

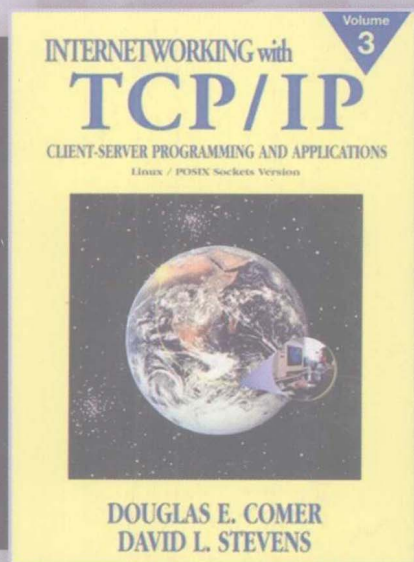
Douglas E. Comer

PEARSON

第三卷

# 用TCP/IP 进行网际互连

—— 客户-服务器编程与应用  
(Linux/POSIX套接字版)



Internetworking

With TCP/IP

Volume  
III

Client-Server Programming  
and Applications

Linux/POSIX Sockets Version

英文版

[美] Douglas E. Comer 著  
David L. Stevens



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

国外计算机科学教材系列

# 用 TCP/IP 进行网际互连

## 第三卷——客户 - 服务器编程与应用

### ( Linux/POSIX 套接字版 )

#### ( 英文版 )

Internetworking With TCP/IP  
Volume III: Client-Server Programming and Applications  
Linux/POSIX Sockets Version

[ 美 ] Douglas E. Comer 著  
David L. Stevens

電子工業出版社.  
Publishing House of Electronics Industry  
北京 · BEIJING

## 内 容 简 介

本书是关于计算机网络的经典教材,是目前美国大多数大学所开设的计算机网络课程的主要参考书。目前国内外能见到的各种关于TCP/IP的书籍,其主要内容都参考了本书。本书的特点是强调原理,概念准确,深入浅出,内容丰富新颖。全书共分为三卷。第三卷主要讨论应用软件如何使用TCP/IP,重点研究了客户-服务器范例,并考察了分布式程序中的客户和服务,举例说明了各种设计,讨论了应用网关和隧道技术。

Original edition, entitled INTERNETWORKING WITH TCP/IP, Volume III: CLIENT-SERVER PROGRAMMING AND APPLICATIONS, LINUX/POSIX SOCKETS VERSION, 1E, 9780130320711 by DOUGLAS E. COMER, DAVID L. STEVENS, published by Pearson Education, Inc., publishing as Prentice Hall, Copyright © 2001 by Prentice Hall, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

China edition published by PEARSON EDUCATION ASIA LTD., and PUBLISHING HOUSE OF ELECTRONICS INDUSTRY, Copyright © 2009.

This edition is manufactured in the People's Republic of China, and is authorized for sale only in the mainland of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).

本书由 Pearson Education 培生教育出版亚洲有限公司授予电子工业出版社。未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

此版本仅限在中国大陆出版发行。

本书贴有 Pearson Education 培生教育出版集团激光防伪标签,无标签者不得销售。

版权贸易合同登记号 图字:01-2009-3937

### 图书在版编目(CIP)数据

用TCP/IP进行网际互连.第3卷,客户-服务器编程与应用:Linux/POSIX套接字版=Internetworking with TCP/IP. Vol. 3, Client-Server Programming and Applications, Linux/Posix Sockets Version: 英文/(美)科默(Comer, D. E.), (美)史蒂文(Stevens, D. L.)著.-北京:电子工业出版社,2009.8  
(国外计算机科学教材系列)

ISBN 978-7-121-09187-2

I. 用… II. ①科… ②史… III. ①计算机网络-通信协议-教材-英文 ②Linux操作系统-程序设计-教材-英文 IV. TN915.04 TP316.89

中国版本图书馆CIP数据核字(2009)第109629号

策划编辑:谭海平

责任编辑:许菊芳

印 刷:北京市天竺颖华印刷厂

装 订:三河市鑫金马印装有限公司

出版发行:电子工业出版社

北京市海淀区万寿路173信箱 邮编:100036

开 本:787×980 1/16 印张:39.5 字数:885千字

印 次:2009年8月第1次印刷

定 价:69.00元

凡所购买电子工业出版社的图书有缺损问题,请向购买书店调换;若书店售缺,请与本社发行部联系。联系及邮购电话:(010)88254888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010)88258888。

## 出版说明

21 世纪初的 5 至 10 年是我国国民经济和社会发展的关键时期,也是信息产业快速发展的关键时期。在我国加入 WTO 后的今天,培养一支适应国际化竞争的一流 IT 人才队伍是我国高等教育的重要任务之一。信息科学和技术方面人才的优劣与多寡,是我国面对国际竞争时成败的关键因素。

当前,正值我国高等教育特别是信息科学领域的教育调整、变革的重大时期,为使我国教育体制与国际化接轨,有条件的高等院校正在为某些信息学科和技术课程使用国外优秀教材和优秀原版教材,以使我国在计算机教学上尽快赶上国际先进水平。

电子工业出版社秉承多年来引进国外优秀图书的经验,翻译出版了“国外计算机科学教材系列”丛书,这套教材覆盖学科范围广、领域宽、层次多,既有本科专业课程教材,也有研究生课程教材,以适应不同院系、不同专业、不同层次的师生对教材的需求,广大师生可自由选择和自由组合使用。这些教材涉及的学科方向包括网络与通信、操作系统、计算机组织与结构、算法与数据结构、数据库与信息处理、编程语言、图形图像与多媒体、软件工程等。同时,我们也适当引进了一些优秀英文原版教材,本着翻译版本和英文原版并重的原则,对重点图书既提供英文原版又提供相应的翻译版本。

在图书选题上,我们大都选择国外著名出版公司出版的高校教材,如 Pearson Education 培生教育出版集团、麦格劳—希尔教育出版集团、麻省理工学院出版社、剑桥大学出版社等。撰写教材的许多作者都是蜚声世界的教授、学者,如道格拉斯·科默(Douglas E. Comer)、威廉·斯托林斯(William Stallings)、哈维·戴特尔(Harvey M. Deitel)、尤利斯·布莱克(Uyless Black)等。

为确保教材的选题质量和翻译质量,我们约请了清华大学、北京大学、北京航空航天大学、复旦大学、上海交通大学、南京大学、浙江大学、哈尔滨工业大学、华中科技大学、西安交通大学、国防科学技术大学、解放军理工大学等著名高校的教授和骨干教师参与了本系列教材的选题、翻译和审校工作。他们中既有讲授同类教材的骨干教师、博士,也有积累了几十年教学经验的老教授和博士生导师。

在该系列教材的选题、翻译和编辑加工过程中,为提高教材质量,我们做了大量细致的工作,包括对所选教材进行全面论证;选择编辑时力求达到专业对口;对排版、印制质量进行严格把关。对于英文教材中出现的错误,我们通过与作者联络和网上下载勘误表等方式,逐一进行了修订。

此外,我们还将与国外著名出版公司合作,提供一些教材的教学支持资料,希望能为授课老师提供帮助。今后,我们将继续加强与各高校教师的密切联系,为广大师生引进更多的国外优秀教材和参考书,为我国计算机科学教学体系与国际教学体系的接轨做出努力。

电子工业出版社

## 教材出版委员会

主 任	杨芙清	北京大学教授 中国科学院院士 北京大学信息与工程学部主任 北京大学软件工程研究所所长
委 员	王 珊	中国人民大学信息学院教授 中国计算机学会副理事长，数据库专业委员会主任
	胡道元	清华大学计算机科学与技术系教授 国际信息处理联合会通信系统中国代表
	钟玉琢	清华大学计算机科学与技术系教授、博士生导师 清华大学深圳研究生院信息学部主任
	谢希仁	中国人民解放军理工大学教授 全军网络技术研究中心主任、博士生导师
	尤晋元	上海交通大学计算机科学与工程系教授 上海分布计算技术中心主任
	施伯乐	上海国际数据库研究中心主任、复旦大学教授 中国计算机学会常务理事、上海市计算机学会理事长
	邹 鹏	国防科学技术大学计算机学院教授、博士生导师 教育部计算机基础课程教学指导委员会副主任委员
	张昆藏	青岛大学信息工程学院教授

# Foreword

It is a singular honor to introduce open source readers to the third volume of Dr. Douglas E. Comer's remarkable series: *Internetworking with TCP/IP*.

The history of open source and TCP/IP are magnificently intertwined: you can't have collaboration without a network to connect the collaborators! Further, some of the very first open source software were implementations of the TCP/IP protocols. I remember the early 80's, long before "open source" was today's media darling. In those days, there were a handful of researchers who understood the problems of network architectures and implementations. Doug was among their leaders — the principal of an extensive research program and waging a multi-front attack on the problems we faced.

I remember the early 90's, when we first began to see the push toward moving the technology to the large engineering communities that were hungering for knowledge and solutions. In those days, it was a mighty struggle for those engineers to build internet-based environments for their corporations. Doug was there to educate and inform them — making the underlying complexity accessible to them and providing them with hard-earned insights.

And now, I see the early 00's, where a new generation of designers are writing distributed applications for the Internet. In these days, we hear of many exciting Internet applications, such as napster, gnutella, and infrasearch. Surprisingly, few of today's developers have a grasp of solid network engineering principles — bluntly, they lack an understanding of the basics, and this lack of understanding inevitably leads to applications that don't scale well or that just plain do not work.

It is for this reason, that *Volume 3, Client-Server Programming and Applications*, which Doug has authored with David L. Stevens, is particularly relevant to the Internet today. It teaches us how to architect and build client-server applications, and — more importantly — how to understand what trade-offs are involved with each design decision.

My hope is that you, the reader, can benefit from Dr. Comer's wisdom as much as your humble predecessors.

Marshall T. Rose  
Theorist, Implementor, and Agent Provocateur  
Petaluma, California  
June, 2000

# Preface

The Linux operating system is soaring in popularity, and especially important in the networking community as the system for many servers. This new version of Volume 3, which uses Linux, is aimed at programmers who want to understand how to create networking applications. Broadly speaking, it examines the question, “How does application software use TCP/IP protocols to communicate across an internet?” The text focuses on the client-server paradigm, and examines algorithms for both the client and server components of a distributed program. It shows an implementation that illustrates each design, and discusses techniques including application-level gateways and tunneling. In addition, it reviews several standard application protocols, and uses them to illustrate the algorithms and implementation techniques.

Although this volume can be read and used alone, it forms part of a larger picture completed by two other volumes in the series. Volume 1 considers the question “What is a TCP/IP internet?” Volume 2 examines the question, “How does TCP/IP software work?” It presents more details, examines working code, and explores greater depth than the first volume. Thus, although a programmer can learn to create network applications from Volume 3 alone, the other volumes can be used to provide a better understanding of the underlying technologies.

This version of Volume 3 includes the latest technologies. For example, a chapter explains how a Linux program can use the POSIX thread facilities to create a concurrent server. The chapter on NFS discusses version 3, the version that is about to emerge in the Linux community. In addition, sections have been included to explain concepts behind programs like *slirp* that provide Internet access over a dialup telephone connection without requiring each computer to have a unique IP address.

Two chapters that stand out as especially timely concentrate on *streaming* and the associated technologies used to send audio and video across the Internet. Chapter 28 describes fundamental concepts such as the Real-time Transport Protocol (RTP), encoding, and jitter buffers. Chapter 29 shows an implementation of RTP that is used to receive and play MP3 audio.

The code for all examples in the text is available online. To access a copy via the Web, look for Volume 3 in the list of networking books at location:

<http://www.cs.purdue.edu/homes/comer/netbooks.html>

To access the code via FTP, use location:

<ftp://ftp.cs.purdue.edu/pub/Xinu/TCPIP-vol3.linux.dist.tar.Z>

The text is organized as follows. Beginning chapters introduce the client-server paradigm and the socket interface that application programs use to access TCP/IP protocol software. They also describe concurrent processes and the operating system functions used to create them. Chapters that follow the introductory material discuss client and server designs.

The text explains that the myriad of possible designs are not random. Instead, they follow a pattern that can be understood by considering the choice of concurrency and transport. For example, one chapter discusses a nonconcurrent server design that uses connection-oriented transport (e.g., TCP), while another discusses a similar design that uses connectionless transport (e.g., UDP).

We describe how each design fits into the space of possible implementations, but do not try to develop an abstract “theory” of client-server interactions. Instead, we emphasize practical design principles and techniques that are important to programmers. Each technique has advantages in some circumstances, and each has been used in working software. We believe that understanding the conceptual ties among the designs will help the reader appreciate the strengths and weaknesses of each approach, and will make it easier to choose among them.

The text contains example programs that show how each design operates in practice. Most of the examples implement standard TCP/IP application protocols. In each case, we tried to select an application protocol that would convey a single design idea without being too complex to understand. Thus, while few of the example programs are exciting, they each illustrate one important concept. This version of Volume 3 uses the Linux socket mechanism (i.e., socket API) in all programming examples; two other versions of the text contain many of the same examples using Microsoft’s Windows Sockets interface and AT&T’s TLI interface.

Later chapters focus on middleware. They discuss the remote procedure call concept and describe how it can be used to construct distributed programs. The chapters relate the remote procedure call technique to the client-server model, and show how software can be used to generate client and server programs from a remote procedure call description. The chapters on TELNET show how small details dominate a production program and how complex the code can become for even a simple, character-oriented protocol. The section ends with the two chapters on streaming transport described earlier.

Much of the text concentrates on concurrent processing. Many of the concepts described may seem familiar to students who have written concurrent programs because they apply to all concurrent programs, not only network applications. Students who have not written concurrent programs may find the concepts difficult.



The text is suitable for a single semester undergraduate course on “socket programming” or a beginning graduate-level course on distributed computing. Because the text concentrates on how to use an internet rather than on how it works, students need little background in networking to understand the material. No particular concept is too difficult for an undergraduate course as long as the instructor proceeds at a suitable pace. A basic course in operating systems concepts or experience with concurrent programming may provide the best background.

Students will not appreciate the material until they use it first hand. Thus, any course should have programming exercises that force the students to apply the ideas to practical programs. Undergraduates can learn the basics by repeating the designs on other application protocols. Graduate students should build more complex distributed programs that emphasize some of the subtle techniques (e.g., the concurrency management techniques in Chapter 16 and the interconnection techniques in Chapters 18 and 19).

We thank many people for their help. Members of the Internet Research Group at Purdue contributed technical information and suggestions to the original text. Michael Evangelista proofread the text and wrote the RTP code. Gustavo Rodriguez-Rivera read several chapters, ran experiments to test details, and edited Appendix 1. Dennis Brylow commented on several chapters. Christine Comer edited the entire text and improved both wording and consistency.

Douglas E. Comer

David L. Stevens

July, 2000

## About The Authors

Dr. Douglas Comer is an internationally recognized expert on TCP/IP protocols and the Internet. One of the researchers who contributed to the Internet as it was being formed in the late 1970s and 1980s, he was a member of the Internet Architecture Board, the group responsible for guiding the Internet's development. He was also chairman of the CSNET technical committee and a member of the CSNET executive committee.

Comer consults for companies on the design and implementation of networks, and gives professional seminars on TCP/IP and internetworking to both technical and nontechnical audiences around the world. His operating system, Xinu, and implementation of TCP/IP protocols are documented in his books, and used in commercial products.

Comer is a professor of computer science at Purdue University, where he teaches courses and does research on computer networking, internetworking, and operating systems. In addition to writing a series of best-selling technical books, he serves as the North American editor of the journal *Software — Practice and Experience*. Comer is a Fellow of the ACM.

Additional information can be found at:

[www.cs.purdue.edu/people/comer](http://www.cs.purdue.edu/people/comer)

David Stevens received his BS (1985) and MS (1993) in Computer Science from Purdue University. He has been a UNIX systems programmer working primarily on BSD UNIX kernels since 1983. He has done implementations of most of the Internet Protocol Suite and co-authored several Computer Science textbooks with Dr. Comer. His areas of professional interest are operating systems, computer networking, and large-scale software systems design.

In recent years, Stevens has worked in the area of scalable networking on high-performance multiprocessor systems for Sequent Computer Systems and the IBM Corporation. He is a member of the ACM and IEEE.

## What Others Have Said About The Linux Version Of Internetworking With TCP/IP Volume 3

“This is by far the best book on the topic I have *ever* read. Thank you for making sockets easy to understand.”

*Dustin Boswell  
Caltech*

“An excellent book for learning TCP/IP client-server programming. This book explains important concepts clearly and provides working example code; the combination produces an extremely effective way to learn the subject.”

*John Lin  
Bell Labs*

“Your book has been extremely valuable to me — thank you very much indeed.”

*Jacoby Thwaites*

“I enjoy the clarity and depth!”

*Rob Moloney*

“Volume 3, Client-Server Programming and Applications, which Doug has authored with David L. Stevens, is particularly relevant to the Internet today. It teaches us how to architect and build client-server applications, and — more importantly — how to understand what trade-offs are involved with each design decision.”

*Marshall Rose*

# Contents

<b>Foreword</b>	22
-----------------	----

<b>Preface</b>	23
----------------	----

<b>Chapter 1 Introduction And Overview</b>	<b>1</b>
--------------------------------------------	----------

1.1	<i>Internet Applications Using TCP/IP</i>	1
1.2	<i>Designing Applications For A Distributed Environment</i>	1
1.3	<i>Standard And Nonstandard Application Protocols</i>	2
1.4	<i>An Example Of Standard Application Protocol Use</i>	2
1.5	<i>An Example TELNET Connection</i>	3
1.6	<i>Using TELNET To Access An Alternative Service</i>	4
1.7	<i>Application Protocols And Software Flexibility</i>	5
1.8	<i>Viewing Services From The Provider's Perspective</i>	6
1.9	<i>The Remainder Of This Text</i>	6
1.10	<i>Summary</i>	7

<b>Chapter 2 The Client Server Model And Software Design</b>	<b>9</b>
--------------------------------------------------------------	----------

2.1	<i>Introduction</i>	9
2.2	<i>Motivation</i>	10
2.3	<i>Terminology And Concepts</i>	10
2.3.1	<i>Clients And Servers</i>	11
2.3.2	<i>Privilege And Complexity</i>	11
2.3.3	<i>Standard Vs. Nonstandard Client Software</i>	12
2.3.4	<i>Parameterization Of Clients</i>	12
2.3.5	<i>Connectionless Vs. Connection-Oriented Servers</i>	13
2.3.6	<i>Stateless Vs. Stateful Servers</i>	14

2.3.7	<i>A Stateless File Server Example</i>	15
2.3.8	<i>A Stateful File Server Example</i>	15
2.3.9	<i>Identifying A Client</i>	16
2.3.10	<i>Statelessness Is A Protocol Issue</i>	18
2.3.11	<i>Servers As Clients</i>	19
2.4	<i>Summary</i>	20

## **Chapter 3 Concurrent Processing In Client-Server Software**

**23**

3.1	<i>Introduction</i>	23
3.2	<i>Concurrency In Networks</i>	23
3.3	<i>Concurrency In Servers</i>	25
3.4	<i>Terminology And Concepts</i>	26
3.4.1	<i>The Process Concept</i>	26
3.4.2	<i>Sharing Of Local And Global Variables</i>	27
3.4.3	<i>Procedure Calls</i>	28
3.5	<i>An Example Of Concurrent Process Creation</i>	29
3.5.1	<i>A Sequential C Example</i>	29
3.5.2	<i>A Concurrent Version</i>	30
3.5.3	<i>Timeslicing</i>	31
3.5.4	<i>Singly-Threaded Process Assumption</i>	32
3.5.5	<i>Making Processes Diverge</i>	33
3.6	<i>Executing New Code</i>	34
3.7	<i>Context Switching And Protocol Software Design</i>	34
3.8	<i>Concurrency And Asynchronous I/O</i>	35
3.9	<i>Summary</i>	36

## **Chapter 4 Application Interface To Protocols**

**39**

4.1	<i>Introduction</i>	39
4.2	<i>Loosely Specified Protocol Software Interface</i>	39
4.2.1	<i>Advantages And Disadvantages</i>	40
4.3	<i>Interface Functionality</i>	40
4.4	<i>Conceptual Interface Specification</i>	41
4.5	<i>System Calls</i>	42
4.6	<i>Two Basic Approaches To Network Communication</i>	43
4.7	<i>The Basic I/O Functions Available In Linux</i>	43
4.8	<i>Using Linux I/O With TCP/IP</i>	45
4.9	<i>Summary</i>	45

5.1	<i>Introduction</i>	47
5.2	<i>Berkeley Sockets</i>	47
5.3	<i>Specifying A Protocol Interface</i>	48
5.4	<i>The Socket Abstraction</i>	49
5.4.1	<i>Socket Descriptors And File Descriptors</i>	49
5.4.2	<i>System Data Structures For Sockets</i>	50
5.4.3	<i>Making A Socket Active Or Passive</i>	51
5.5	<i>Specifying An Endpoint Address</i>	52
5.6	<i>A Generic Address Structure</i>	52
5.7	<i>Major System Calls In The Socket API</i>	54
5.7.1	<i>The Socket Call</i>	54
5.7.2	<i>The Connect Call</i>	54
5.7.3	<i>The Send Call</i>	55
5.7.4	<i>The Recv Call</i>	55
5.7.5	<i>The Close Call</i>	55
5.7.6	<i>The Bind Call</i>	56
5.7.7	<i>The Listen Call</i>	56
5.7.8	<i>The Accept Call</i>	56
5.7.9	<i>Using Read And Write With Sockets</i>	56
5.7.10	<i>Summary Of Socket Calls</i>	57
5.8	<i>Utility Routines For Integer Conversion</i>	58
5.9	<i>Using Socket Calls In A Program</i>	58
5.10	<i>Symbolic Constants For Socket Call Parameters</i>	59
5.11	<i>Summary</i>	60

6.1	<i>Introduction</i>	63
6.2	<i>Learning Algorithms Instead Of Details</i>	63
6.3	<i>Client Architecture</i>	64
6.4	<i>Identifying The Location Of A Server</i>	64
6.5	<i>Parsing An Address Argument</i>	66
6.6	<i>Looking Up A Domain Name</i>	67
6.7	<i>Looking Up A Well-Known Port By Name</i>	68
6.8	<i>Port Numbers And Network Byte Order</i>	68
6.9	<i>Looking Up A Protocol By Name</i>	69
6.10	<i>The TCP Client Algorithm</i>	69
6.11	<i>Allocating A Socket</i>	70
6.12	<i>Choosing A Local Protocol Port Number</i>	71
6.13	<i>A Fundamental Problem In Choosing A Local IP Address</i>	71
6.14	<i>Connecting A TCP Socket To A Server</i>	72

6.15	<i>Communicating With The Server Using TCP</i>	72
6.16	<i>Receiving A Response From A TCP Connection</i>	73
6.17	<i>Closing A TCP Connection</i>	74
6.17.1	<i>The Need For Partial Close</i>	74
6.17.2	<i>A Partial Close Operation</i>	74
6.18	<i>Programming A UDP Client</i>	75
6.19	<i>Connected And Unconnected UDP Sockets</i>	76
6.20	<i>Using Connect With UDP</i>	76
6.21	<i>Communicating With A Server Using UDP</i>	76
6.22	<i>Closing A Socket That Uses UDP</i>	77
6.23	<i>Partial Close For UDP</i>	77
6.24	<i>A Warning About UDP Unreliability</i>	77
6.25	<i>Summary</i>	77

## **Chapter 7 Example Client Software**

**81**

7.1	<i>Introduction</i>	81
7.2	<i>The Importance Of Small Examples</i>	81
7.3	<i>Hiding Details</i>	82
7.4	<i>An Example Procedure Library For Client Programs</i>	82
7.5	<i>Implementation Of ConnectTCP</i>	83
7.6	<i>Implementation Of ConnectUDP</i>	84
7.7	<i>A Procedure That Forms Connections</i>	85
7.8	<i>Using The Example Library</i>	88
7.9	<i>The DAYTIME Service</i>	88
7.10	<i>Implementation Of A TCP Client For DAYTIME</i>	89
7.11	<i>Reading From A TCP Connection</i>	90
7.12	<i>The TIME Service</i>	91
7.13	<i>Accessing The TIME Service</i>	91
7.14	<i>Accurate Times And Network Delays</i>	92
7.15	<i>A UDP Client For The TIME Service</i>	92
7.16	<i>The ECHO Service</i>	94
7.17	<i>A TCP Client For The ECHO Service</i>	94
7.18	<i>A UDP Client For The ECHO Service</i>	96
7.19	<i>Summary</i>	98

## **Chapter 8 Algorithms And Issues In Server Software Design**

**101**

8.1	<i>Introduction</i>	101
8.2	<i>The Conceptual Server Algorithm</i>	101
8.3	<i>Concurrent Vs. Iterative Servers</i>	102
8.4	<i>Connection-Oriented Vs. Connectionless Access</i>	102

8.5	<i>Transport Protocol Semantics</i>	103
8.5.1	<i>TCP Semantics</i>	103
8.5.2	<i>UDP Semantics</i>	103
8.6	<i>Choice Of Transport</i>	104
8.7	<i>Connection-Oriented Servers</i>	104
8.8	<i>Connectionless Servers</i>	105
8.9	<i>Failure, Reliability, And Statelessness</i>	106
8.10	<i>Optimizing Stateless Servers</i>	106
8.11	<i>Four Basic Types Of Servers</i>	109
8.12	<i>Request Processing Time</i>	109
8.13	<i>Iterative Server Algorithms</i>	110
8.14	<i>An Iterative, Connection-Oriented Server Algorithm</i>	110
8.15	<i>Binding To A Well-Known Address Using INADDR_ANY</i>	111
8.16	<i>Placing The Socket In Passive Mode</i>	112
8.17	<i>Accepting Connections And Using Them</i>	112
8.18	<i>An Iterative, Connectionless Server Algorithm</i>	112
8.19	<i>Forming A Reply Address In A Connectionless Server</i>	113
8.20	<i>Concurrent Server Algorithms</i>	114
8.21	<i>Master And Slaves</i>	114
8.22	<i>A Concurrent, Connectionless Server Algorithm</i>	115
8.23	<i>A Concurrent, Connection-Oriented Server Algorithm</i>	116
8.24	<i>Implementations Of Server Concurrency</i>	117
8.25	<i>Using Separate Programs As Slaves</i>	118
8.26	<i>Apparent Concurrency Using A Single Thread</i>	118
8.27	<i>When To Use Each Server Type</i>	119
8.28	<i>A Summary of Server Types</i>	120
8.29	<i>The Important Problem Of Server Deadlock</i>	121
8.30	<i>Alternative Implementations</i>	122
8.31	<i>Summary</i>	122

## **Chapter 9 Iterative, Connectionless Servers (UDP)**

**125**

9.1	<i>Introduction</i>	125
9.2	<i>Creating A Passive Socket</i>	125
9.3	<i>Process Structure</i>	129
9.4	<i>An Example TIME Server</i>	130
9.5	<i>Summary</i>	132

## **Chapter 10 Iterative, Connection-Oriented Servers (TCP)**

**135**

10.1	<i>Introduction</i>	135
10.2	<i>Allocating A Passive TCP Socket</i>	135



10.3	<i>A Server For The DAYTIME Service</i>	136
10.4	<i>Process Structure</i>	136
10.5	<i>An Example DAYTIME Server</i>	137
10.6	<i>Closing Connections</i>	140
10.7	<i>Connection Termination And Server Vulnerability</i>	140
10.8	<i>Summary</i>	141
<b>Chapter 11 Concurrent, Connection-Oriented Servers (TCP)</b>		<b>143</b>
11.1	<i>Introduction</i>	143
11.2	<i>ECHO Service</i>	143
11.3	<i>Iterative Vs. Concurrent Implementations</i>	144
11.4	<i>Process Structure</i>	144
11.5	<i>An Example Concurrent ECHO Server</i>	145
11.6	<i>Cleaning Up Errant Processes</i>	149
11.7	<i>Summary</i>	150
<b>Chapter 12 Using Threads For Concurrency (TCP)</b>		<b>151</b>
12.1	<i>Introduction</i>	151
12.2	<i>Overview Of Linux Threads</i>	151
12.3	<i>Advantages Of Threads</i>	152
12.4	<i>Disadvantages Of Threads</i>	153
12.5	<i>Descriptors, Delay, And Exit</i>	153
12.6	<i>Thread Exit</i>	154
12.7	<i>Thread Coordination And Synchronization</i>	154
12.7.1	<i>Mutex</i>	154
12.7.2	<i>Semaphore</i>	155
12.7.3	<i>Condition Variable</i>	155
12.8	<i>An Example Server Using Threads</i>	156
12.9	<i>Monitor And Control</i>	160
12.10	<i>Summary</i>	161
<b>Chapter 13 Single-Thread, Concurrent Servers (TCP)</b>		<b>163</b>
13.1	<i>Introduction</i>	163
13.2	<i>Data-driven Processing In A Server</i>	163
13.3	<i>Data-Driven Processing With A Single Thread</i>	164
13.4	<i>Process Structure Of A Single-Thread Server</i>	165
13.5	<i>An Example Single-Thread ECHO Server</i>	166
13.6	<i>Summary</i>	168