务器错误

返回响应消息之后，浏览器会将数据提取出来并显示在屏幕上，我们就能够看到网页的样子了。如果网页的内容只有文字，那么到这里就全部处理完毕了，但如果网页中还包括图片等资源，则还有下文。

当网页中包含图片时，会在网页中的相应位置嵌入表示图片文件的标签^①的控制信息。浏览器会在显示文字时搜索相应的标签，当遇到图片相关


① 标签：编写网页所使用的 HTML 语言中规定的控制信息。例如，当需要在网页中插入图片时，需要在相应位置嵌入形如 `` 的标签。

的标签时，会在屏幕上留出用来显示图片的空间，然后再次访问 Web 服务器，按照标签中指定的文件名向 Web 服务器请求获取相应的图片并显示在预留的空间中。这个步骤和获取网页文件时一样，只要在 URI 部分写上图片的文件名并生成和发送请求消息就可以了。

由于每条请求消息中只能写 1 个 URI，所以每次只能获取 1 个文件，如果需要获取多个文件，必须对每个文件单独发送 1 条请求。比如 1 个网页中包含 3 张图片，那么获取网页加上获取图片，一共需要向 Web 服务器发送 4 条请求。

判断所需的文件，然后获取这些文件并显示在屏幕上，这一系列工作的整体指挥也是浏览器的任务之一，而 Web 服务器却毫不知情。Web 服务器完全不关心这 4 条请求获取的文件到底是 1 个网页上的还是不同网页上的，它的任务就是对每一条单独的请求返回 1 条响应而已。

到这里，我们已经介绍了浏览器与 Web 服务器进行交互的整个过程。作为参考，图 1.7 展示了浏览器与 Web 服务器之间交互消息的一个实例。在这个例子中，我们需要获取一张名为 `sample1.htm` 的网页，网页中包含一张名为 `picture.jpg` 的图片，图中展示了这个过程中产生的消息。



1 条请求消息中只能写 1 个 URI。如果需要获取多个文件，必须对每个文件单独发送 1 条请求。

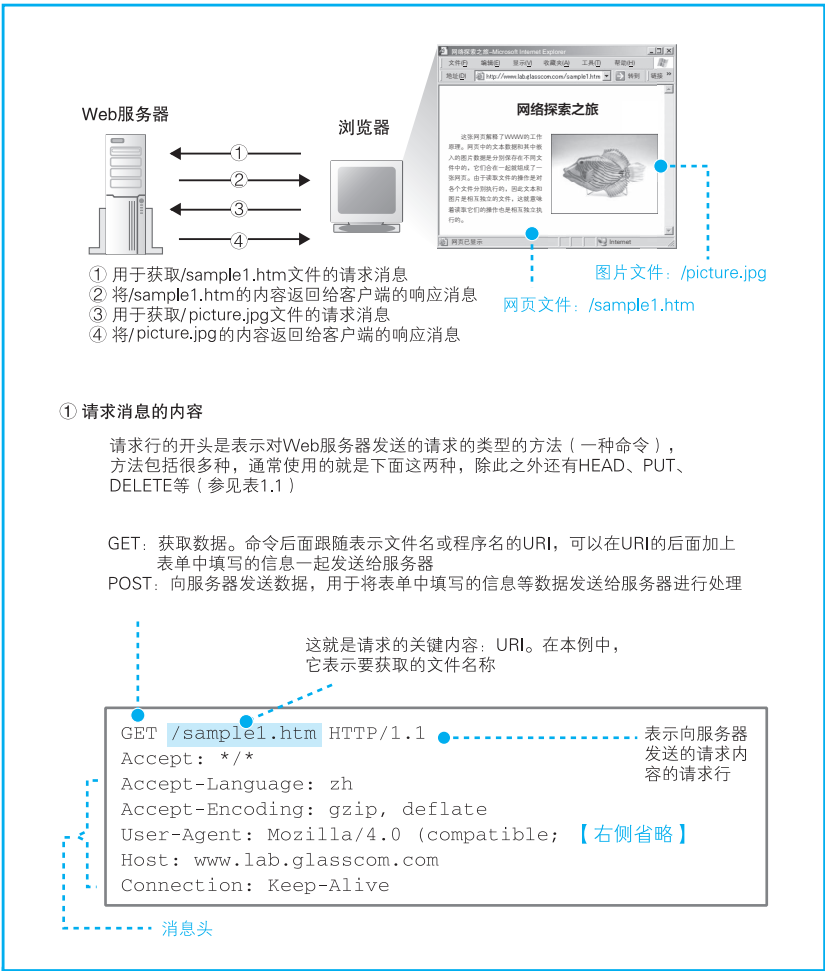


图 1.7 HTTP 消息示例

② 将/sample1.htm的内容返回给客户端的响应消息

服务器程序类型

状态行。“200 OK”表示请求成功完成

```

HTTP/1.1 200 OK
Date: Wed, 21 Feb 2007 09:19:14 GMT
Server: Apache
Last-Modified: Mon, 19 Feb 2007 12:24:51 GMT
ETag: "5a9da-279-3c726b61"
Accept-Ranges: bytes
Content-Length: 632
Connection: close
Content-Type: text/html

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>网络探索之旅</title>
</head>

<body>
<h1 align="center">网络探索之旅</h1>


</body>
</html>

```

以MIME规格表示的数据格式。text/html表示HTML文档。如果是JPEG格式的图片，这里应该是image/jpeg

这是嵌入的图片文件的名字。在接下来的请求中向服务器获取这个文件（下页❶）

图 1.7 （续）

③ 用于获取/picture.jpg文件的请求消息

```
GET /picture.jpg HTTP/1.1
Accept: */*
Referer: http://www.lab.glasscom.com/sample1.htm
Accept-Language: zh
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; 【右侧省略】
Host: www.lab.glasscom.com
Connection: Keep-Alive
```

④ 将/picture.jpg的内容返回给客户端的响应消息

```
HTTP/1.1 200 OK
Date: Wed, 21 Feb 2007 09:19:14 GMT
Server: Apache
Last-Modified: Mon, 19 Feb 2007 13:50:32 GMT
ETag: "5a9d1-1913-3aefa236"
Accept-Ranges: bytes
Content-Length: 6419
Connection: close
Content-Type: image/jpeg
```

image/jpeg表示JPEG格式的图
片数据

【下面就是图片数据，因为这些数据都是二进制的，所以我们在此省略】

图 1.7（续）

1.2 向 DNS 服务器查询 Web 服务器的 IP 地址

1.2.1 IP 地址的基本知识

生成 HTTP 消息之后，接下来我们需要委托操作系统将消息发送给 Web 服务器。尽管浏览器能够解析网址并生成 HTTP 消息，但它本身并不具备将消息发送到网络中的功能，因此这一功能需要委托操作系统来实现^①。在进行这一操作时，我们还有一个工作需要完成，那就是查询网址中

① 发送消息的功能对于所有的应用程序来说都是通用的，因此让操作系统来实现这一功能，其他应用程序委托操作系统来进行操作，这是一个比较合理的做法。

服务器域名对应的 IP 地址。在委托操作系统发送消息时，必须要提供的不是通信对象的域名，而是它的 IP 地址。因此，在生成 HTTP 消息之后，下一个步骤就是根据域名查询 IP 地址。在讲解这一操作之前，让我们先来简单了解一下 IP 地址。

互联网和公司内部的局域网都是基于 TCP/IP 的思路来设计的，所以我们先来了解 TCP/IP 的基本思路。TCP/IP 的结构如图 1.8 所示，就是由一些小的子网，通过路由器^①连接起来组成一个大的网络。这里的子网可以理解为用集线器^②连接起来的几台计算机^③，我们将它看作一个单位，称为子网。将子网通过路由器连接起来，就形成了一个网络^④。

在网络中，所有的设备都会被分配一个地址。这个地址就相当于现实中某条路上的“××号××室”。其中“号”对应的号码是分配给整个子网的，而“室”对应的号码是分配给子网中的计算机的，这就是网络中的地址。“号”对应的号码称为网络号，“室”对应的号码称为主机号，这个地址的整体称为 IP 地址^⑤。通过 IP 地址我们可以判断出访问对象服务器的位置，从而将消息发送到服务器。消息传送的具体过程在后面的章节有详细讲解，不过现在我们先简单了解一下。发送者发出的消息首先经过子网

① 路由器：一种对包进行转发的设备，在第 3 章有详细介绍。

② 集线器：一种对包进行转发的设备，分为中继式集线器和交换式集线器两种，在第 3 章有详细介绍。

③ 当计算机数量较少时，可以用一台集线器连接起来；当计算机数量较多时，一台集线器可能无法连接这么多计算机，可以增加集线器数量并将集线器相互连接起来，这时，凡是通过集线器连接起来的所有设备都属于同一个子网。

④ 一些家用路由器中已经内置了集线器功能，因此大家可以理解为这种路由器内部同时包含路由器和集线器两种设备，它们在里面已经连接起来了。

⑤ IP 地址和现实中的地址含义是相同的，因此就像“××号××室”不能有两户人家的号码相同一样，也不能有两台设备使用相同的 IP 地址。现实中其实存在因为疏漏两台设备被分配了相同的 IP 地址的情况，但这种情况下网络会发生故障，无法正常工作。

中的集线器^①，转发到距离发送者最近的路由器上(图 1.8 ①)。接下来，路由器会根据消息的目的地判断下一个路由器的位置，然后将消息发送到下一个路由器，即消息再次经过子网内的集线器被转发到下一个路由器(图 1.8 ②)。前面的过程不断重复，最终消息就被传送到了目的地。

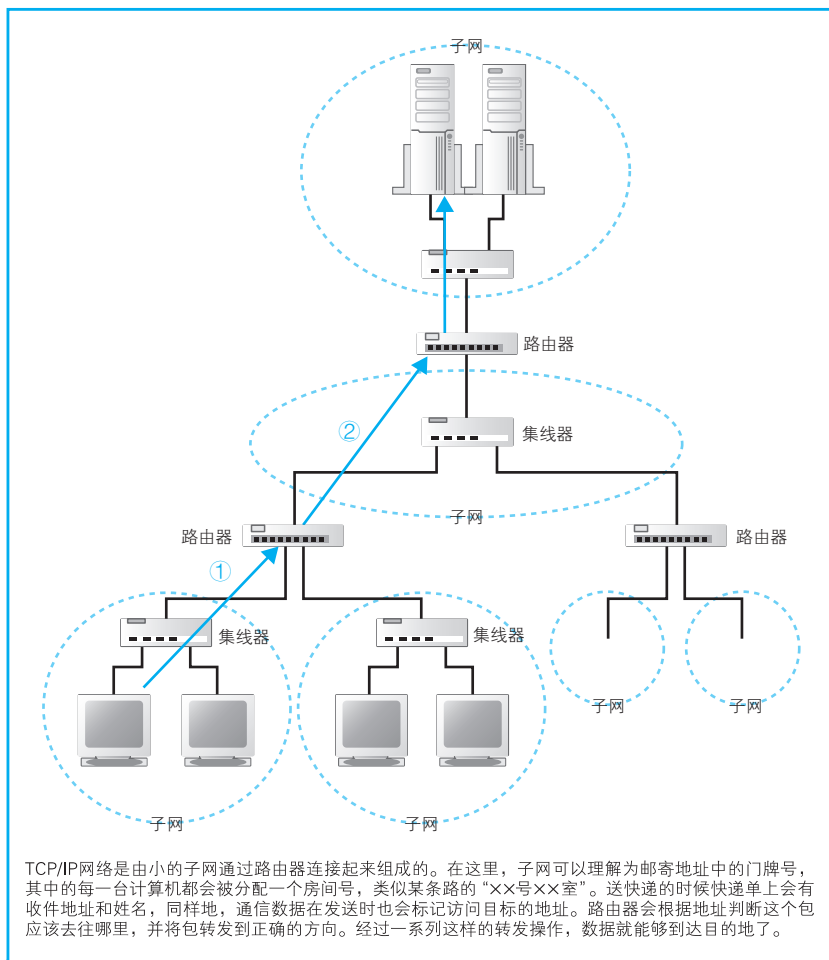


图 1.8 IP 的基本思路

① 数据是以包的形式传送的。

前面这些就是 TCP/IP 中 IP 地址的基本思路。了解了这些知识之后，让我们再来看一下实际的 IP 地址。如图 1.9 所示，实际的 IP 地址是一串 32 比特的数字，按照 8 比特（1 字节）为一组分成 4 组，分别用十进制表示然后再用圆点隔开。这就是我们平常经常见到的 IP 地址格式，但仅凭这一串数字我们无法区分哪部分是网络号，哪部分是主机号。在 IP 地址的规则中，网络号和主机号连起来总共是 32 比特，但这两部分的具体结构是不固定的。在组建网络时，用户可以自行决定它们之间的分配关系，因此，我们还需要另外的附加信息来表示 IP 地址的内部结构。

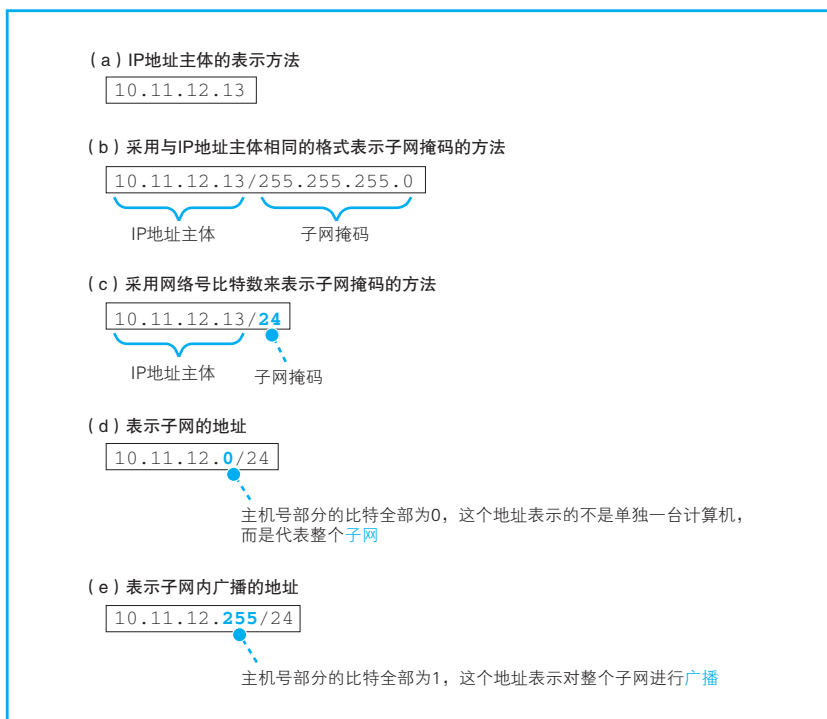


图 1.9 IP 地址的表示方法

这一附加信息称为子网掩码。子网掩码的格式如图 1.10 ②所示，是一串与 IP 地址长度相同的 32 比特数字，其左边一半都是 1，右边一半都是

是

0。其中，子网掩码为 1 的部分表示网络号，子网掩码为 0 的部分表示主机号。将子网掩码按照和 IP 地址一样的方式以每 8 比特为单位用圆点分组后写在 IP 地址的右侧，这就是图 1.9 (b) 的方法。这种写法太长，我们也可以把 1 的部分的比特数用十进制表示并写在 IP 地址的右侧，如图 1.9 (c) 所示。这两种方式只是写法上的区别，含义是完全一样的。

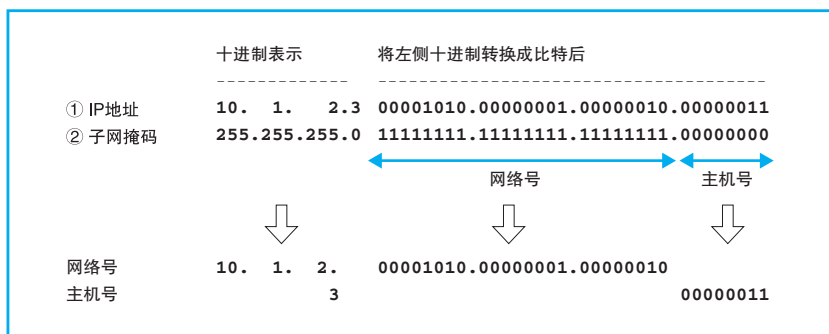


图 1.10 IP 地址的结构

子网掩码表示网络号与主机号之间的边界。在本例中，这个边界与字节的边界是正好吻合的，也就是正好划分在句点的位置上，实际上也可以划分在字节的中间位置。

顺带一提，主机号部分的比特全部为 0 或者全部为 1 时代表两种特殊的含义。主机号部分全部为 0 代表整个子网而不是子网中的某台设备（图 1.9 (d)）。此外，主机号部分全部为 1 代表向子网上所有设备发送包，即广播（图 1.9 (e)）。

IP 地址的主机号

全 0：表示整个子网

全 1：表示向子网上所有设备发送包，即“广播”

1.2.2 域名和 IP 地址并用的理由

TCP/IP 网络是通过 IP 地址来确定通信对象的，因此不知道 IP 地址就

无法将消息发送给对方，这和我们打电话的时候必须要知道对方的电话号码是一个道理。因此，在委托操作系统发送消息时，必须要先查询好对方的 IP 地址。

可能会问“既然如此，那么在网址中不写服务器的名字，直接写 IP 地址不就好了吗？”实际上，如果用 IP 地址来代替服务器名称也是能够正常工作的^①。然而，就像你很难记住电话号码一样，要记住一串由数字组成的 IP 地址也非常困难。因此，相比 IP 地址来说，网址中还是使用服务器名称比较好^②。

那么又有人问了：“既然如此，那干脆不要用 IP 地址，而是用名称来确定通信对象不就好了吗？互联网中使用的是最新的网络技术，和电话那种老古董可不一样，这样的功能应该还是做得到的吧？”这样的想法其实并不奇怪^③。

不过从运行效率上来看，这并不能算是一个好主意。互联网中存在无数的路由器，它们之间相互配合，根据 IP 地址来判断应该把数据传送到什么地方。那么如果我们不用 IP 地址而是改用名称会怎么样呢？IP 地址的长度为 32 比特，也就是 4 字节，相对地，域名最短也要几十个字节，最长甚至可以达到 255 字节。换句话说，使用 IP 地址只需要处理 4 字节的数字，而域名则需要处理几十个到 255 个字节的字符，这增加了路由器的负担，传送数据也会花费更长的时间^④。可能有人会说：“那使用高性能路由器不就能解决这个问题了吗？”然而，路由器的速度是有极限的，而互联网内部流动的数据量已然让路由器疲于应付了，因此我们不应该再采用效率更低的设计。随着技术的发展，路由器的性能也会不断提升，但与此同时，数据量也在以更快的速度增长，在可预见的未来，这样的趋势应该不会发生变化。出

① 如果 Web 服务器使用了虚拟主机功能，有可能无法通过 IP 地址来访问。

② 也有人说域名也很难记啊，不过在设计 TCP/IP 架构的当时，在技术上还无法实现我们今天的搜索引擎，因此用名称来代替地址本身是有价值的。

③ 实际上真的存在以名称来确定通信对象的网络，Windows 网络的原型 PC-Networks 就是其中的一个例子。

④ 域名并不仅是长，而且其长度是不固定的。处理长度不固定的数据比处理长度固定的数据要复杂，这也是造成效率低下的重要原因之一。

于这样的原因，使用名称本身来确定通信对象并不是一个聪明的设计。

于是，现在我们使用的方案是让人来使用名称，让路由器来使用 IP 地址。为了填补两者之间的障碍，需要有一个机制能够通过名称来查询 IP 地址，或者通过 IP 地址来查询名称，这样就能够在人和机器双方都不做出牺牲的前提下完美地解决问题。这个机制就是 DNS^①。

1.2.3 Socket 库提供查询 IP 地址的功能

查询 IP 地址的方法非常简单，只要询问最近的 DNS 服务器“www.lab.glasscom.com 的 IP 地址是什么”就可以了，DNS 服务器会回答说“该服务器的 IP 地址为 xxx.xxx.xxx.xxx”。这一步非常简单，很多读者也都很熟悉，那么浏览器是如何向 DNS 服务器发出查询的呢？让我们把向 Web 服务器发送请求消息的事情放一放，先来探索一下 DNS。

向 DNS 服务器发出查询，也就是向 DNS 服务器发送查询消息，并接收服务器返回的响应消息。换句话说，对于 DNS 服务器，我们的计算机上一定有相应的 DNS 客户端，而相当于 DNS 客户端的部分称为 DNS 解析器，或者简称解析器。通过 DNS 查询 IP 地址的操作称为域名解析，因此负责执行解析（resolution）这一操作的就叫解析器（resolver）了。

解析器实际上是一段程序，它包含在操作系统的 Socket 库中，在介绍解析器之前，我们先来简单了解一下 Socket 库。首先，库到底是什么东西呢？库就是一堆通用程序组件的集合，其他的应用程序都需要使用其中的组件。库有很多好处。首先，使用现成的组件搭建应用程序可以节省编程工作量；其次，多个程序使用相同的组件可以实现程序的标准化。除此之外还有很多其他的好处，因此使用库来进行软件开发的思路已经非常普及，库的种类和数量也非常之多。Socket 库也是一种库，其中包含的程序组件

① DNS: Domain Name System, 域名服务系统。将服务器名称和 IP 地址进行关联是 DNS 最常见的用法，但 DNS 的功能并不仅限于此，它还可以将邮件地址和邮件服务器进行关联，以及为各种信息关联相应的名称。

可以让其他的应用程序调用操作系统的网络功能^①，而解析器就是这个库中的其中一种程序组件。

Socket 库中包含很多用于发送和接收数据的程序组件，这些功能我们暂且放一放，先来集中精力探索一下解析器。

Socket 库是用于调用网络功能的程序组件集合。

1.2.4 通过解析器向 DNS 服务器发出查询

解析器的用法非常简单。Socket 库中的程序都是标准组件，只要从应用程序中进行调用就可以了。具体来说，在编写浏览器等应用程序的时候，只要像图 1.11 这样写上解析器的程序名称“gethostbyname”以及 Web 服务器的域名“www.lab.glasscom.com”就可以了，这样就完成了对解析器的调用^②。

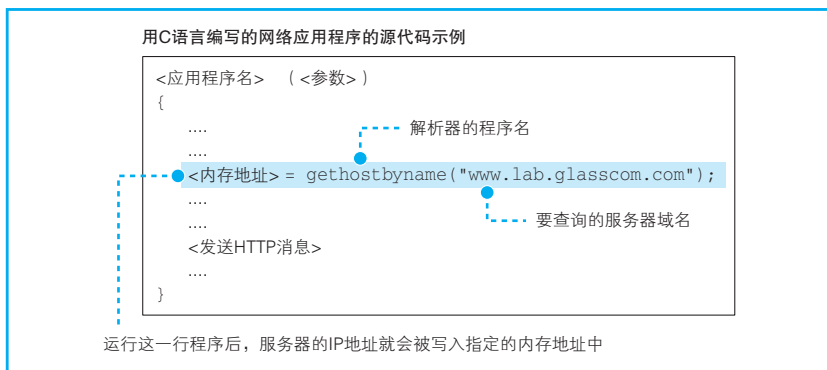


图 1.11 解析器的调用方法

在应用程序中编写上图中的一行代码后就能够调用解析器完成向 DNS 服务器查询 IP 地址的操作。

- ① Socket 库是在加州大学伯克利分校开发的 UNIX 系操作系统 BSD 中开发的 C 语言库，互联网中所使用的大多数功能都是基于 Socket 库来开发的。因此，BSD 之外的其他操作系统以及 C 语言之外的其他编程语言也参照 Socket 库开发了相应的网络库。可以说，Socket 库是网络开发中的一种标准库。
- ② 实际上，除此之外还需要编写一些用于分配保存 IP 地址的内存空间的语句，并在程序开头使用 #include 命令将其包含进来。

调用解析器后，解析器会向 DNS 服务器发送查询消息，然后 DNS 服务器会返回响应消息。响应消息中包含查询到的 IP 地址，解析器会取出 IP 地址，并将其写入浏览器指定的内存地址中。只要运行图 1.11 中的这一行程序，就可以完成前面所有这些工作，我们也就完成了 IP 地址的查询。接下来，浏览器在向 Web 服务器发送消息时，只要从该内存地址取出 IP 地址，并将它与 HTTP 请求消息一起交给操作系统就可以了。

根据域名查询 IP 地址时，浏览器会使用 Socket 库中的解析器。

1.2.5 解析器的内部原理

下面来看一看当应用程序调用解析器时，解析器内部是怎样工作的（图 1.12）。网络应用程序（在我们的场景中就是指浏览器）调用解析器时，程序的控制流程就会转移到解析器的内部。“控制流程转移”这个说法对于没有编程经验的人来说可能不容易理解，所以这里简单解释一下。

一般来说，应用程序编写的操作内容是从上往下按顺序执行的，当到达需要调用解析器的部分时，对应的那一行程序就会被执行，应用程序本身的工作就会暂停（图 1.12 ①）。然后，Socket 库中的解析器开始运行（图 1.12 ②），完成应用程序委托的操作。像这样，由于调用了其他程序，原本运行的程序进入暂停状态，而被调用的程序开始运行，这就是“控制流程转移”^①。

当控制流程转移到解析器后，解析器会生成要发送给 DNS 服务器的查询消息。这个过程与浏览器生成要发送给 Web 服务器的 HTTP 请求消息的过程类似，解析器会根据 DNS 的规格，生成一条表示“请告诉我 www.lab.glasscom.com 的 IP 地址”^②的数据，并将它发送给 DNS 服务器（图 1.12 ③）。发送消息这个操作并不是由解析器自身来执行，而是要委托给操作系统内

① 在图 1.12 中，我们假设 gethostbyname 这个程序实现了解析器的全部功能，实际上，实现解析器的功能需要多个程序相互配合，可能还会从 gethostbyname 程序中调用其他的程序。但如果继续深挖下去的话会变得复杂难懂，因此在这里我们假设 gethostbyname 实现了解析器的全部功能。

② HTTP 消息是用文本编写的，但 DNS 消息是使用二进制数据编写的。

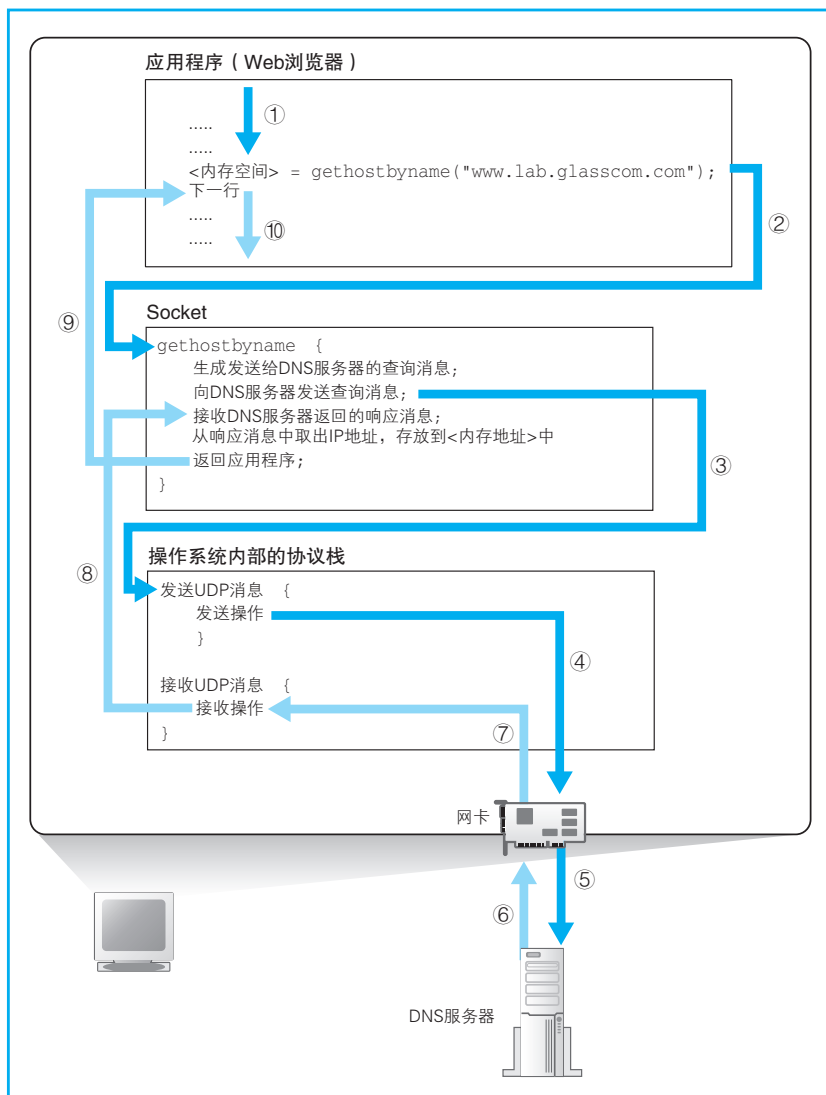


图 1.12 调用解析器时计算机内部的工作流程

通过让多个程序按顺序执行操作, 数据就被发送出去了。