

大学计算机教育丛书(影印版)

网络互连技术

Second Edition

INTERNETWORKING WITH

TCP/IP

VOLUME III

CLIENT-SERVER PROGRAMMING
AND APPLICATIONS

BSD
Socket
Version
with
ANSI C

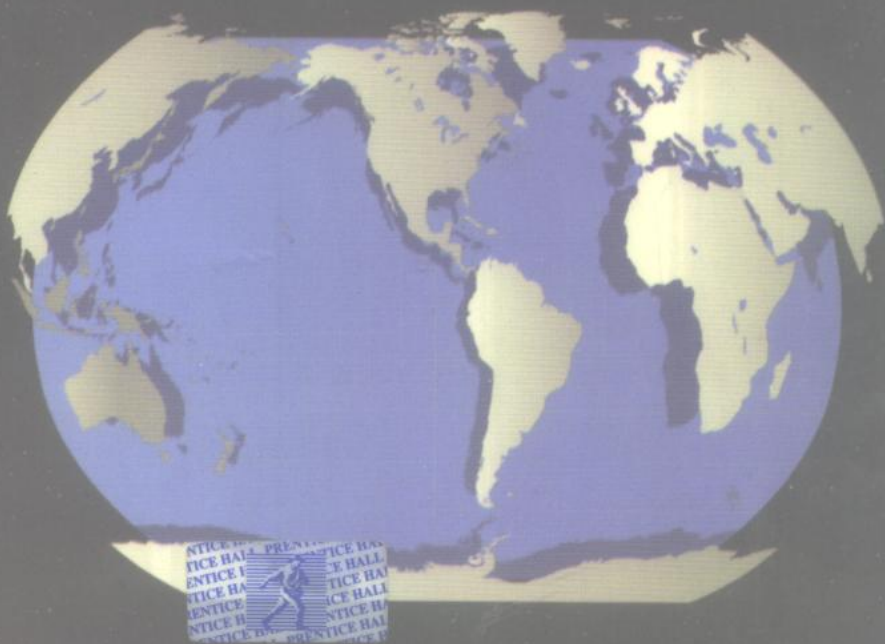
TCP/IP

网络互连技术

卷III：客户服务器编程和应用

BSD套接字版 第2版

Douglas E. Comer · David L. Stevens



清华大学出版社 · PRENTICE HALL

<http://www.tup.tsinghua.edu.cn>

7-17
Lafayette
2-3

Internetworking With TCP/IP

Vol III :

**Client-Server Programming And
Applications BSD Socket Version**

Second Edition

TCP/IP 网络互连技术

卷 III

客户/服务器编程和应用 BSD 套接字版

第 2 版

DOUGLAS E. COMER

DAVID L. STEVENS

Department of Computer Sciences

Purdue University

West Lafayette, IN 47907

清华大学出版

Prentice-Hall International, Inc.

7-302-118
(京)新登字158号

Internetworking with TCP/IP Vol III: Client-Server programming and applications, BSD Socket Version, 2nd ed./Douglas E. Comer, David L. Stevens

© 1996 by Prentice Hall, Inc.

Original edition published by Prentice Hall, Inc., a Simon & Schuster Company.

Prentice Hall 公司授权清华大学出版社在中国境内(不包括中国香港特别行政区、澳门地区和台湾地区)独家出版发行本书影印本。

本书任何部分之内容,未经出版者书面同意,不得用任何方式抄袭、节录或翻印。

本书封面贴有 Prentice Hall 激光防伪标签,无标签者不得销售。

北京市版权局著作权合同登记号:01-98-0959

图书在版编目(CIP)数据

TCP/IP 网络互连技术 卷Ⅲ:BSD 套接字版:第2版:英文/(美)科默(Comer, D. E.),史蒂文斯(Stevens, D. L.).—影印版.—北京:清华大学出版社,1998.7

(大学计算机教育丛书)

ISBN 7-302-02948-2

I. T… II. ①科… ②史… III. 计算机网络-连接技术-英文 IV. TP393

中国版本图书馆 CIP 数据核字(98)第 09324 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

<http://www.tup.tsinghua.edu.cn>

印刷者:清华大学印刷厂

发行者:新华书店总店北京科技发行所

开本:850×1168 1/32 印张:17.125

版次:1998年9月第1版 1998年12月第2次印刷

书号:ISBN 7-302-02948-2/TP·1559

印数:3001~8000

定价:32.00 元

出版前言

清华大学出版社与 Prentice Hall 出版公司合作推出的“大学计算机教育丛书(影印版)”和“ATM 与 B-ISDN 技术丛书(影印版)”受到了广大读者的欢迎。很多读者通过电话、信函、电子函件给我们的工作以积极的评价,并提出了不少中肯的建议。其中,很多读者希望我们能够出版一些网络方面较深层次的书籍,这也就成为我们出版这套“网络互连技术系列”的最初动机。

众所周知,网络协议是网络与通信技术的关键组成部分。而今,因特网技术、移动通信技术的飞速发展,为网络协议注入了新内容。本套丛书以 Douglas Comer 教授的网络协议的经典名著 TCP/IP 网络互连技术系列为主干,并补充以论述新协议如 IPV6 和移动 IP 等国外最新专著,力求为从事网络互连技术与开发的人员以及大专院校师生提供充分的技术支持。

衷心希望所有阅读这套丛书的读者能从中受益。

清华大学出版社
Prentice Hall 公司

1998.9

Foreword

It is indeed a pleasure to introduce the reader to the revised third volume of Dr. Douglas E. Comer's remarkable series: *Internetworking with TCP/IP*. This series, which began so innocently back in 1987, is now the premiere source for learning about the suite of protocols that have made vendor-independent computer-communications possible – the Internet suite of protocols.

To my mind, this seminal work is our best hope against the “dumbing down of the Internet.” Whilst the media and entrepreneurs fill the popular imagination with visions of “Internet mysticism,” it is Dr. Comer who clearly explains the technical reality of the technology that makes the Internet possible.

Although I have learned from all three books in the series, I feel 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.

So, I invite you to undertake a memorable journey into “how’s and why’s” of the theory, design, and realization of internetworking technology.

Marshall T. Rose
Theorist, Implementor, and Agent Provocateur
Del Mar, California

Preface

We are pleased to introduce a revised version of Volume 3 in the Internetworking Series. Broadly speaking, Volume 1 examines the question, “What is a TCP/IP internet?” Volume 2 examines the question, “How does TCP/IP software work?” It presents more details and explores greater depth than the first volume. This volume examines the question, “How does application software use TCP/IP?” It 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.

The revision follows the latest standards. For example, code in examples has been rewritten to use ANSI C, and the chapter on NFS discusses changes in version 3. In addition, new sections have been added to explain concepts behind programs like *slirp* that provide Internet access across a dialup telephone connection. The discussion of ways client-server systems fail has been expanded: an entire new chapter focuses on deadlock and livelock. The chapter examines causes of the problems and techniques for preventing them. Finally, minor typos and ambiguities in wording have been corrected throughout the text.

The code is available on-line. 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/books.html>

To access the code via FTP, use location:

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

The organization of the text remains the same as the previous version. 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 BSD UNIX socket mechanism in all programming examples; a companion edition contains the same examples using AT&T's TLI protocol interface.

Later chapters discuss the remote procedure call concept and describe how it can be used to construct distributed programs. They 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.

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 introductory networking course at the senior or graduate level. 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 lower level courses 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 15 and the interconnection techniques in Chapter 17).

Many people deserve credit for their help. Members of the Internet Research Group at Purdue contributed technical information and suggestions to the original text. Christine Comer edited the revision and improved both wording and consistency.

Douglas E. Comer

David L. Stevens

Contents

Foreword **xxiii**

Preface **xxv**

Chapter 1 Introduction And Overview **1**

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

Chapter 2 The Client Server Model And Software Design **9**

- 2.1 Introduction* 9
- 2.2 Motivation* 10
- 2.3 Terminology And Concepts* 10
 - 2.3.1 Clients And Servers* 10
 - 2.3.2 Privilege And Complexity* 11
 - 2.3.3 Standard Vs. Nonstandard Client Software* 11
 - 2.3.4 Parameterization Of Clients* 12
 - 2.3.5 Connectionless Vs. Connection-Oriented Servers* 13
 - 2.3.6 Stateless Vs. Stateful Servers* 14
 - 2.3.7 A Stateful File Server Example* 14

2.3.8	<i>Statelessness Is A Protocol Issue</i>	16
2.3.9	<i>Servers As Clients</i>	17
2.4	<i>Summary</i>	18

Chapter 3 Concurrent Processing In Client-Server Software **21**

3.1	<i>Introduction</i>	21
3.2	<i>Concurrency In Networks</i>	21
3.3	<i>Concurrency In Servers</i>	23
3.4	<i>Terminology And Concepts</i>	24
3.4.1	<i>The Process Concept</i>	25
3.4.2	<i>Programs vs. Processes</i>	25
3.4.3	<i>Procedure Calls</i>	26
3.5	<i>An Example Of Concurrent Process Creation</i>	26
3.5.1	<i>A Sequential C Example</i>	26
3.5.2	<i>A Concurrent Version</i>	27
3.5.3	<i>Timeslicing</i>	29
3.5.4	<i>Making Processes Diverge</i>	30
3.6	<i>Executing New Code</i>	31
3.7	<i>Context-Switching And Protocol Software Design</i>	32
3.8	<i>Concurrency And Asynchronous I/O</i>	32
3.9	<i>Summary</i>	33

Chapter 4 Program Interface To Protocols **35**

4.1	<i>Introduction</i>	35
4.2	<i>Loosely Specified Protocol Software Interface</i>	35
4.2.1	<i>Advantages And Disadvantages</i>	36
4.3	<i>Interface Functionality</i>	36
4.4	<i>Conceptual Interface Specification</i>	37
4.5	<i>System Calls</i>	37
4.6	<i>Two Basic Approaches To Network Communication</i>	38
4.7	<i>The Basic I/O Functions Available In UNIX</i>	39
4.8	<i>Using UNIX I/O With TCP/IP</i>	40
4.9	<i>Summary</i>	40

Chapter 5 The Socket Interface **43**

5.1	<i>Introduction</i>	43
5.2	<i>Berkeley Sockets</i>	43
5.3	<i>Specifying A Protocol Interface</i>	44

5.4	<i>The Socket Abstraction</i>	45
5.4.1	<i>Socket Descriptors And File Descriptors</i>	45
5.4.2	<i>System Data Structures For Sockets</i>	46
5.4.3	<i>Using Sockets</i>	47
5.5	<i>Specifying An Endpoint Address</i>	47
5.6	<i>A Generic Address Structure</i>	48
5.7	<i>Major System Calls Used With Sockets</i>	49
5.7.1	<i>The Socket Call</i>	49
5.7.2	<i>The Connect Call</i>	50
5.7.3	<i>The Write Call</i>	50
5.7.4	<i>The Read Call</i>	50
5.7.5	<i>The Close Call</i>	50
5.7.6	<i>The Bind Call</i>	51
5.7.7	<i>The Listen Call</i>	51
5.7.8	<i>The Accept Call</i>	51
5.7.9	<i>Summary Of Socket Calls Used With TCP</i>	51
5.8	<i>Utility Routines For Integer Conversion</i>	52
5.9	<i>Using Socket Calls In A Program</i>	53
5.10	<i>Symbolic Constants For Socket Call Parameters</i>	54
5.11	<i>Summary</i>	54

Chapter 6 Algorithms And Issues In Client Software Design

57

6.1	<i>Introduction</i>	57
6.2	<i>Learning Algorithms Instead Of Details</i>	57
6.3	<i>Client Architecture</i>	58
6.4	<i>Identifying The Location Of A Server</i>	58
6.5	<i>Parsing An Address Argument</i>	60
6.6	<i>Looking Up A Domain Name</i>	61
6.7	<i>Looking Up A Well-Known Port By Name</i>	62
6.8	<i>Port Numbers And Network Byte Order</i>	62
6.9	<i>Looking Up A Protocol By Name</i>	63
6.10	<i>The TCP Client Algorithm</i>	63
6.11	<i>Allocating A Socket</i>	64
6.12	<i>Choosing A Local Protocol Port Number</i>	65
6.13	<i>A Fundamental Problem In Choosing A Local IP Address</i>	65
6.14	<i>Connecting A TCP Socket To A Server</i>	66
6.15	<i>Communicating With The Server Using TCP</i>	66
6.16	<i>Reading A Response From A TCP Connection</i>	67
6.17	<i>Closing A TCP Connection</i>	68
6.17.1	<i>The Need For Partial Close</i>	68
6.17.2	<i>A Partial Close Operation</i>	68
6.18	<i>Programming A UDP Client</i>	69

6.19	<i>Connected And Unconnected UDP Sockets</i>	69
6.20	<i>Using Connect With UDP</i>	70
6.21	<i>Communicating With A Server Using UDP</i>	70
6.22	<i>Closing A Socket That Uses UDP</i>	70
6.23	<i>Partial Close For UDP</i>	71
6.24	<i>A Warning About UDP Unreliability</i>	71
6.25	<i>Summary</i>	71

Chapter 7 Example Client Software

75

7.1	<i>Introduction</i>	75
7.2	<i>The Importance Of Small Examples</i>	75
7.3	<i>Hiding Details</i>	76
7.4	<i>An Example Procedure Library For Client Programs</i>	76
7.5	<i>Implementation Of ConnectTCP</i>	77
7.6	<i>Implementation Of ConnectUDP</i>	78
7.7	<i>A Procedure That Forms Connections</i>	79
7.8	<i>Using The Example Library</i>	81
7.9	<i>The DAYTIME Service</i>	82
7.10	<i>Implementation Of A TCP Client For DAYTIME</i>	82
7.11	<i>Reading From A TCP Connection</i>	84
7.12	<i>The TIME Service</i>	84
7.13	<i>Accessing The TIME Service</i>	85
7.14	<i>Accurate Times And Network Delays</i>	85
7.15	<i>A UDP Client For The TIME Service</i>	86
7.16	<i>The ECHO Service</i>	88
7.17	<i>A TCP Client For The ECHO Service</i>	88
7.18	<i>A UDP Client For The ECHO Service</i>	90
7.19	<i>Summary</i>	92

Chapter 8 Algorithms And Issues In Server Software Design

95

8.1	<i>Introduction</i>	95
8.2	<i>The Conceptual Server Algorithm</i>	95
8.3	<i>Concurrent Vs. Iterative Servers</i>	96
8.4	<i>Connection-Oriented Vs. Connectionless Access</i>	96
8.5	<i>Connection-Oriented Servers</i>	97
8.6	<i>Connectionless Servers</i>	97
8.7	<i>Failure, Reliability, And Statelessness</i>	98
8.8	<i>Optimizing Stateless Servers</i>	99
8.9	<i>Four Basic Types Of Servers</i>	101
8.10	<i>Request Processing Time</i>	102

8.11	<i>Iterative Server Algorithms</i>	102
8.12	<i>An Iterative, Connection-Oriented Server Algorithm</i>	103
8.13	<i>Binding To A Well-Known Address Using INADDR_ANY</i>	103
8.14	<i>Placing The Socket In Passive Mode</i>	104
8.15	<i>Accepting Connections And Using Them</i>	104
8.16	<i>An Iterative, Connectionless Server Algorithm</i>	104
8.17	<i>Forming A Reply Address In A Connectionless Server</i>	105
8.18	<i>Concurrent Server Algorithms</i>	106
8.19	<i>Master And Slave Processes</i>	106
8.20	<i>A Concurrent, Connectionless Server Algorithm</i>	107
8.21	<i>A Concurrent, Connection-Oriented Server Algorithm</i>	107
8.22	<i>Using Separate Programs As Slaves</i>	108
8.23	<i>Apparent Concurrency Using A Single Process</i>	109
8.24	<i>When To Use Each Server Type</i>	110
8.25	<i>A Summary of Server Types</i>	111
8.26	<i>The Important Problem Of Server Deadlock</i>	112
8.27	<i>Alternative Implementations</i>	112
8.28	<i>Summary</i>	113

Chapter 9 Iterative, Connectionless Servers (UDP)

115

9.1	<i>Introduction</i>	115
9.2	<i>Creating A Passive Socket</i>	115
9.3	<i>Process Structure</i>	119
9.4	<i>An Example TIME Server</i>	119
9.5	<i>Summary</i>	121

Chapter 10 Iterative, Connection-Oriented Servers (TCP)

123

10.1	<i>Introduction</i>	123
10.2	<i>Allocating A Passive TCP Socket</i>	123
10.3	<i>A Server For The DAYTIME Service</i>	124
10.4	<i>Process Structure</i>	124
10.5	<i>An Example DAYTIME Server</i>	125
10.6	<i>Closing Connections</i>	128
10.7	<i>Connection Termination And Server Vulnerability</i>	128
10.8	<i>Summary</i>	129

Chapter 11	Concurrent, Connection-Oriented Servers (TCP)	131
11.1	<i>Introduction</i>	131
11.2	<i>Concurrent ECHO</i>	131
11.3	<i>Iterative Vs. Concurrent Implementations</i>	132
11.4	<i>Process Structure</i>	132
11.5	<i>An Example Concurrent ECHO Server</i>	133
11.6	<i>Cleaning Up Errant Processes</i>	137
11.7	<i>Summary</i>	138
Chapter 12	Single-Process, Concurrent Servers (TCP)	139
12.1	<i>Introduction</i>	139
12.2	<i>Data-driven Processing In A Server</i>	139
12.3	<i>Data-Driven Processing With A Single Process</i>	140
12.4	<i>Process Structure Of A Single-Process Server</i>	141
12.5	<i>An Example Single-Process ECHO Server</i>	142
12.6	<i>Summary</i>	144
Chapter 13	Multiprotocol Servers (TCP, UDP)	147
13.1	<i>Introduction</i>	147
13.2	<i>The Motivation For Reducing The Number Of Servers</i>	147
13.3	<i>Multiprotocol Server Design</i>	148
13.4	<i>Process Structure</i>	148
13.5	<i>An Example Multiprotocol DAYTIME Server</i>	149
13.6	<i>The Concept Of Shared Code</i>	153
13.7	<i>Concurrent Multiprotocol Servers</i>	153
13.8	<i>Summary</i>	153
Chapter 14	Multiservice Servers (TCP, UDP)	155
14.1	<i>Introduction</i>	155
14.2	<i>Consolidating Servers</i>	155
14.3	<i>A Connectionless, Multiservice Server Design</i>	156
14.4	<i>A Connection-Oriented, Multiservice Server Design</i>	157
14.5	<i>A Concurrent, Connection-Oriented, Multiservice Server</i>	158
14.6	<i>A Single-Process, Multiservice Server Implementation</i>	158
14.7	<i>Invoking Separate Programs From A Multiservice Server</i>	159
14.8	<i>Multiservice, Multiprotocol Designs</i>	160

14.9	<i>An Example Multiservice Server</i>	161
14.10	<i>Static and Dynamic Server Configuration</i>	168
14.11	<i>The UNIX Super Server, Inetd</i>	169
14.12	<i>An Example Inetd Server</i>	171
14.13	<i>Summary</i>	173

Chapter 15 Uniform, Efficient Management Of Server Concurrency 175

15.1	<i>Introduction</i>	175
15.2	<i>Choosing Between An Iterative And A Concurrent Design</i>	175
15.3	<i>Level Of Concurrency</i>	176
15.4	<i>Demand-Driven Concurrency</i>	177
15.5	<i>The Cost Of Concurrency</i>	177
15.6	<i>Overhead And Delay</i>	177
15.7	<i>Small Delays Can Matter</i>	178
15.8	<i>Process Preallocation</i>	179
15.8.1	<i>Preallocation In UNIX</i>	180
15.8.2	<i>Preallocation In A Connection-Oriented Server</i>	180
15.8.3	<i>Preallocation In A Connectionless Server</i>	181
15.8.4	<i>Preallocation, Bursty Traffic, And NFS</i>	182
15.8.5	<i>Process Preallocation On A Multiprocessor</i>	183
15.9	<i>Delayed Process Allocation</i>	183
15.10	<i>The Uniform Basis For Both Techniques</i>	184
15.11	<i>Combining Techniques</i>	185
15.12	<i>Summary</i>	185

Chapter 16 Concurrency In Clients 187

16.1	<i>Introduction</i>	187
16.2	<i>The Advantages Of Concurrency</i>	187
16.3	<i>The Motivation For Exercising Control</i>	188
16.4	<i>Concurrent Contact With Multiple Servers</i>	189
16.5	<i>Implementing Concurrent Clients</i>	189
16.6	<i>Single-Process Implementations</i>	191
16.7	<i>An Example Concurrent Client That Uses ECHO</i>	192
16.8	<i>Execution Of The Concurrent Client</i>	196
16.9	<i>Concurrency In The Example Code</i>	197
16.10	<i>Summary</i>	198

Chapter 17 Tunneling At The Transport And Application Levels 199

- 17.1 Introduction 199*
- 17.2 Multiprotocol Environments 199*
- 17.3 Mixing Network Technologies 201*
- 17.4 Dynamic Circuit Allocation 202*
- 17.5 Encapsulation And Tunneling 203*
- 17.6 Tunneling Through An IP Internet 203*
- 17.7 Application-Level Tunneling Between Clients And Servers 204*
- 17.8 Tunneling, Encapsulation, And Dialup Phone Lines 205*
- 17.9 Summary 206*

Chapter 18 Application Level Gateways 209

- 18.1 Introduction 209*
- 18.2 Clients And Servers In Constrained Environments 209*
 - 18.2.1 The Reality Of Multiple Technologies 209*
 - 18.2.2 Computers With Limited Functionality 210*
 - 18.2.3 Connectivity Constraints That Arise From Security 210*
- 18.3 Using Application Gateways 211*
- 18.4 Interoperability Through A Mail Gateway 212*
- 18.5 Implementation Of A Mail Gateway 213*
- 18.6 A Comparison Of Application Gateways And Tunneling 213*
- 18.7 Application Gateways And Limited Functionality Systems 215*
- 18.8 Application Gateways Used For Security 216*
- 18.9 Application Gateways And The Extra Hop Problem 217*
- 18.10 An Example Application Gateway 219*
- 18.11 Implementation Of An Application Gateway 220*
- 18.12 Code For The Application Gateway 221*
- 18.13 An Example Gateway Exchange 223*
- 18.14 Using Rfcd With UNIX's .forward 223*
- 18.15 A General-Purpose Application Gateway 224*
- 18.16 Operation Of SLIRP 224*
- 18.17 How SLIRP Handles Connections 225*
- 18.18 IP Addressing And SLIRP 225*
- 18.19 Summary 226*

Chapter 19 External Data Representation (XDR) 229

- 19.1 Introduction 229*
- 19.2 Representations For Data In Computers 229*

19.3	<i>The N-Squared Conversion Problem</i>	230
19.4	<i>Network Standard Byte Order</i>	231
19.5	<i>A De Facto Standard External Data Representation</i>	232
19.6	<i>XDR Data Types</i>	233
19.7	<i>Implicit Types</i>	234
19.8	<i>Software Support For Using XDR</i>	234
19.9	<i>XDR Library Routines</i>	234
19.10	<i>Building A Message One Piece At A Time</i>	234
19.11	<i>Conversion Routines In The XDR Library</i>	236
19.12	<i>XDR Streams, I/O, and TCP</i>	238
19.13	<i>Records, Record Boundaries, And Datagram I/O</i>	239
19.14	<i>Summary</i>	239

Chapter 20 Remote Procedure Call Concept (RPC)

241

20.1	<i>Introduction</i>	241
20.2	<i>Remote Procedure Call Model</i>	241
20.3	<i>Two Paradigms For Building Distributed Programs</i>	242
20.4	<i>A Conceptual Model For Conventional Procedure Calls</i>	243
20.5	<i>An Extension Of the Procedural Model</i>	243
20.6	<i>Execution Of Conventional Procedure Call And Return</i>	244
20.7	<i>The Procedural Model In Distributed Systems</i>	245
20.8	<i>Analogy Between Client-Server And RPC</i>	246
20.9	<i>Distributed Computation As A Program</i>	247
20.10	<i>Sun Microsystems' Remote Procedure Call Definition</i>	248
20.11	<i>Remote Programs And Procedures</i>	248
20.12	<i>Reducing The Number Of Arguments</i>	249
20.13	<i>Identifying Remote Programs And Procedures</i>	249
20.14	<i>Accommodating Multiple Versions Of A Remote Program</i>	250
20.15	<i>Mutual Exclusion For Procedures In A Remote Program</i>	251
20.16	<i>Communication Semantics</i>	252
20.17	<i>At Least Once Semantics</i>	252
20.18	<i>RPC Retransmission</i>	253
20.19	<i>Mapping A Remote Program To A Protocol Port</i>	253
20.20	<i>Dynamic Port Mapping</i>	254
20.21	<i>RPC Port Mapper Algorithm</i>	255
20.22	<i>ONC RPC Message Format</i>	257
20.23	<i>Marshaling Arguments For A Remote Procedure</i>	258
20.24	<i>Authentication</i>	258
20.25	<i>An Example Of RPC Message Representation</i>	259
20.26	<i>An Example Of The UNIX Authentication Field</i>	260
20.27	<i>Summary</i>	261

Chapter 21 Distributed Program Generation (Rpcgen Concept) 265

- 21.1 Introduction 265
- 21.2 Using Remote Procedure Calls 266
- 21.3 Programming Mechanisms To Support RPC 267
- 21.4 Dividing A Program Into Local And Remote Procedures 268
- 21.5 Adding Code For RPC 269
- 21.6 Stub Procedures 269
- 21.7 Multiple Remote Procedures And Dispatching 270
- 21.8 Name Of The Client-Side Stub Procedure 271
- 21.9 Using Rpcgen To Generate Distributed Programs 272
- 21.10 Rpcgen Output And Interface Procedures 272
- 21.11 Rpcgen Input And Output 273
- 21.12 Using Rpcgen To Build A Client And Server 274
- 21.13 Summary 274

Chapter 22 Distributed Program Generation (Rpcgen Example) 277

- 22.1 Introduction 277
- 22.2 An Example To Illustrate Rpcgen 278
- 22.3 Dictionary Look Up 278
- 22.4 Eight Steps To A Distributed Application 279
- 22.5 Step 1: Build A Conventional Application Program 280
- 22.6 Step 2: Divide The Program Into Two Parts 284
- 22.7 Step 3: Create An Rpcgen Specification 290
- 22.8 Step 4: Run Rpcgen 292
- 22.9 The .h File Produced By Rpcgen 292
- 22.10 The XDR Conversion File Produced By Rpcgen 293
- 22.11 The Client Code Produced By Rpcgen 294
- 22.12 The Server Code Produced By Rpcgen 296
- 22.13 Step 5: Write Stub Interface Procedures 299
 - 22.13.1 Client-Side Interface Routines 299
 - 22.13.2 Server-Side Interface Routines 301
- 22.14 Step 6: Compile And Link The Client Program 303
- 22.15 Step 7: Compile And Link The Server Program 307
- 22.16 Step 8: Start The Server And Execute The Client 309
- 22.17 Using The UNIX Make Utility 309
- 22.18 Summary 311