TURING 图灵程序设计丛书

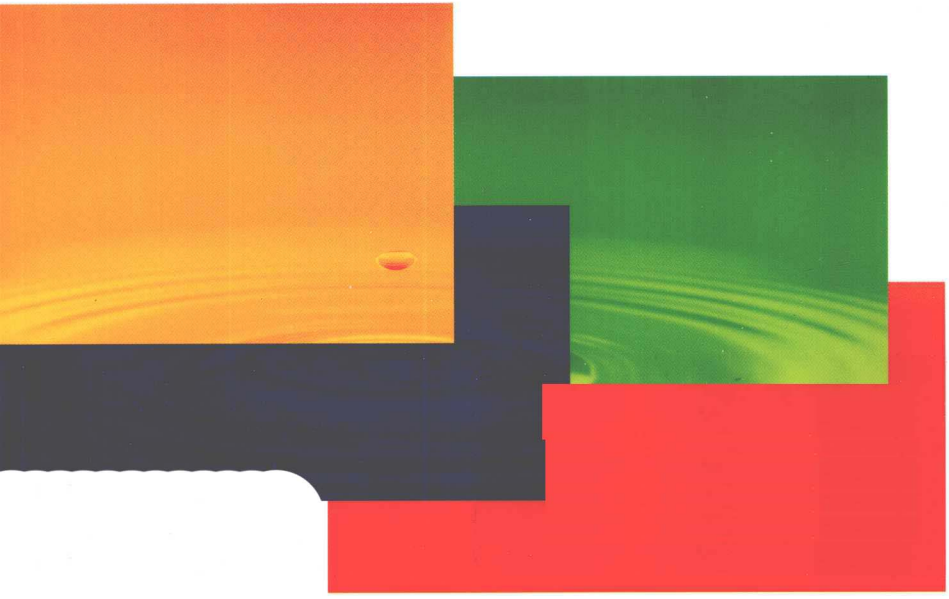
Addison Wesley

Effective TCP/IP Programming

44 Tips to Improve Your Network Programs

TCP/IP高效编程

改善网络程序的44个技巧



[美] Jon C. Snader 著
陈涓 赵振平 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Effective TCP/IP Programming

44 Tips to Improve Your Network Programs

TCP/IP高效编程

改善网络程序的44个技巧

[美] Jon C. Snader 著
陈涓 赵振平 译

人民邮电出版社
北京

图书在版编目（CIP）数据

TCP/IP高效编程 : 改善网络程序的44个技巧 / (美)斯纳德 (Snader, J.C.) 著 ; 陈涓, 赵振平译. -- 北京: 人民邮电出版社, 2011.4
(图灵程序设计丛书)
书名原文: Effective TCP/IP Programming: 44 Tips to Improve Your Network Programs
ISBN 978-7-115-24937-1

Ⅰ. ①T… Ⅱ. ①斯… ②陈… ③赵… Ⅲ. ①计算机网络－通信协议 Ⅳ. ①TN915.04

中国版本图书馆CIP数据核字(2011)第026264号

内 容 提 要

本书是TCP/IP领域的经典著作，对TCP/IP网络编程中存在的各种问题进行了全面解析，旨在帮助读者深入透彻地理解TCP/IP网络编程。本书组织方式比较特别，正文部分包括4章，将网络编程中存在的常见问题组织成44个技巧，探讨问题的过程中构建并运行了多个程序，并且指出了代码的源地址，便于读者查看。全书以技巧的形式解答了日常工作中遇到的经典问题，将本书作为手册使用，极其方便。

本书主要面向有一定经验的初学者或中级网络程序员，也可作为计算机相关专业人士的参考读物。

图灵程序设计丛书

TCP/IP 高效编程
改善网络程序的44个技巧

◆ 著 [美] Jon C. Snader
译 陈 涓 赵振平
责任编辑 朱 巍
执行编辑 刘美英

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
中国铁道出版社印刷厂印刷

◆ 开本：800×1000 1/16
印张：17.5
字数：420千字 2011年4月第1版
印数：1－3 000册 2011年4月北京第1次印刷

著作权合同登记号 图字：01-2010-4224号
ISBN 978-7-115-24937-1

定价：55.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

译 者 序

网络已经无处不在，和我们的生活息息相关，绝大部分有趣和有用的程序都需要进行网络访问。

TCP/IP 是因特网的核心协议，它最流行的编程接口是套接字。应用程序通过套接字可以很容易地进行网络通信。然而，实际的程序在运行时，常会碰到各种各样的性能和连通性方面的问题。如果对网络协议的底层运作细节没有深入的了解，就会被这些问题所困扰，难以开发出高效和稳健的应用程序。

Jon C. Snader 的这本书是深入学习 TCP/IP 网络编程的必备书籍。通过一个个独立的技巧，本书对网络编程中可能碰到的各种微妙问题一一进行了深入的分析和阐释，并提供了很多可以方便扩展的实例代码。有些复杂的问题还在多个技巧中反复提起，从多种角度帮助读者加深理解。本书还对解决联网问题的常用工具进行了精要的介绍，帮助读者利用工具提高分析和解决联网问题的能力。

如今的编程工作除了靠埋头苦干，还要充分利用网络带来的便利，学习和利用世界上其他编程人员的优秀成果，避免毫无意义的重复劳动，切不可闭门造车或重蹈覆辙。本书最后的阅读代码技巧中，对网络和操作系统方面的可以公开获得的优秀书籍和开源项目进行了梳理，旨在帮助读者找到适宜的学习目标，从而少走弯路，加快学习和成长的速度。

本书适用读者范围广泛，对有一定网络编程基础和经验丰富的工程师都具有很高的参考价值。我自身虽然具有多年网络编程教学和产品开发方面的实践经验，但在翻译本书的过程中也学习和完善了很多知识。

IT 技术发展迅猛，许多书籍很快就会过时。而本书的英文原版虽然出版于 2000 年，但它是作者长期分析实践总结出的真知灼见，如今看来，它不但没有过时，还将会伴随 TCP/IP 体系的进一步发展，产生深远的影响。

陈涓

2011 年 1 月

献给 Maria

前　言

简介

因特网、无线通信以及联网技术总体上的爆炸性发展，使得编写联网应用程序的程序员和工程师的数量也在激增。TCP/IP 编程看起来很容易，因而很有诱惑力。API（Application Programming Interface，应用编程接口）也很易懂，即使是超级新手也可以把客户端或服务器模板充实为能够工作的应用程序。

但是，在最初的生产率激增之后，新手们往往会陷入细节的泥沼之中，并发现自己的应用程序饱受性能及健壮性问题的困扰。网络编程过程中的黑暗角落和易误解的地方比比皆是。本书就像一盏明灯照亮了这些角落，并通过对 TCP/IP 编程中一些微妙问题的解释来帮助大家纠正了很多错误的理解。

看完本书之后，你会透彻地理解网络编程中的许多问题。书中讨论了很多看起来好像与一个网络程序员所需掌握的核心知识只有一些外围联系的内容。但我们会看到，理解了这些细节问题，就可以帮助我们更好地理解网络协议的内部运作机制是怎样与应用程序进行交互的。有了这种洞察力，原来看起来让人困惑的程序行为就会变得很好理解，怎么解决问题也就很明显了。

本书的组织方式有些与众不同，我们将一些常见问题组织成一系列的技巧，每个技巧讨论一个问题。在讨论特定问题的过程中，通常会对 TCP/IP 编程的某些方面进行深入的研究。完成对特定问题的讨论之后，我们不仅能够掌握这些常见问题的特点和解决方案，还会对 TCP/IP 协议是怎样工作的，以及它如何与我们编写的应用程序进行交互有相当全面的理解。

将内容以技巧的形式组织起来会显得不够连贯，为了便于理解，第 1 章提供了相应的内容介绍，对每章涵盖的技巧以及这些技巧之间的联系进行了说明。目录中列出了每一个技巧，可以从中看出正文的组织结构。每个技巧的标题都是祈使句，因此也可以把目录当作网络编程的规则来用。

从另一方面来看，将内容以技巧的形式组织起来，使本书更像一本实用而又方便查阅的手册。在日常工作中遇到问题时，可以很方便地去重温其中的技巧，加深对特定问题的理解。你会发现很多话题都会在多个技巧中提到，有时只是换个角度来讨论。这种重复有助于巩固概念，让你对

它们达成更自然的理解。

读者对象

本书主要是面向有一定经验的初学者或中级网络程序员，但经验更丰富的读者会发现本书也是很有用的。我们假设读者熟悉联网知识，以及基本的套接字 API，但第 1 章也对基本的套接字调用进行了复习，并用它们构建了初级的客户端和服务器。技巧 4 更详细地介绍了各种客户端和服务器模型。因此，哪怕只具有少量背景知识的读者也能理解本书，并从中获益。

由于所有的示例基本上都是用 C 语言编写的，因此要看懂本书的程序，至少需要有基本的 C 语言编程经验。技巧 31 展示了一些用 Perl 编写的例子，但并不需要读者掌握 Perl 的知识。类似地，还有几个小的 shell 程序示例，理解它们同样也不需要有 shell 编程的经验。

本书所有示例和内容都争取做到独立于平台。除了少数几个例子之外，其他示例都可以在任何 UNIX 或 Win32 系统中编译并运行。即使那些没有工作在 UNIX 或 Windows 系统下的程序员，要把这些例子移植到他们所使用的平台上去也应该不会有什么问题。

排版约定

在探讨技巧的过程中，我们会构建并运行很多小程序，设计这些小程序是为了从不同方面来说明问题。在显示交互式输入输出时，会使用下列约定：

- 输入的文本被设置为粗体的 **Courier**；
- 系统输出的文本被设置为标准的 Courier；
- 非实际输入或输出的注释内容，用楷体表示。

下面是技巧 9 中的一个例子。

```
bsd: $ tcprw localhost 9000
hello
received message 1                    延迟 5 秒之后打印此消息
                                      服务器在此被杀死
hello again
tcprw: readline failed: Connection reset by peer (54)
bsd: $
```

注意，我们在 shell 提示符中包含了系统的名称。在上面的例子中可以看到，tcprw 运行在一台名为 bsd 的主机上。

在介绍新的 API 函数时，不管是自己开发的，还是标准系统调用中的函数，都将其包含在一个方框中。标准的系统调用包含在实线框中，如下所示。

```
#include <sys/socket.h>       /* UNIX */
#include <winsock2.h>         /* Windows */

int connect( SOCKET s, const struct sockaddr *peer, int peer_len );
```

返回：成功时返回 0，失败时返回-1（UNIX）或非零值（Windows）

自己开发的 API 函数包含在虚线框中，如下所示。

```
#include "etcp.h"

SOCKET tcp_server ( char *host, char *port );
```

返回：一个监听套接字（出错时终止）

插入的说明按正文格式排版，并像本段这样字体设置为楷体，就不放在脚注中了。通常，在第一次阅读时可以跳过这些内容。

源代码及勘误

本书所有示例源代码网上都有，可以在 http://www.netcom.com/~jsnader 上找到。本书的例子都是从编程文件中拿来直接排版使用的，读者也可以下载这些例子并进行试验。框架和库代码也可以从网上下载。

同一个 Web 站点还列出了勘误表。

Richard Stevens

我很推崇 Richard Stevens 的书的版式风格。我开始写这本书的时候，给 Richard 写了一封信，询问他是否介意我使用他的版式风格。大度的 Richard 不仅回复说他没有异议，还给我发了一份他排版自己的书时使用的 Groff 宏的副本，帮助并参与了我的“偷盗”行为。

如果你觉得本书的外观和设计赏心悦目，那么应该感谢 Richard。如果布局上有些地方看起来不太对劲儿，或者不够赏心悦目，可以去看看 Richard 的书，那是我的目标。

工具及脚本作者

我用 James Clark 的 Groff 包和 Richard Stevens 修改过的 `ms` 宏生成了本书的待印版本。插图是用（包含 Richard Stevens 和 Gary Wright 所写宏的）`gpic` 生成的，表格是用 `gtbl` 生成的，（有限的）数学表达式是用 `geqn` 生成的。索引是利用 Jon Bentley 和 Brian Kernighan 编写的一组 `awk` 脚本生成的。示例代码是在 Dave Hanson 的 loom 程序的帮助下，从程序文件中直接插入到正文中的。

致谢

作者感谢他们的家人在书籍编写过程中所给予的帮助和支持已经成为一种惯例了，现在我才知道这是为什么。如果没有我妻子 Maria，就不可能有这本书。如果不是她承担了比我通常的“一半”更多的工作，我就不会有时间编写更谈不上完成这本书。遗憾的是，与她所承担的额外家务以及她独自熬过的孤单夜晚相比，这些感谢的话还是显得太单薄了。

目　录

第1章 概 述

编写本书是为了帮助有一定经验的初学者或中级网络程序员向熟练程序员，甚至网络专家转变。要成为专家主要取决于经验以及对特定知识（有时可能是比较难以理解的知识）的积累。经验只有花时间从实践中获得，但本书可以提供知识方面的帮助。

当然，网络编程是个范围很广的领域，要在两台或多台机器之间进行通信，可选择的联网技术有很多。有简单的，比如串行链路，有复杂的，比如 IBM 的 SNA（System Network Archtecture，系统网络结构）都有可能。如今，日益明确的一点是 TCP/IP 协议族已经成为了构建网络的首选技术。这很大程度上是由因特网及其最广泛的应用——WWW（World Wide Web，万维网）推动的。

> 当然，Web 实际上并不是一个应用程序。它也不是协议，尽管它既使用了应用程序（Web 浏览器和服务器），也使用了协议（比如 HTTP）。也就是说，Web 是运行在因特网上的、用户可见的、最流行的联网技术的应用。

在 Web 出现之前，TCP/IP 就已经是一种流行的网络构建方法了。因为它是一种开放的标准，可以连接来自不同厂商的机器，所以越来越多地被用于构建网络和网络应用程序。到 20 世纪 90 年代末，TCP/IP 已成为主导的联网技术，而且可能在很长一段时间内都会保持这种状态。鉴于此，我们把重点放在了 TCP/IP 以及运行它的网络上。

要想掌握网络编程技术，首先必须掌握一些必要的背景知识，以便更完整地理解和体会这门技术的真义。我们将通过研究初级网络程序员面临的一些常见问题来介绍这些知识。很多常见问题都是由于对 TCP/IP 协议以及与之通信的 API 的某些方面产生了误解或理解得不全面造成的。所有这些问题都是实际存在的。这些问题不断地困扰着大家，也是网络新闻组中的常见话题。

1.1 几个约定

除了几个明显的例子外，本书的内容和程序都尽量做到与（32 位或 64 位的）UNIX 系统和微软的 Win32 API 兼容。我们不打算去兼容 16 位 Windows 应用程序。但几乎所有内容和很多程序都适用于其他环境。

对可移植性的追求使得代码实例中存在一些别扭的地方。比如，UNIX 程序员可能不习惯把本来是自然的 `int` 类型的套接字描述符定义成 `SOCKET` 类型，而 Windows 程序员则会注意到我们使用的都是控制台程序。技巧 4 对这些约定进行了描述。

类似，在大多数情况下，我们都避免在套接字上使用 read 和 write，因为 Win32 API 在套接字上不支持这些系统调用。我们经常会提到从套接字上读取或者向套接字中写入，但指的都是一般意义上的读写。我们所说的“读”指的是 recv、recvfrom 或者 recvmsg；我们所说的“写”指的是 send、sendto 或者 sendmsg。特指“read”这种系统调用时，我们会用 read 这样的 Courier 字体显示，对“write”的处理也一样。

最难作的决定之一就是是否包含与 IPv6 有关的内容，IPv6 是 IP（Internet Protocol，网际协议）当前版本（IPv4）的替换版本。最终我们决定不包含 IPv6 的内容。这么做的原因有很多，其中包括：

- 本书中几乎所有的内容同时适用于 IPv4 和 IPv6；
- 两者之间确实存在的区别，通常都局限于 API 的寻址部分；
- 本书的编写主要基于熟练网络程序员的经验和知识，而 IPv6 才刚刚开始大规模实施，我们确实还没有使用 IPv6 的经验。

因此，如果没有特别声明，我们所说的 IP 指的就是 IPv4。在的确提到 IPv6 的地方，我们会有明确的说明。

最后一点是，我们将 8 比特数据单元称为一字节。在网络世界中经常将这种数据单元称为八位位组（octet）。这是由历史原因造成的。以前，字节的长度是依赖于平台的，而且，对其确切的长度也没有达成一致。为了避免二义性，早期的网络文献发明了八位位组这个术语，用来明确地说明这个长度。现在，大家普遍认为一个字节就是 8 比特长[Kernighan and Pike, 1999]，使用八位位组就显得学究气太浓，没有必要了。

> 尽管如此，偶尔仍然会看到有人说字节是 8 比特长的时候，然后，就会有“嗨，孩子们！我还是个孩子的时候，在 Frumbaz-6 上工作过，那上面一个字节是五个半比特长的。可别告诉我一个字节总是八个比特长的”这样的帖子在 Usenet 上引发争论。

2

1.2　本书其余部分的内容介绍

本章的其余部分回顾了编写 TCP/IP 应用程序时会用到的基本套接字 API 和客户端-服务器架构，这是掌握技术的必要基础。

第 2 章讨论了一些与 TCP/IP 和联网有关的基本概念以及对它们的误解。比如，我们讨论了面向连接和无连接协议之间的区别。对 IP 寻址和一些容易混淆的问题，比如子网、CIDR（Classless Interdomain Routing，无类别域间路由）和 NAT（Network Address Translation，网络地址翻译）进行了研究。我们看到，实际上 TCP/IP 并不能保证数据的可靠传送，我们必须提防对等实体或用户端的异常行为，而且应用程序在 WAN（Wide Area Network，广域网）中的行为很可能与在 LAN（Local Area Network，局域网）中的行为有所不同。

这一章会提醒我们，TCP 是一种流协议，以及这对我们这些程序员意味着什么。同样，我们还会了解到 TCP 不会自动检测连接是否丢失，为什么这是一件好事，以及我们能做些什么。

我们会了解到为什么套接字 API 比 TLI/XTI（Transport Layer Interface/ X/Open Transport

Interface，传输层接口以及 X/Open 传输接口）受欢迎，以及为什么不能把 OSI（Open Systems Interconnection，开放系统互连）模型太当回事。我们还会看到 TCP 是一个性能优越、效率极高的协议，用 UDP 来重现它的功能通常是毫无意义的。

在第 2 章中我们还为几个 TCP/IP 应用模型开发了框架代码，并用它构建了一个常用函数库。有了这些框架和库，我们在编写应用程序时就不用考虑地址转换、连接管理等繁琐的常规问题了，这是非常重要的。正是因为有了这些框架，我们才不太会去走捷径，比如把地址和端口号硬编码到程序中，忽略返回错误等。

在本书中我们不断地使用这些框架和库来构建测试用例、示例代码、甚至能够独立工作的应用程序。在某个框架中添加几行代码常常就能构建出一个特定用途的应用程序或测试用例。

第 3 章对几个看起来很琐碎的问题进行了深入的研究。比如，一开始我们讨论了 TCP 的写操作以及它做了什么事情。乍一看这个问题好像很简单：我们写入 n 字节，TCP 将其发送到对等实体。但正如我们会看到的那样，通常，情况不是这样的。TCP 有一系列复杂的规则来决定它是否会在收到写操作时立即将数据发送出去，以及如果立即发送的话，应该发送多少。要想编写出健壮、高效的程序，理解这些规则以及它们怎样与应用程序交互是非常必要的。

读取数据和终止连接时也要考虑类似的问题。我们对这些操作进行了研究，并学习了如何进行有序终止以确保没有数据丢失。我们还对 connect 操作进行了研究，包括如何使其超时，以及如何将其用于 UDP 应用程序等。

我们研究了 UNIX 超级服务器 inetd 的用法，以及为什么使用这个程序可以大大降低编写网络感知（network-aware）程序的难度。同时，了解了如何使用 tcpmux 来降低为服务器分配知名端口的复杂性。我们说明了 tcpmux 是如何工作的，并且构建了自己的程序版本，以运行在不具备此功能的系统上。

我们对一些大家知之甚少的主题，比如 TIME-WAIT 状态、Nagle 算法、缓冲区大小的选择以及套接字选项 SO_REUSEADDR 的正确用法进行了深入的讨论。还讲解了如何使应用程序变成事件驱动的，以及如何为每个事件提供独立的定时器。我们对一些即使是经验丰富的网络程序员都常犯的错误，以及一些能够提高应用程序性能的技巧进行了研究。

最后，介绍了一些网络编程和脚本语言。通过一些 Perl 脚本展示了如何快速、方便地构建实用的网络应用程序和测试驱动程序。

第 4 章涉及两个领域。第一部分研究了几种每个网络程序员都必须掌握的工具。首先介绍了令人尊敬的 ping 程序，以及如何用它来进行基本的错误分析。接下来概括地介绍了网络嗅探器，并详细介绍了 tcpdump，用 tcpdump 诊断应用程序问题和难点的例子贯穿了第 4 章。我们还研究了 traceroute，并用它探查了一小部分因特网的情况。

ttcp 程序（由 ping 的开发者 Mike Muuss 与他人合作开发）是个很有用的工具，可以用来研究网络性能以及某些特定的 TCP 参数对性能的影响。我们用它展示了一些诊断技术。另一种工具 lsof 也非常有用，它可以将网络连接与打开网络连接的进程匹配起来。很多时候，lsof 提供的信息是通过其他途径很难得到的。

我们花了很大的篇幅来讨论 netstat 程序及其提供的很多不同类型的信息。还对 ktrace

和 truss 这样的系统调用跟踪程序进行了研究。

我们构建了一个侦听并显示 ICMP（Internet Control Message Protocol，因特网控制报文协议）数据报的程序，并以此结束了对网络诊断工具的讨论。这个程序不仅是对我们工具箱的有效补充，而且是一个使用原始套接字的实例。

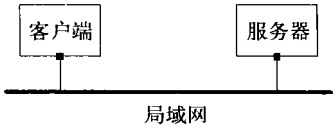
第4章的第二部分讨论了一些可以进一步拓展和加深我们对 TCP/IP 和网络编程理解的资源。我们介绍了 Richard Stevens 的几本经典著作，一些可以进行研究并从中学习的网络源代码，以及可以从 IETF（Internet Engineering Task Force，因特网工程任务组）和 Usenet 新闻组中找到的 RFC（Request For Comments，请求注释）文档。

1.3　客户端-服务器结构

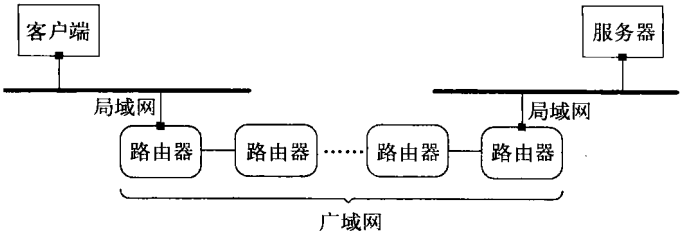
尽管我们经常谈及客户端（client）和服务器（server），但通常情况下，一个特定程序扮演的角色并不总是非常清晰的。通常这些程序更像对等实体，它们相互交换信息，而不是很明确地由服务器将信息提供给客户端。然而，TCP/IP 使这种区别明确起来。服务器监听来自一个或多个客户端的 TCP 连接或未经请求的 UDP 数据报。从客户端的角度来看，可以说是客户端先“开口说话”的。

如图 1-1 所示，本书考虑了三种典型的客户端-服务器配置情况。第一种情况如图 1-1A 所示，客户端和服务器运行在同一台机器上。这种配置不包含物理网络，因此是最简单的一种。输出数据照常沿着 TCP/IP 协议栈向下传送，但数据会被内部环回，像输入数据一样传回协议栈顶，不会被放到网络设备输出队列中去。

（A）客户端和服务器位于同一台主机上



（B）客户端和服务器位于同一个局域网内



（C）客户端和服务器位于不同的局域网中

图 1-1　典型的客户端-服务器配置情况

即使客户端和服务器最终运行在不同机器上，在开发过程中采用这种配置也有几个优点。首先，由于没有网络时延，采用这种配置更容易判断客户端和服务器应用程序的原始性能。第二，这种方法提供了一种理想的实验环境，分组不会被丢弃、延迟，传输时也不会失序。

> 至少在大多数情况下是这样的。在技巧 7 中可以看到，即使在这种环境中，施加足够的压力也会造成 UDP 报文丢失。

5

最后，在同一台机器上调试时，开发工作通常会变得更简单，更方便。

当然，即使在产品环境中，客户端和服务器也可能运行在同一台机器上。技巧 26 中有一个这样的例子。

如图 1-1B 所示，第二种客户端-服务器设置是客户端和服务器运行在同一个局域网中不同的机器上。这种配置中包含了一个实际网络，但这种环境还是近乎理想的。分组很少丢失，并且实际上从不会出现错序的情况。这是一种很常见的产品环境，大多数情况下应用程序都不会运行在其他环境中。

这种情况的典型实例就是打印服务器。一个小型局域网可能为多台主机只配置一台打印机。其中一台主机（或者一台内建了 TCP/IP 协议栈的打印机）作为服务器使用，从位于其他主机的客户端接收打印请求，并将数据假脱机到打印机中进行打印。

第三种类型的客户端-服务器配置中，客户端和服务器被广域网隔开（如图 1-1C 所示）。WAN 可以是因特网，也可以是公司内部网络，关键是这两个应用程序不在同一个局域网中。从一个应用程序传向另一个应用程序的 IP 数据报必须经过一台或多台路由器。

这种环境显然要比前两种更容易出问题。随着 WAN 流量的增加，用来在转发分组之前临时存储分组的路由器队列会被填满。路由器的队列空间耗尽时就开始丢弃分组。这就会导致重传，重传又会导致分组的重复和乱序传输。在技巧 38 中可以看到，这些不仅仅是理论上的问题，而且也并不少见。

技巧 12 更详细地介绍了局域网和广域网环境的区别，现在只需要知道它们的行为可能有很大差异就可以了。

1.4 对基本套接字 API 的回顾

本节回顾基本套接字 API，并用它构建基本的客户端和服务器应用程序。尽管这些应用程序只是些“骨架子”，但可以用来说明 TCP 客户端和服务器的一些基本特性。

首先介绍一个简单的客户端所需的 API 调用。图 1-2 显示了每个客户端都要用到的基本套接字调用。如图 1-2 所示，对等实体的地址是在传送给 connect 的 sockaddr_in 结构中指定的。

通常我们要做的第一件事就是获取连接的套接字。可以用 socket 系统调用来实现。

```
#include <sys/socket.h>     /* UNIX */
#include <winsock2.h>       /* Windows */

SOCKET socket( int domain, int type, int protocol );
```

返回：成功时返回套接字描述符，失败时返回-1（UNIX 系统）或者 INVALID_SOCKET（Windows）

6

套接字API与协议无关，可以支持几个不同的通信域（communications domain）。*domain*参数是一个常量，用来表示所期望的通信域。

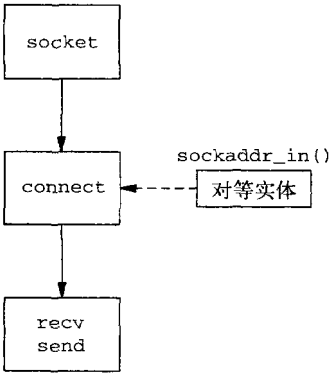


图1-2　客户端的基本套接字调用

最常见的两个域是AF_INET（也就是因特网）和AF_LOCAL（或AF_UNIX）。本书只关心AF_INET域。AF_LOCAL域用于同一台机器上的IPC（InterProcess Communication，进程间通信）。

> 对于*domain*常量应该用AF_*还是PF_*有过一场温和的论战。PF_*的支持者强调了4.1c/2.8/2.9BSD中现已无效的socket调用版本历史，以及PF表示协议族（protocol family）这个事实。AF_*的支持者则指出内核套接字代码将*domain*参数和AF_*常量匹配了起来。这两组常量的定义是一样的——实际上，其中一种经常会参照另一种的定义——因此，使用哪一种都没什么实质的区别。

参数*type*说明了要创建的套接字类型。最常见的，以及本书中用到的值如下所示。

- SOCK_STREAM——这些套接字提供了一个可靠的、全双工、面向连接的字节流。在TCP/IP中，就表示TCP。
- SOCK_DGRAM——这些套接字提供的是一种不可靠，尽力而为的数据报服务。在TCP/IP中，就表示UDP。
- SOCK_RAW——这些套接字允许对IP层的某些数据报进行访问。可用于一些特殊目的，比如监听ICMP报文。

7

*protocol*字段说明了应该在套接字上使用哪种协议。对TCP/IP来说，这个字段通常都由套接字类型隐式说明，参数被设置为零。在某些情况下，比如对原始套接字来说，有几种可能的协议，就要指定希望使用的协议。技巧40中有一个这样的例子。

对最简单的TCP客户端来说，与对等实体建立会话时需要使用的其他套接字API只有connect一种，connect是用来建立连接的，下面是一个例子。