计 算 机 科 学 丛 书

TCP/IP详解

卷3：TCP事务协议、HTTP、NNTP和UNIX域协议

TCP/IP Illustrated

Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols

(美) W. Richard Stevens 著
胡谷雨 吴礼发 等译
谢希仁 校

机械工业出版社
China Machine Press
Addison-Wesley

计算机科学丛书

TCP/IP详解 卷3: TCP事务协议、HTTP、NNTP和UNIX域协议

(美) W. Richard Stevens 著
胡谷雨 吴礼发 等译
谢希仁 校

机械工业出版社
China Machine Press

本书是“TCP/IP详解系列”的延续。主要内容包括：TCP事务协议，即T/TCP，这是对TCP的扩展，使客户-服务器事务更快、更高效和更可靠；TCP/IP应用，主要是HTTP和NNTP；UNIX域协议，这些协议提供了进程之间通信的一种手段。当客户与服务器进程在同一台主机上时，UNIX域协议通常要比TCP/IP快一倍。本书同样采用了大量的实例和实现细节，并参考引用了卷2中的大量源程序。

本书适用于希望理解TCP/IP如何工作的人，包括编写网络应用程序的程序员以及利用TCP/IP维护计算机网络的系统管理员。

本书版权登记号：图字：01-1999-2857

图书在版编目(CIP)数据

TCP/IP详解　卷3：TCP事务协议、HTTP、NNTP和UNIX域协议/(美)史蒂文斯(Stevens, W. R.)著；胡谷雨等译. -北京：机械工业出版社，2000.9

（计算机科学丛书）

书名原文：TCP/IP Illustrated, Volume 3: TCP for Transactions, HHTP, NNTP and the UNIX Domain Protocols

ISBN 7-111-07568-4

Ⅰ. T…　Ⅱ. ① 史…　② 胡…　Ⅲ. 计算机网络-传输控制协议　Ⅳ. TN915.04

中国版本图书馆CIP数据核字 (2000) 第36105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：陈贤舜

北京忠信诚胶印厂印刷 · 新华书店北京发行所发行

2000年9月第1版第1次印刷

787mm × 1092mm 1/16 · 16.75印张

印数：0 001-10 000册

定价：35.00元

译 者 序

我们愿意向广大的读者推荐W. Richard Stevens关于TCP/IP的经典著作(共3卷)的中译本。这里是其中的第3卷:《TCP/IP详解 卷3: TCP事务协议、HTTP、NNTP、UNIX域协议》。

大家知道，TCP/IP已成为计算机网络的事实上的标准。在关于TCP/IP的论著中，最有影响的就是两部著作。一部是Douglas E. Comer的《用TCP/IP进行网际互连》，一套共3卷(中译本已由电子工业出版社于1998年出版)，而另一部就是Stevens写的这3卷书。这两套巨著都很有名，各有其特点。无论是从事计算机网络的教师还是进行科研的技术人员，这两套书都应当是必读的。

本书的特点是内容丰富，概念清楚且准确，讲解详细，例子很多。作者在书中举出的所有例子均在作者安装的计算机网络上经过实际验证。在本书的最后，作者给出了许多经典的参考文献，并一一写出评注。

第3卷是第1、2卷的继续和深入。读者在学习这一卷时，应当先具备第1卷和第2卷所阐述的、关于TCP/IP的基本知识和实现知识。本卷仍然采用了大量的源代码来讲述协议及其应用的实现，并且本卷使用的一部分源代码是对第1卷和第2卷中有关源代码的修改，需要对照参考。这些内容对于编写TCP/IP网络应用程序的程序员和研究TCP/IP的计算机网络研究人员是非常有用的。

本卷书的前言由胡谷雨翻译，第1~5章由胡谷雨、马春华翻译，第6~12章由胡谷雨、张晖翻译，第13~15章由吴礼发、李旺翻译，第16~18章由吴礼发、金凤林翻译，附录由胡谷雨翻译。全书由谢希仁进行校阅。

限于水平，翻译中不妥或错误之处在所难免，敬请广大读者批评指正。

译 者

2000年5月于解放军理工大学，南京

译、校者介绍

谢希仁，中国人民解放军理工大学(南京)计算机系教授，全军网络技术研究中心主任，博士研究生导师，1952年毕业于清华大学电机系电信专业。所编写的《计算机网络》于1992年获全国优秀教材奖。1999年再版的《计算机网络》第2版为普通高等教育“九五”国家级重点教材。近来还主持翻译了Comer写的《TCP/IP网际互联》计算机网络经典教材一套三卷本(电子工业出版社1998年出版)，Harnedy写的《简单网络管理协议教程》(电子工业出版社1999年出版)。

胡谷雨，1963年出生，现为中国人民解放军理工大学(南京)指挥自动化学院计算机系教授、博士研究生导师，全军网络技术研究中心副主任。1983年于浙江大学无线电系获学士学位，并分别于1989和1992年在中国人民解放军通信工程学院计算机网络专业获硕士和博士学位。曾参加和负责了多项军队和863计划的重点科研项目，独立编著出版了《现代通信网和计算机网管理》(电子工业出版社，1996)，在《电子学报》、《通信学报》等刊物上发表论文多篇。目前主要从事计算机网络、通信网管理等方面的教学和研究工作。

吴礼发，1968年出生，现为中国人民解放军理工大学(南京)博士后。分别于1991和1995年在中国人民解放军通信工程学院获学士和硕士学位。1998年10月在南京大学获博士学位。曾作为主要成员参加过多项军队和地方重要科研项目，在《中国科学》、《计算机学报》、《通信学报》和《软件学报》等刊物上发表论文10余篇。目前主要从事计算机网络管理、VSAT卫星通信系统的网控系统和软件工程等方面的教学和研究工作。

对《TCP/IP详解 卷3：TCP事务协议、HTTP、NNTP和UNIX域协议》的评语

“是绝对值得一读的实例，它说明了如何将科学的思想方法和分析方法应用于实际的技术问题……它体现了技术写作和思考的最高水平”。

——Marcus J. Ranum，防火墙设计师

“是继既清楚又准确的系列卓越标准之后又一个杰出后继。该书的内容覆盖了T/TCP和HTTP，剖析了WWW，特别及时。”

——Vern Paxson，劳伦斯伯克利国家实验室网络研究小组

“该书内容中对HTTP的介绍对需要理解Web服务器行为细节的任何人都是无价之宝。”

——Jeffrey Mogul，数字设备公司

“卷3是对该书系列的自然补充，包括了Web服务中的网络技术和TCP事务传输的深入介绍。”

——Pete Haverlock，程序管理员，IBM

“在这TCP/IP详解的最后一卷中，Rich Stevens保持了他在前两卷中给自己设定的高标准：清楚的表达和准确的技术细节。”

——Andras Olah，Twente大学

“这一卷保持了该系列书前几卷中的极高质量，在新的方向上扩充了对网络实现技术的深入介绍。该书整个系列对于渴望了解当今Internet如何工作的任何人都是不可不读的。”

——Ian Lance Taylor，《GNU/Talyor UUCP》的作者

前　言

引言和本书的组织

本书是“TCP/IP详解系列”的延续：此系列的卷1是[Stevens 1994]，卷2是[Wright and Stevens 1995]。本书分成三个部分，每个部分覆盖了不同的内容。

(1) TCP事务协议，通常叫做T/TCP。这是对TCP的扩展，其设计目的是使客户-服务器事务更快、更高效和更可靠。这个目标的实现省略了连接开始时TCP的三次握手，并缩短了连接结束时TIME_WAIT状态的持续时间。我们将会看到，在客户-服务器事务中，T/TCP的性能与UDP相当，而且T/TCP具有可靠性和适应性，这两点相对UDP来说都是很大的改进。

事务是这样定义的：一个客户向服务器发出请求，接下来是服务器给出响应(这里的名词“事务”(transaction)并非数据库中的事务处理，数据库中的事务处理有封锁、两步提交和回退)。

(2) TCP/IP应用，特别是HTTP(超文本传送协议，WWW的基础)和NNTP(网络新闻传送协议，Usenet新闻系统的基础)。

(3) Unix域协议。这些协议是所有Unix中的TCP/IP实现中都提供的，在许多非Unix的实现中也都提供。这些协议提供了进程之间通信(IPC)的一种手段，采用了与TCP/IP中一样的插口接口。当客户与服务器进程在同一主机上时，Unix域协议通常要比TCP/IP快一倍。

第一部分，即对T/TCP的介绍，又分成两个小部分。第1~4章介绍协议，并给出了大量实例来说明它们是怎样工作的。这些材料主要是对卷1中24.7节的补充，在那里对T/TCP只是做了简单的介绍。第2小部分，即第5~12章，介绍T/TCP在4.4BSD-Lite网络代码(即，卷2中给出的代码)中的确切实现。由于最早的T/TCP实现迟至1994年9月才发布，已经是本书卷1出版一年以后了，那时卷2也快完成了，因此T/TCP的详细叙述，包括诸多实例和所有的实现细节都只好放在本系列书的卷3中了。

第二部分，即HTTP和NNTP应用，是卷1的第25~30章中介绍TCP/IP应用的延续。在卷1出版后的两年里，随着Internet的发展，HTTP得到了极大的流行，而NNTP的使用则在最近的10多年中每年增长了大约75%。T/TCP对HTTP来说也是非常好的，可以这样来用TCP：在少量数据传输中缩短连接时间，因为这种时候连接的建立和拆除时间往往占总时间的大头。在繁忙的Web服务器上，成千上万个不同而且不断变化的客户对HTTP(因此也对TCP)的高负荷使用，也提供了唯一可以对服务器上确切的分组进行考查的机会(第14章)，可以观察卷1和卷2中给出的TCP/IP的许多特性。

第三部分中的Unix域协议原本是准备在卷2中介绍的，但由于卷2已多达1200页[⊖]而删

⊖ 指原书英文版。——编者注

去了。在书名为《TCP/IP详解》这样的系列书中夹杂着TCP/IP以外的协议不免令人生奇，但Unix域协议几乎15年前就已经伴随着BSD版TCP/IP的实现在4.2BSD中发布了。今天，它们在任何一个从伯克利衍生而来的内核中都在频繁地使用，但它们的使用往往“被掩盖在后台”，大多数用户不知道它们的存在。除了在从伯克利衍生而来的内核中充当Unix管道的基础外，它们的另一个大用户是当客户程序和服务器程序在同一主机(典型的情况是工作站)上时的X Window系统。Unix域的插口也用于进程之间传递描述符，也是进程之间通信的一个强大工具。由于Unix域协议所用的插口API(应用编程接口)与TCP/IP所用的插口API几乎是相同的，Unix域协议以最小的代码变化提供了一个简单的手段来增强本地应用的性能。

以上三个部分的每个部分都可以独立阅读。

读者

与本系列书的前两卷一样，这一卷是为所有想要理解TCP/IP如何工作的人写的：编写网络应用的程序员，负责维护采用TCP/IP的计算机和网络的系统管理员，以及在日常工作中经常与TCP/IP应用程序打交道的用户。

第一和第二部分是理解TCP/IP工作原理的基础。不熟悉TCP/IP的读者应该看看本系列书的卷1，见[Stevens 1994]，以便对TCP/IP协议集有一个全面的了解。第一部分的前半部分(第1~4章，TCP/IP中的概念和例子)与卷2无关，可以直接阅读。但后半部分(第5~12章，T/TCP的实现)则需要先熟悉4.4 BSD-Lite网络程序，这些内容在卷2中介绍。

在整本书中有大量的向前和向后参考索引，这些参考索引是针对本书的两个主题，以及对卷1和卷2的内容，为想要了解更详细内容的读者提供的。在本书最后有书中用到的所有缩略语，封底背面则有书中介绍的所有结构、函数和宏(以字母顺序排列)及其介绍起始页码的交叉参考列表。如果本书引用了卷2中的定义，则该交叉参考列表也列出了卷2中的定义。

源码版权

本书中引自4.4BSD-Lite版的所有源码(源程序)都包括有下面这样的版权说明：

```
/*
 * Copyright (c) 1982, 1986, 1988, 1990, 1993, 1994
 *      The Regents of the University of California.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in the
 *    documentation and/or other materials provided with the distribution.
 * 3. All advertising materials mentioning features or use of this software
 *    must display the following acknowledgement:
 *      This product includes software developed by the University of
 *      California, Berkeley and its contributors.
 * 4. Neither the name of the University nor the names of its contributors
```

```
 *    may be used to endorse or promote products derived from this software
 *    without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */
```

第6章路由表的源码则包括下面这样的版权说明:

```
/*
 * Copyright 1994, 1995 Massachusetts Institute of Technology
 *
 * Permission to use, copy, modify, and distribute this software and
 * its documentation for any purpose and without fee is hereby
 * granted, provided that both the above copyright notice and this
 * permission notice appear in all copies, that both the above
 * copyright notice and this permission notice appear in all
 * supporting documentation, and that the name of M.I.T. not be used
 * in advertising or publicity pertaining to distribution of the
 * software without specific, written prior permission.  M.I.T. makes
 * no representations about the suitability of this software for any
 * purpose.  It is provided "as is" without express or implied
 * warranty.
 *
 * THIS SOFTWARE IS PROVIDED BY M.I.T. ``AS IS''.  M.I.T. DISCLAIMS
 * ALL EXPRESS OR IMPLIED WARRANTIES WITH REGARD TO THIS SOFTWARE,
 * INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
 * MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT
 * SHALL M.I.T. BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
 * SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
 * LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF
 * USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
 * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
 * OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
 */
```

印刷惯例

当需要显示交互的输入和输出信息时，将用黑体表示键盘输入，而计算机输出则用Courrier体，并用中文宋体作注释。

```
sun % telnet www.aw.com 80  连接到HTTP服务器
Trying 192.207.117.2...     本行和下一行由Telnet服务器输出
Connected to aw.com.
```

书中总是把系统名作为命令解释程序提示符的一部分(例如sun)，以说明命令是在哪个主机上执行的。在正文中引用的程序名通常都是首字母大写(如Telnet和Tcpdump)，以避免过多的字体形式。

在整个这本书中，我们将用像上述这段文字一样的缩进、附加的注释来说明实现细节中的历史阶段。

W. Richard Stevens
Tucson, Arizona
1995年11月
rstevens@noao.edu
http://www.noao.edu/~rstevens

目　录

第二部分 TCP的其他应用

第三部分 Unix域协议

第一部分 TCP事务协议

第1章 T/TCP 概 述

1.1 概述

本章首先介绍客户-服务器事务概念。我们从使用UDP的客户-服务器应用开始，这是最简单的情形。接着我们编写使用TCP的客户和服务器程序，并由此考察两台主机间交互的TCP/IP分组。然后我们使用T/TCP，证明利用T/TCP可以减少分组数，并给出为利用T/TCP需要对两端的源代码所做的最少改动。

接下来介绍了运行书中示例程序的测试网络，并对分别使用UDP、TCP和T/TCP的客户-服务器应用程序进行了简单的时间耗费比较。我们考察了一些使用TCP的典型Internet应用程序，看看如果两端都支持T/TCP，将需要做哪些修改。紧接着，简要介绍了Internet协议族中事务协议的发展历史，概略叙述了现有的T/TCP实现。

本书全文以及有关T/TCP的文献中，事务一词的含义都是指客户向服务器发出一个请求，然后服务器对该请求作出应答。Internet中最常见的一个例子是，客户向域名服务器(DNS)发出请求，查询域名对应的IP地址，然后域名服务器给出响应。本书中的事务这个术语并没有数据库中的事务那样的含义：加锁、两步提交、回退，等等。

1.2 UDP上的客户-服务器

我们先来看一个简单的UDP客户-服务器应用程序的例子，其客户程序源代码如图1-1所示。在这个例子中，客户向服务器发出一个请求，服务器处理该请求，然后发回一个应答。

udpcli.c

```
 1 #include    "cliserv.h"

 2 int
 3 main(int argc, char *argv[])
 4 {                                   /* simple UDP client */
 5     struct sockaddr_in serv;
 6     char    request[REQUEST], reply[REPLY];
 7     int     sockfd, n;

 8     if (argc != 2)
 9         err_quit("usage: udpcli <IP address of server>");

10     if ((sockfd = socket(PF_INET, SOCK_DGRAM, 0)) < 0)
11         err_sys("socket error");

12     memset(&serv, 0, sizeof(serv));
13     serv.sin_family = AF_INET;
14     serv.sin_addr.s_addr = inet_addr(argv[1]);
15     serv.sin_port = htons(UDP_SERV_PORT);
```

图1-1 UDP上的简单客户程序

```
16      /* form request[] ... */
17      if (sendto(sockfd, request, REQUEST, 0,
18                 (SA) &serv, sizeof(serv)) != REQUEST)
19          err_sys("sendto error");
20      if ((n = recvfrom(sockfd, reply, REPLY, 0,
21                        (SA) NULL, (int *) NULL)) < 0)
22          err_sys("recvfrom error");
23      /* process "n" bytes of reply[] ... */
24      exit(0);
25  }
```

udpcli.c

图1-1 (续)

本书中所有源代码的格式都是这样。每一非空行前面都标有行号。正文中叙述某段源代码时，这段源代码的起始和结束行号标记于正文段落的左边，如下面的正文所示。有时这些段落前面会有一小段说明、对所描述的源代码进行概要说明。源代码段开头和结尾处的水平线标明源代码段所在的文件名。这些文件名通常都是指我们在1.9节中将介绍的4.4版BSD-Lite中发布的文件。

我们来讨论这个程序的一些有关特性，但不详细描述插口函数，因为我们假设读者对这些函数有一些基本的认识。关于插口函数的细节在参考书[Stevens 1990]的第6章中可以找到。图1-2给出了头文件cliserv.h。

1. 创建UDP插口

10-11 socket函数用于创建一个UDP插口，并将一个非负的插口描述符返回给调用进程。出错处理函数err_sys参见参考书[Stevens 1992]的附录B.2。这个函数可以接受任意数目的参数，但要用vsprintf函数对它们格式化，然后这个函数会打印出系统调用所返回的errno值所对应的Unix出错信息，然后终止进程。

2. 填写服务器地址

12-15 首先用memset函数将Internet插口地址结构清零，然后填入服务器的IP地址和端口号。为简明起见，我们要求用户在程序运行中通过命令行输入一个点分十进制数形式的IP地址(argv[1])。服务器端口号(UDP_SERV_PORT)在头文件cliserv.h中用#define定义，在本章的所有程序首部中都包含了该头文件。这样做是为了使程序简洁，并避免使调用gethostbyname和getservbyname函数的源代码复杂化。

3. 构造并向服务器发送请求

16-19 客户程序构造一个请求(只用一行注释来表示)，并用sendto函数将其发出，这样就有一个UDP数据报发往服务器。同样是为了简明起见，我们假设请求(REQUEST)和应答(REPLY)的报文长度为固定值。实用的程序应当按照请求和应答的最大长度来分配缓存空间，但实际的请求和应答报文长度是变化的，而且一般都比较小。

4. 读取和处理服务器的应答

20-23 调用recvfrom函数将使进程阻塞(即置为睡眠状态)，直至收到一个数据报。接着客户进程处理应答(用一行注释来表示)，然后进程终止。

由于recvfrom函数中没有超时机制，请求报文或应答报文中任何一个丢失都将造成该进程永久挂起。事实上，UDP客户-服务器应用的一个基本问题就是对现实世界中的此类错误缺少健壮性。在本节的末尾将对这个问题做更详细的讨论。

在头文件cliserv.h中，我们将SA定义为struct sockaddr*，即指向一般的插口地址结构的指针。每当有一个插口函数需要一个指向插口地址结构的指针时，该指针必须被置为指向一个一般性插口地址结构的指针。这是由于插口函数先于ANSI C标准出现，在80年代早期开发插口函数的时候，void*(空类型)指针类型尚不可用。问题是，“struct sockaddr*”总共有17个字符，这经常使这一行源代码超出屏幕(或书本页面)的右边界，因此我们将其缩写成为SA。这个缩写是从BSD内核源代码中借用过来的。

图1-2给出了在本章所有程序中都包含的头文件cliserv.h。

cliserv.h

```
 1 /* Common includes and defines for UDP, TCP, and T/TCP
 2  * clients and servers */

 3 #include    <sys/types.h>
 4 #include    <sys/socket.h>
 5 #include    <netinet/in.h>
 6 #include    <arpa/inet.h>
 7 #include    <stdio.h>
 8 #include    <stdlib.h>
 9 #include    <string.h>
10 #include    <unistd.h>
11 #define REQUEST 400             /* max size of request, in bytes */
12 #define REPLY   400             /* max size of reply, in bytes */

13 #define UDP_SERV_PORT   7777    /* UDP server's well-known port */
14 #define TCP_SERV_PORT   8888    /* TCP server's well-known port */
15 #define TTCP_SERV_PORT  9999    /* T/TCP server's well-known port */

16     /* Following shortens all the type casts of pointer arguments */
17 #define SA  struct sockaddr *

18 void    err_quit(const char *,...);
19 void    err_sys(const char *,...);
20 int     read_stream(int, char *, int);
```

cliserv..

图1-2 本章各程序中均包含的头文件cliserv.h

图1-3给出了相应的UDP服务器程序。

udpserv.

```
 1 #include    "cliserv.h"

 2 int
 3 main()
 4 {                               /* simple UDP server */
 5     struct sockaddr_in serv, cli;
 6     char    request[REQUEST], reply[REPLY];
 7     int     sockfd, n, clilen;

 8     if ((sockfd = socket(PF_INET, SOCK_DGRAM, 0)) < 0)
 9         err_sys("socket error");

10     memset(&serv, 0, sizeof(serv));
11     serv.sin_family = AF_INET;
12     serv.sin_addr.s_addr = htonl(INADDR_ANY);
13     serv.sin_port = htons(UDP_SERV_PORT);
```

图1-3 与图1-1的UDP客户程序对应的UDP服务器程序

```
14      if (bind(sockfd, (SA) &serv, sizeof(serv)) < 0)
15          err_sys("bind error");

16      for (;;) {
17          clilen = sizeof(cli);
18          if ((n = recvfrom(sockfd, request, REQUEST, 0,
19                            (SA) &cli, &clilen)) < 0)
20              err_sys("recvfrom error");

21          /* process "n" bytes of request[] and create reply[] ... */

22          if (sendto(sockfd, reply, REPLY, 0,
23                     (SA) &cli, sizeof(cli)) != REPLY)
24              err_sys("sendto error");
25      }
26  }
```

udpserv.c

图1-3 (续)

5. 创建UDP插口和绑定本机地址

8-15 调用socket函数创建一个UDP插口，并在其Internet插口地址结构中填入服务器的本机地址。这里本机地址设置为通配符(INADDR_ANY)，这意味着服务器可以从任何一个本机接口接收数据报(假设服务器是多宿主的，即可以有多个网络接口)。端口号设为服务器的知名端口(UDP_SERV_PORT)，该常量也在前面讲过的头文件cliserv.h中定义。本机IP地址和知名端口用bind函数绑定到插口上。

6. 处理客户请求

16-25 接下来，服务器程序就进入一个无限循环：等待客户程序的请求到达(recvfrom)，处理该请求(我们只用一行注释来表示处理动作)，然后发出应答(sendto)。

这只是最简单的UDP客户-服务器应用。实际中常见的例子是域名服务系统(DNS)。DNS客户(称作解析器)通常是一般客户应用程序(例如，Telnet客户、FTP客户或WWW浏览器)的一个部分。解析器向DNS服务器发出一个UDP数据报，查询某一域名对应的IP地址。服务器发回的应答通常也是一个UDP数据报。

如果观察客户向服务器发送请求时双方交换的分组，我们就会得到图1-4这样的时间系列，页面上时间自上而下递增。服务器程序先启动，其行为过程给在图1-4的右半部，客户程序稍后启动。

我们分别来看客户和服务器程序中调用的函数及其相应内核执行的动作。在对socket函数的两次调用中，上下紧挨着的两个箭头表示内核执行请求的动作并立即返回。在调用sendto函数时，尽管内核也立即返回，但实际上已经发出了一个UDP数据报。为简明起见，我们假设客户程序的请求和服务器程序的应答所生成的IP数据报的长度都小于网络的最大传输单元(MTU)，IP数据报不必分段。

在这个图中，有两次调用recvfrom函数使进程睡眠，直到有数据报到达才被唤醒。我们把内核中相应的例程记为sleep和wakeup。

最后，我们还在图中标出了事务所耗费的时间。图1-4的左侧标示的是客户端测得的事务时间：从客户发出请求到收到服务器的应答所经历的时间。组成这段事务时间的数值标在图的右侧：RTT + SPT，其中RTT是网络往返时间，SPT是服务器处理客户请求的时间。UDP客户-服务器事务的最短时间就是RTT + SPT。