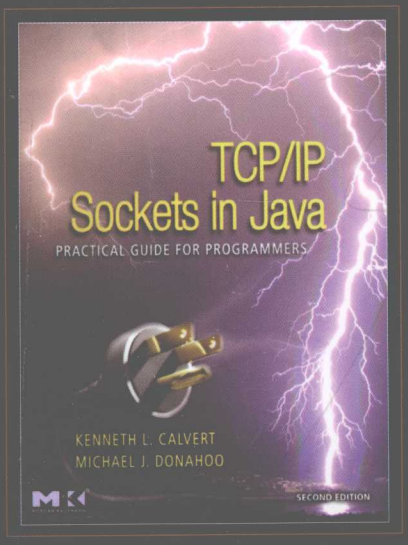
HZ BOOKS
ELSEVIER

华章程序员书库

Java TCP/IP Socket 编程（原书第2版）

TCP/IP Sockets in Java
Practical Guide for Programmers，Second Edition



（美） Kenneth L. Calvert　Michael J. Donahoo　著
周恒民　译

Java TCP/IP Socket
编程的快速实践指南

机械工业出版社
China Machine Press

华章程序员书库

Java TCP/IP Socket 编程（原书第2版）

TCP/IP Sockets in Java
Practical Guide for Programmers，Second Edition

（美） Kenneth L. Calvert Michael J. Donahoo 著
周恒民 译

机械工业出版社
China Machine Press

本书基于TCP/IP Socket相关原理，对如何在Java中进行Socket编程作了深入浅出的介绍。

本书内容简明扼要，条理清晰，并在讲解相应的概念或编程技巧时列举了大量的示例程序，每章附有练习。

本书适合作为Java Socket编程的入门教程，也可供从事网络相关专业的技术人员参考。

TCP/IP Sockets in Java:Practical Guide for Programmers,Second Edition

Kenneth L. Calvert and Michael J. Donahoo

ISBN: 978-0-12-374255-1

ISBN:978-981-272-215-7

本书版权登记号：图字：01-2008-3906

图书在版编目（CIP）数据

Java TCP/IP Socket编程（原书第2版）/（美）卡尔弗特（Calvert, K.L.），（美）多纳霍（Donahoo, M. J.）著；周恒民译. —北京：机械工业出版社，2009.1

（华章程序员书库）

书名原文：TCP/IP Sockets in Java:Practical Guide for Programmers，Second Edition

ISBN 978-7-111-25756-1

Ⅰ. J… Ⅱ. ① 卡… ② 多… ③ 周… Ⅲ. Java语言—程序设计 Ⅳ. TP 312

中国版本图书馆CIP数据核字（2008）第194493号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：李东震

北京瑞德印刷有限公司印刷

2009年1月第1版第1次印刷

186mm × 240mm · 11.5印张

标准书号：ISBN 978-7-111-25756-1

定价：29.00元

译者序

如今，TCP/IP已成为计算机网络协议事实上的标准，而Java凭借其跨平台特性和对网络编程的强大支持能力，在网络应用中已占据了主导地位。本书基于TCP/IP套接字的相关原理，对如何在Java中进行套接字编程作了深入浅出的介绍。

本书内容简明扼要，条理清晰，并在讲解相应的概念或编程技巧时列举了大量的示例程序，能够使读者在动手过程中加深理解，而每章结束时的练习可以帮助读者检查自己对已学知识的掌握程度，因此非常适合作为Java套接字编程的入门教程。虽然本书专注于介绍如何使用Java进行TCP/IP套接字编程，但其涉及的套接字相关概念和基本原理与具体编程语言无关，从而使读者能够抓住套接字编程的本质，并轻松地转向其他编程语言。

译者在翻译本书时尽量忠实于原文，必要时对原书中提到的概念作了一定的解释，并力求做到言简意赅。限于水平，翻译过程中难免有疏漏之处，敬请广大读者批评指正。

——周恒民

于北京中关村东路

2008年10月

前 言

多年来，大学里的计算机网络课程使学生很少或几乎没有动手实践的机会。由于各种各样的原因（其中也包含一些积极因素），教师仅仅通过公式、分析以及对协议栈的抽象描述来讲授计算机网络的原理。教科书里可能会包含一些代码，但都没有与学生能够动手实践的任何东西结合起来。但是我们相信，如果能让学生看到（然后实现）这些原理在实际应用中的具体例子，他们将学得更好。所幸的是情况已经发生了变化。互联网已经成为人们日常生活的一部分，大部分学生（以及他们的程序）都能快速方便地访问网络服务，而且能免费获得大量正式软件（不分优劣）。

我们基于写《TCP/IP Sockets in C》同样的目的编写了本书：我们需要一些编程练习资源来支持计算机网络课程的学习。我们旨在为学生提供充足的引导，使他们能够在真实的网络服务中实践，而不会手足无措。在掌握了基本原理后，学生就能够进一步接触一些高级任务，并从中学习到路由算法、多媒体协议、介质访问控制等相关知识。我们尽量使本书像我们之前其他书一样，让学生选择自己熟悉的编程语言和技术，从而保证他们能学会相同的技能并理解相同的概念。当然，目前尚不清楚这一目标是否可以实现，但是无论如何，本书的范围、定价以及介绍的深度都力求做到这点。

面向的读者

本书面向两种类型的读者。第一类是学习计算机网络课程的本科生或研究生，他们是促使我们写这本书的首要因素。第二类是了解Java，想要学习利用Java来编写互联网应用程序的人。我们尽量保持了内容的简洁和专一性，因此本书既可以作为学生的辅助教程，也可以作为从业者涉足这一领域的入门指南。但是，你不能期望自己读完本书后就成为这一领域的专家！本书的目的只是引导读者入门并掌握足够的知识，从而能够进行独立研究和学习。

为配合练习，读者应有一台安装Java的计算机。本书基于Java 1.6版和Java虚拟机（JVM），然而，除少量较新的方法外，本书的代码也能在更早版本的Java中运行。由于Java具有可移植性，在不同硬件和操作系统上运行程序没有差别。

内容主线

第1章对计算机网络的概念进行了总体概述。从各方面看，这一介绍并不全面，但能够使读者与贯穿全书的概念和术语相同步。第2章介绍了简单的客户端和服务器的结构，这章中的代码能作为进行各种练习的起点。第3章涵盖了有关消息的创建和解析的基础内容。读者若能理解并消化前3章的内容，将能够为简单应用协议实现一个客户端和服务器。第4章和第5章介绍了建立具有扩展性和健壮性的客户端与服务器端的高级技术，其中，第5章专注于工具的应用，并对

“New I/O”包进行了讲解。最后，为了与“通过程序来阐明原理”的目的相一致，第6章从细节上讨论了程序的构造和底层协议的实现之间的关系。

本书主要通过简单的程序实例来介绍一些编程概念，每一个实例后都附有对每行代码的注解，用以说明程序各部分的功能。这样使读者能够结合程序的上下文来理解重要的对象和方法。当你阅读代码时，就能理解每行代码的作用。

我们的例子并没有涵盖Java中所有库的应用。有些功能，特别是序列化技术，要求相互通信的所有节点都是由Java实现的。同时，为了尽快地介绍实例，我们刻意避免介绍引入之后将被清除的类和方法。我们尽量保持了内容的简洁，尤其是前面几个章节。

本书不包含哪些内容

作为一本辅导教程，为了使其定价保持在合理的范围内，我们必须对本书所涉及的内容有所限制，同时也要严格专注于前面所提出的目标。由于我们省略了某些方面的主题，因此有必要说明本书不包含哪些内容：

- 本书不是一本介绍Java编程语言的书。我们只专注于TCP/IP套接字编程，同时希望读者已经熟悉Java语言的基本语法特征和类库（包括后期发布版所包含的内容，如泛型等），并知道如何使用Java进行程序开发。
- 本书不是一本介绍协议的书。通过阅读本书并不能使你成为IP、TCP、FTP、HTTP或其他已知协议（可能反馈协议除外）的专家。我们的关注点在于套接字抽象层为TCP/IP服务所提供的接口。如果你已经对TCP协议和IP协议工作机制有所了解，这将对后续的学习有所帮助，不过第1章已经对相关内容做了足够的介绍。
- 本书并不是一本介绍隐藏了通信细节的Java类库集（如HTTPConnection）而使程序员工作变得更轻松的实用指南。本书讲授进行通信协议相关开发的基础，而不是去回避它，因此书中并没有对那些隐藏了通信细节的API进行介绍。我们希望读者能够从通信线路的传输内容上理解协议，所以本书在大部分情况下直接使用了简单的字节流和显式的字符编码，并不对URL、URLConnection等类进行介绍。相信读者一旦理解了底层的基本原理，对那些更方便的类的使用就很容易上手了。
- 本书不是一本讲解面向对象设计的书。我们致力于介绍TCP/IP套接字编程的重要原理，并通过实例对这些原理进行简要说明。本书尽可能使实例代码符合面向对象设计的思想，但如果这样做会增加代码的复杂度从而使套接字的原理变得模糊，或使代码变得臃肿，我们将把清晰性放在第一位，舍弃面向对象设计的思想。本书也没有包含有关网络编程的设计模式。（尽管我们认为本书也为理解这类设计模式提供了一些必要的背景知识！）
- 本书不是一本讲解如何编写适用于生产环境的高质量代码的书。再次声明，虽然我们尽量使代码具有一定的健壮性，但这些实例代码的主要目的还是为了教学。为了避免由于使用了大量的错误处理代码而导致原理的含糊，我们放弃了一定的健壮性，使代码更加简洁清晰。
- 本书不是一本介绍如何用Java实现自定义的本地套接字的书。我们仅专注于Java标准库所提供的TCP/IP套接字，并没有对各种实现了套接字的包装器类进行介绍（如SocketImpl类）。

- 为了避免本书中的实例聚集了过多的无关代码（即与套接字编程无关的代码），我们所有例子都是基于命令行的。在本书的网站上[⊖]有一些基于图形界面的网络应用程序的例子，本书没有将其纳入或进行讲解。
- 本书不是关于Java Applet的书。applet使用了相同的Java网络API，因此一些通信代码看起来非常相似，不过Applet所能进行的通信方式有着非常严格的安全限制。我们对这些限制进行了有限的讨论，并在本书的网站上提供了一个Applet应用程序的例子。然而，对Applet网络编程的完整介绍不属于本书讨论的范围。

致谢

感谢所有为完成本书提供了帮助的人，是他们的努力使本书得以出版。虽然本书很简短，但仍需花大量的时间来审阅原始稿件，审稿人的意见对本书的最终定稿产生了非常重要的影响。

感谢Michel Barbeau、Chris Edmondson-Yurkanan、Ted Herman、Dave Hollinger、Jim Leone、Dan Schmidt、Erick Wagner、EDS；感谢贝勒大学CSI4321班的学生以及肯塔基大学CS471班的学生。本书中存在的任何疏漏都是我们的责任。

本书不会使你成为一个专家——那需要多年的经验积累。不过我们希望本书能够成为一个有用的资源，即使对那些已经了解了很多Java套接字编程的人也有所帮助。我们享受了写书的过程并从中学到了很多。

反馈

欢迎对本书的各方面提出建议。如果你发现了书中的错误，请联系我们。我们将在本书的网站中维护一个勘误表。你可以通过本书的网站提交反馈：books.elsevier.com/ companions/9780123742551

或者向以下地址发送电子邮件：

Kenneth L. Calvert：calvert@uky.edu

Michael J. Donahoo：Jeff_Donahoo@baylor.edu

⊖ *books.elsevier.com/companions/9780123742551*。

目 录

第1章 简 介

如今，人们可以通过电脑来打电话，看电视，与朋友聊天，与其他人玩游戏，甚至可以通过电脑买到你能想到的任何东西，包括从歌曲到SUV[㊀]。计算机程序能够通过互联网相互通信使这一切成为了可能。很难统计现在有多少个人电脑接入互联网，但可以肯定，这个数量增长得非常迅速，相信不久就能达到10亿。除此之外，新的应用程序每天在互联网上层出不穷。随着日益增加的互联网访问带宽，我们可以预见，互联网将会对人们将来的生活产生长远的影响。

那么程序是如何通过网络进行相互通信的呢？本书的目的就是通过在Java编程语言环境下，带领你进入对这个问题的解答之路。Java语言从一开始就是为了让人们使用互联网而设计的，它为实现程序的相互通信提供了许多有用的抽象应用编程接口（*Application Programming Interface*，*API*），这类应用编程接口被称为套接字（*socket*）。

在我们开始探究套接字的细节之前，有必要向读者简单介绍计算机网络和通信协议的整体框架，以使读者能清楚我们的代码将应用的地方。本章的目的不是向读者介绍计算机网络和TCP/IP协议是如何工作的（已经有很多相关内容的教程[㊁㊂㊃㊄㊅]），而是介绍一些基本的概念和术语。

㊀ SUV，英文Sports Utility Vehicles的缩写，中文意思是运动型多用途汽车。——译者注

㊁ Comer, Douglas E., *Internetworking with TCP/IP, Volume I: Principles, Protocols, and Architecture* (fourth edition), Prentice-Hall, 2000.

㊂ Comer, Douglas E., and Stevens, David L., *Internetworking with TCP/IP, Volume III: Client-Server Programming and Applications* (Linux/POSIX Sockets Version), Prentice-Hall, 2001.

㊃ Peterson, Larry L., and Davie, Bruce S., *Computer Networks: A Systems Approach* (third edition), Morgan Kaufmann, 2003.

㊄ Stevens, W. Richard, *UNIX Network Programming: Networking APIs: Sockets and XTI* (second edition), Prentice-Hall, 1997.

㊅ Stevens, W. Richard, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.

1.1 计算机网络、分组报文和协议

计算机网络由一组通过通信信道相互连接的机器组成。我们把这些机器称为主机（*hosts*）和路由器（*routers*）。主机是指运行应用程序的计算机，这些应用程序包括网络浏览器（Web browser）、即时通信代理（IM agent）或者文件共享程序。运行在主机上的应用程序才是计算机网络的真正“用户”。路由器的作用是将信息从一个通信信道传递或转发（*forward*）到另一个通信信道。路由器上可能会运行一些程序，但大多数情况下它们是不运行应用程序的。基于本书的目的对通信信道（*communication channel*）进行解释：它是将字节序列从一个主机传输到另一个主机的一种手段，可能是有线电缆，如以太网（Ethernet），也可能是无线的，如WiFi[⊖]，或是其他方式的连接。

路由器非常重要，因为要想直接将所有不同主机连接起来是不可行的。相反，一些主机先得连接到路由器，这些路由器再连接到其他路由器，这样就形成了网络。这种布局使每个主机只需要用到数量相对较少的通信信道，大部分主机仅需要一条信道。在网络上相互传递信息的程序并不直接与路由器进行交互，它们基本上感觉不到路由器的存在。

这里的信息（*information*）是指由程序创建和解释的字节序列。在计算机网络环境中，这些字节序列称为分组报文（*packet*）。一组报文包括了网络用来完成工作的控制信息，有时还包括一些用户数据。用于定位分组报文目的地址的信息就是一个例子。路由器正是利用了这些控制信息来实现对每个报文的转发。

协议（*protocol*）相当于相互通信的程序间达成的一种约定，它规定了分组报文的交换方式和它们包含的意义。一组协议规定了分组报文的结构（例如报文中的哪一部分表明了其目的地址）以及怎样对报文中所包含的信息进行解析。设计一组协议，通常是为了在一定约束条件下解决某一特定的问题。比如，超文本传输协议（HyperText Transfer Protocol，HTTP）是为了解决在服务器间传递超文本对象的问题，这些超文本对象在服务器中创建和存储，并由Web浏览器进行可视化，以使其对用户有用。即时消息协议是为了使两个或更多用户间能够交换简短的文本信息。

要实现一个有用的网络，必须解决大量各种各样的问题。为了使这些问题可管理和模块化，人们设计了不同的协议来解决不同类型的问题。TCP/IP协议就是这样一组解决

⊖ WiFi，全称Wireless Fidelity，即无线保真，是一种短距离无线技术。——译者注

方案，有时也称为协议族（*protocol suite*）。它刚好是互联网所使用的协议，不过也能用在独立的专用网络中。本书以后所提到的网络（*network*）都是指任何使用了TCP/IP协议族的网络。TCP/IP协议族主要协议有IP协议（*Internet Protocol*[⊖]互联网协议），TCP协议（*Transmission Control Protocol*[⊜]，传输控制协议）和UDP协议（*User Datagram Protocol*[⊜]，用户数据报协议）。

事实证明将各种协议分层组织是一种非常有用的措施，TCP/IP协议族（实际上其他所有协议族）都是按这种方式组织的。图1-1展示了主机和路由器中的通信协议、应用程序和套接字API之间的关系，同时也展示了数据流从一个应用程序到另一个应用程序的过程（使用TCP协议）。标记为TCP、UDP和IP的方框分别代表了这些协议的实现，它们通常驻留在主机的操作系统中。应用程序通过套接字API对UDP协议和TCP协议所提供的服务进行访问。箭头描述了数据流从一个应用程序，经过TCP协议层和IP协议层，通过网络，再经过IP协议层和TCP协议层传输到另一端的应用程序。

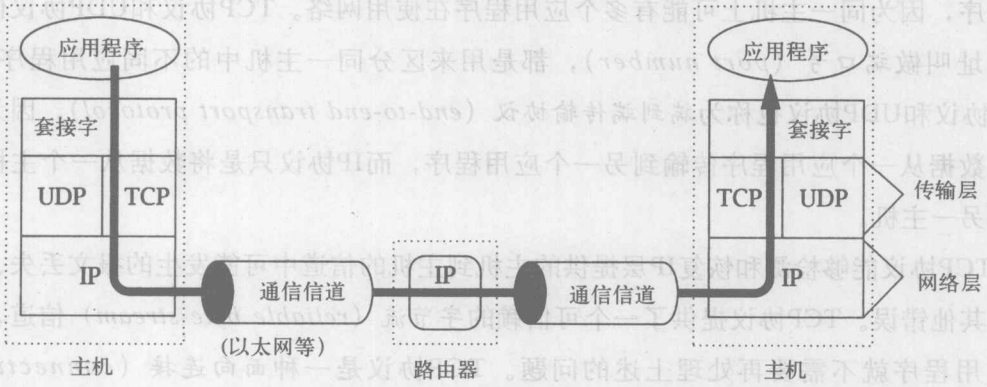


图1-1 一个TCP/IP网络

在TCP/IP协议族中，底层由基础的通信信道构成，如以太网或调制解调器拨号连接。这些信道由网络层（*network layer*）使用，而网络层则完成将分组报文传输到它们的目的地址的工作（也就是路由器的功能）。TCP/IP协议族中属于网络层的唯一协议是IP协议，它使两个主机间的一系列通信信道和路由器看起来像是单独一条主机到

⊖ Postel, John, "Internet Protocol," Internet Request for Comments 791, September 1981.

⊜ Postel, John, "Transmission Control Protocol," Internet Request for Comments 793, September 1981.

⊜ Postel, John, "User Datagram Protocol," Internet Request for Comments 768, August 1980.

主机的信道。

IP协议提供了一种数据报服务：每组分组报文都由网络独立处理和分发，就像信件或包裹通过邮政系统发送一样。为了实现这个功能，每个IP报文必须包含一个保存其目的地址（address）的字段，就像你所投递的每份包裹都写明了收件人地址。（我们随即会对地址进行更详细的说明。）尽管绝大部分递送公司会保证将包裹送达，但IP协议只是一个“尽力而为”（best-effort）的协议：它试图分发每一个分组报文，但在网络传输过程中，偶尔也会发生丢失报文，使报文顺序被打乱，或重复发送报文的情况。

IP协议层之上称为*传输层*（*transport layer*）。它提供了两种可选择的协议：TCP协议和UDP协议。这两种协议都建立在IP层所提供的服务基础上，但根据*应用程序协议*（*application protocol*）的不同需求，它们使用了不同的方法来实现不同方式的传输。TCP协议和UDP协议有一个共同的功能，即寻址。回顾一下，IP协议只是将分组报文分发到了不同的主机，很明显，还需要更细粒度的寻址将报文发送到主机中指定的应用程序，因为同一主机上可能有多个应用程序在使用网络。TCP协议和UDP协议使用的地址叫做*端口号*（*port number*），都是用来区分同一主机中的不同应用程序的。TCP协议和UDP协议也称为*端到端传输协议*（*end-to-end transport protocol*），因为它们将数据从一个应用程序传输到另一个应用程序，而IP协议只是将数据从一个主机传输到另一主机。

TCP协议能够检测和恢复IP层提供的主机到主机的信道中可能发生的报文丢失、重复及其他错误。TCP协议提供了一个可信赖的*字节流*（*reliable byte-stream*）信道，这样应用程序就不需要再处理上述的问题。TCP协议是一种*面向连接*（*connection-oriented*）的协议：在使用它进行通信之前，两个应用程序之间首先要建立一个TCP连接，这涉及相互通信的两台电脑的TCP部件间完成的握手消息（*handshake message*）的交换。使用TCP协议在很多方面都与文件的*输入输出*（*Input/Output*，*I/O*）相似。实际上，由一个程序写入的文件再由另一个程序读取就是一个TCP连接的适当模型。另一方面，UDP协议并不尝试对IP层产生的错误进行修复，它仅仅简单地扩展了IP协议“尽力而为”的数据报服务，使它能够在应用程序之间工作，而不是在主机之间工作。因此，使用了UDP协议的应用程序必须为处理报文丢失、顺序混乱等问题做好准备。

1.2 关于地址

寄信的时候，要在表格中填上邮政服务能够理解的收信人的地址。在给别人打电话时，必须拨电话号码。同样，一个程序要与另一个程序通信，就要给网络提供足够的信息，使其能够找到另一个程序。在TCP/IP协议中，有两部分信息用来定位一个指定的程序：*互联网地址*（*Internet address*）和*端口号*（*port number*）。其中互联网地址由IP协议使用，而附加的端口地址信息由传输协议（TCP或IP协议）对其进行解析。

互联网地址由二进制数字组成，有两种型式，分别对应了两个版本的标准互联网协议。现在最常用的版本是版本4，即IPv4[⊖]，另一个版本是刚开始开发的版本6，即IPv6[⊜]。IPv4的地址长32位，只能区分大约40亿个独立地址，对于如今的互联网来说，这是不够大的。（也许看起来很多，但由于地址的分配方式的原因，有很多都被浪费了）出于这个原因引入了IPv6，它的地址有128位长。

为了便于人们使用互联网地址（相对于程序内部的表示），两个版本的IP协议有不同的表示方法。IPv4地址被表示为一组4个十进制数，每两个数字之间由圆点隔开（如：10.1.2.3），这种表示方法叫做*点分形式*（*dotted-quad*）。点分形式字符串中的4个数字代表了互联网地址的4个字节，也就是说，每个数字的范围是0～255。

另一方面，IPv6地址的16个字节由几组16进制的数字表示，这些16进制数之间由分号隔开（如：2000:fdb8:0000:0000:0001:00ab:853c:39a1）。每组数字分别代表了地址中的两个字节，并且每组开头的0可以省略，因此前面的例子中，第5组和第6组数字可以缩写为:1:ab:。甚至，只包含0的连续组可以全部省略（但在一个地址中只能这样做一次）。因此，该例子的完整地址可以表示为2000:fdb8::1:00ab:853c:39a1。

从技术角度来讲，每个互联网地址代表了一台主机与底层的通信信道的连接，换句话说，也就是一个*网络接口*（*network interface*）。主机可以有多个接口，这并不少见，例如一台主机同时连接了有线以太网（Ethernet）和无线网（WiFi）。由于每个这样的连接都属于唯一的一台主机，所以只要它连接到网络，一个互联网地址就能定位这条主机。但是反过来，一台主机并不对应一个互联网地址。因为每台主机可以有多个接口，每个

⊖ Postel, John, "Internet Protocol," Internet Request for Comments 791, September 1981.

⊜ Deering, S., and Hinden, R, "Internet Protocol, Version 6 (IPv6) Specification," Internet Request for Comments 2460, December 1998.

接口又可以有多个地址。（实际上一个接口可以同时拥有IPv4地址和IPv6地址）。

TCP或UDP协议中的端口号总与一个互联网地址相关联。回到前面我们作类比的例子，一个端口号就相当于指定街道上一栋大楼的某个房间号。邮政服务通过街道地址把信分发到一个邮箱，再由清空邮箱的人把这封信递送到这栋楼的正确房间中。或者考虑一个公司的内部电话系统：要与这个公司中的某个人通话，首先要拨打该公司的总机电话号码连接到其内部电话系统，然后再拨打你要找的那个人的分机号码。在上面的例子中，互联网地址就相对于街道地址或公司的总机电话号码，端口号就相当于房间号或分机号码。端口号是一组16位的无符号二进制数，每个端口号的范围是1～65 535（0被保留）。

每个版本的IP协议都定义了一些特殊用途的地址。其中值得注意的一个是回环地址（*loopback address*），该地址总是被分配个一个特殊的回环接口（*loopback interface*）。回环接口是一种虚拟设备，它的功能只是简单地将发送给它的报文直接回发给发送者。回环接口在测试中非常有用，因为发送给这个地址的报文能够立即返回到目标地址。而且每台主机上都有回环接口，即使当这台计算机没有其他接口（也就是说没有连接到网络），回环接口也能使用。IPv4的回环地址是127.0.0.1[⊖]，IPv6的回环地址是0:0:0:0:0:0:0:1。

IPv4地址中的另一种特殊用途的保留地址包括那些“私有用途”的地址。它们包括IPv4中所有以10或192.168开头的地址，以及第一个数是172，第二个数在16～31的地址。（在IPv6中没有相应的这类地址）这类地址最初是为了在私有网络中使用而设计的，不属于公共互联网的一部分。现在这类地址通常被用在家庭或小型办公室中，这些地方通过*NAT*（*Network Address Translation*，网络地址转换）设备连接到互联网。NAT设备的功能就像一个路由器，转发分组报文时将转换（重写）报文中的地址和端口。更准确地说，它将一个接口中报文的私有地址端口对（private address, port pairs）映射成另一个接口中的公有地址端口对（public address, port pairs）。这就使一小组主机（如家庭网络）能够有效地共享同一个IP地址。重要的是这些内部地址不能从公共互联网访问。如果你在拥有私有类型地址的计算机上试验本书的例子，并试图与另一台没有这类地址的主机进行通信，通常只有这台拥有私有类型地址的主机发起的通信才能成功。

相关的类型的地址包括本地链接（*link-local*），或称为“自动配置”地址。IPv4中，

⊖ 从技术上讲，任何由127开头的IPv4地址都应该回环。

这类地址由169.254开头，在IPv6中，前16位由FE8开头的地址是本地链接地址。这类地址只能用来在连接到同一网络的主机之间进行通信，路由器不会转发这类地址的信息。

最后，另一类地址由多播（*multicast*）地址组成。普通的IP地址（有时也称为“单播”地址）只与唯一一个目的地址相关联，而多播地址可能与任意数量的目的地址关联。我们将在第4章中简要地对多播技术作进一步介绍。IPv4中的多播地址在点分格式中，第一个数字在224～239之间。IPv6中，多播地址由FF开始。

1.3 关于名字

也许你更习惯于通过名字来指代一个主机，例如：host.example.com。然而，互联网协议只能处理二进制的网络地址，而不是主机名。首先应该明确的是，使用主机名而不使用地址是出于方便性的考虑，这与TCP/IP提供的基本服务是相互独立的。你也可以不使用名字来编写和使用TCP/IP应用程序。当使用名字来定位一个通信终端时，系统将做一些额外的工作把名字解析成地址。有两个原因证明这额外的步骤是值得的：第一，相对于点分形式（或IPv6中的十六进制数字串），人们更容易记住名字；第二，名字提供了一个间接层，使IP地址的变化对用户不可见。在本书英文版第一版的写作期间，网络服务器www.mkp.com的地址就改变过。由于我们通常都使用网络服务器的名字，而且地址的改变很快就被反应到映射主机名和网络地址的服务上（我们马上会对其进行更多的介绍），如www.mkp.com从之前的地址208.164.121.48对应到了现在的地址，这种变化对通过名字访问该网络服务器的程序是透明的。

名字解析服务可以从各种各样的信息源获取信息。两个主要的信息源是域名系统（*Domain Name System*，*DNS*）和本地配置数据库。DNS[⊖]是一种分布式数据库，它将像*www.mkp.com*这样的域名映射到真实的互联网地址和其他信息上。DNS协议[⊜]允许连接到互联网的主机通过TCP或UDP协议从DNS数据库中获取信息。本地配置数据库通常是一种与具体操作系统相关的机制，用来实现本地名称与互联网地址的映射。

⊖ Mockapetris, Paul, “Domain Names-Concepts and Facilities,” Internet Request for Comments 1034, November 1987.

⊜ Mockapetris, Paul, “Domain Names-Implementation and Specification,” Internet Request for Comments 1035, November 1987.

1.4 客户端和服务器

在前面的邮政和电话系统例子中，每次通信都是由发信方或打电话者发起，而另一方则通过发回反馈信或接听电话来对通信的发起者作出响应。互联网通信也与这个过程类似。客户端（*client*）和服务器（*server*）这两个术语代表了两种角色：客户端是通信的发起者，而服务器程序则被动等待客户端发起通信，并对其作出响应。客户端与服务器组成了应用程序（*application*）。客户端和服务器这两个术语对典型的情况作出了描述，服务器具有一定的特殊能力，如提供数据库服务，并使任何客户端能够与之通信。

一个程序是作为客户端还是服务器，决定了它在与其对等端（*peer*）建立通信时使用的套接字API的形式（客户端的对等端是服务器，反之亦然）。更进一步来说，客户端与服务器端的区别非常重要，因为客户端首先需要知道服务器的地址和端口号，反之则不需要。如果有必要，服务器可以使用套接字API，从收到的第一个客户端通信消息中获取其地址信息。这与打电话非常相似：被呼叫者不需要知道拨电话者的电话号码。就像打电话一样，只要通信连接建立成功，服务器和客户端之间就没有区别了。

客户端如何才能找到服务器的地址和端口号呢？通常情况，客户端知道服务器的名字，例如使用*URL*（*Universal Resource Locator*，统一资源定位符）如*http://www.mkp.com*，再通过名字解析服务获取其相应的互联网地址。

获取服务器的端口号则是另一种情况。从原理上来讲，服务器可以使用任何端口号，但客户端必须能够获知这些端口号。在互联网上，一些常用的端口号被约定赋给了某些应用程序。例如，端口号21被*FTP*（*File Transfer Protocol*，文件传输协议）使用。当你运行FTP客户端应用程序时，它将默认通过这个端口号连接服务器。互联网的端口号授权机构维护了一个包含所有已约定使用的端口号列表（见*http://www.iana.org/assignments/port-numbers*）。

1.5 什么是套接字

socket（套接字）是一种抽象层，应用程序通过它来发送和接收数据，就像应用程序打开一个文件句柄，将数据读写到稳定的存储器上一样。使用socket可以将应用程序添加到网络中，并与处于同一个网络中的其他应用程序进行通信。一台计算机上的应用程序向socket写入的信息能够被另一台计算机上的另一个应用程序读取，反之亦然。

不同类型的socket与不同类型的底层协议族以及同一协议族中的不同协议栈相关联，