TCP/IP Sockets in Java
Second Edition
The Morgan Kaufmann Practical Guides Series
Series Editor: Michael J. Donahoo

Kenneth L. Calvert and Michael J. Donahoo

SQL: Practical Guide for Developers
Michael J. Donahoo and Gregory Speegle

C# 2.0: Practical Guide for Programmers
Michel de Champlain and Brian Patrick

Multi-Tier Application Programming with PHP: Practical Guide for Architects and Programmers
David Wall

TCP/IP Sockets in C#: Practical Guide for Programmers
David Makofske, Michael J. Donahoo, and Kenneth L. Calvert

Java Cryptography Extensions: Practical Guide for Programmers
Jason Weiss

JSP: Practical Guide for Programmers
Robert Brunner

JSTL: Practical Guide for JSP Programmers
Sue Spielman

Java: Practical Guide for Programmers
Michael Sikora

Multicast Sockets: Practical Guide for Programmers
David Makofske and Kevin Almeroth

The Struts Framework: Practical Guide for Java Programmers
Sue Spielman

TCP/IP Sockets in C: Practical Guide for Programmers
Kenneth L. Calvert and Michael J. Donahoo

JDBC: Practical Guide for Java Programmers
Gregory Speegle

For further information on these books and for a list of forthcoming titles, please visit our Web site at http://www.mkp.com.
To Tricia and Lisa
## Contents

Preface  xi

1 **Introduction**  1  
   1.1 Networks, Packets, and Protocols  1  
   1.2 About Addresses  4  
   1.3 About Names  6  
   1.4 Clients and Servers  6  
   1.5 What Is a Socket?  7  
   1.6 Exercises  8  

2 **Basic Sockets**  9  
   2.1 Socket Addresses  9  
   2.2 TCP Sockets  15  
      2.2.1 TCP Client  16  
      2.2.2 TCP Server  21  
      2.2.3 Input and Output Streams  25  
   2.3 UDP Sockets  26  
      2.3.1 DatagramPacket  27  
      2.3.2 UDP Client  29  
      2.3.3 UDP Server  34  
      2.3.4 Sending and Receiving with UDP Sockets  36  
   2.4 Exercises  38  

3 **Sending and Receiving Data**  39  
   3.1 Encoding Information  40  
      3.1.1 Primitive Integers  40  

vii
Contents

3.1.2 Strings and Text 45
3.1.3 Bit-Diddling: Encoding Booleans 47
3.2 Composing I/O Streams 48
3.3 Framing and Parsing 49
3.4 Java-Specific Encodings 55
3.5 Constructing and Parsing Protocol Messages 55
  3.5.1 Text-Based Representation 58
  3.5.2 Binary Representation 61
  3.5.3 Sending and Receiving 63
3.6 Wrapping Up 71
3.7 Exercises 71

4 Beyond the Basics 73
4.1 Multitasking 73
  4.1.1 Java Threads 74
  4.1.2 Server Protocol 76
  4.1.3 Thread-per-Client 80
  4.1.4 Thread Pool 82
  4.1.5 System-Managed Dispatching: The Executor Interface 84
4.2 Blocking and Timeouts 86
  4.2.1 accept(), read(), and receive() 87
  4.2.2 Connecting and Writing 87
  4.2.3 Limiting Per-Client Time 87
4.3 Multiple Recipients 89
  4.3.1 Broadcast 90
  4.3.2 Multicast 90
4.4 Controlling Default Behaviors 95
  4.4.1 Keep-Alive 96
  4.4.2 Send and Receive Buffer Size 96
  4.4.3 Timeout 97
  4.4.4 Address Reuse 97
  4.4.5 Eliminating Buffering Delay 98
  4.4.6 Urgent Data 98
  4.4.7 Lingering after Close 99
  4.4.8 Broadcast Permission 99
  4.4.9 Traffic Class 100
  4.4.10 Performance-Based Protocol Selection 100
4.5 Closing Connections 101
4.6 Applets 107
4.7 Wrapping Up 107
4.8 Exercises 108

5 NIO 109
5.1 Why Do We Need This? 109
5.2 Using Channels with Buffers 112
Contents

5.3 Selectors 115
5.4 Buffers in Detail 121
  5.4.1 Buffer Indices 121
  5.4.2 Buffer Creation 122
  5.4.3 Storing and Retrieving Data 124
  5.4.4 Preparing Buffers: clear(), flip(), and rewind() 126
  5.4.5 Compacting Data in a Buffer 128
  5.4.6 Buffer Perspectives: duplicate(), slice(), etc. 129
  5.4.7 Character Coding 131
5.5 Stream (TCP) Channels in Detail 132
5.6 Selectors in Detail 135
  5.6.1 Registering Interest in Channels 135
  5.6.2 Selecting and Identifying Ready Channels 138
  5.6.3 Channel Attachments 140
  5.6.4 Selectors in a Nutshell 140
5.7 Datagram (UDP) Channels 141
5.8 Exercises 145

6 Under the Hood 147
  6.1 Buffering and TCP 150
  6.2 Deadlock Danger 152
  6.3 Performance Implications 155
  6.4 TCP Socket Life Cycle 155
    6.4.1 Connecting 156
    6.4.2 Closing a TCP Connection 160
  6.5 Demultiplexing Demystified 163
  6.6 Exercises 165

Bibliography 167

Index 169
Preface

For years, college courses in computer networking were taught with little or no hands-on experience. For various reasons, including some good ones, instructors approached the principles of computer networking primarily through equations, analyses, and abstract descriptions of protocol stacks. Textbooks might have included code, but it would have been unconnected to anything students could get their hands on. We believe, however, that students learn better when they can see (and then build) concrete examples of the principles at work. And, fortunately, things have changed. The Internet has become a part of everyday life, and access to its services is readily available to most students (and their programs). Moreover, copious examples—good and bad—of nontrivial software are freely available.

We wrote this book for the same reason we wrote *TCP/IP Sockets in C*: We needed a resource to support learning networking through programming exercises in our courses. Our goal is to provide a sufficient introduction so that students can get their hands on real network services without too much hand-holding. After grasping the basics, students can then move on to more advanced assignments, which support learning about routing algorithms, multimedia protocols, medium access control, and so on. We have tried to make this book equivalent to our earlier book to enable instructors to allow students to choose the language they use and still ensure that all students will come away with the same skills and understanding. Of course, it is not clear that this goal is achievable, but in any case the scope, price, and presentation level of the book are intended to be similar.

Intended Audience

This book is intended for two audiences. The first, which motivated us to write it in the first place, consists of students in undergraduate or graduate courses in computer networks. The second consists of practitioners who know something about Java and want to learn about
Preface

writing Java applications that use the Internet. We have tried to keep the book concise and focused, so it can be used by students as a supplementary text and by practitioners as a low-cost introduction to the subject. As a result, you should not expect to be an expert after reading this book! The goal is to take users far enough that they can start experimenting and learning on their own.

Readers are assumed to have access to a computer equipped with Java. This book is based on Version 1.6 of Java and the Java Virtual Machine (JVM); however, the code should work with earlier versions of Java, with the exception of a few new Java methods. Java is about portability, so the particular hardware and operating system (OS) on which you run should not matter.

Approach

Chapter 1 provides a general overview of networking concepts. It is not, by any means, a complete introduction, but rather is intended to allow readers to synchronize with the concepts and terminology used throughout the book. Chapter 2 introduces the mechanics of simple clients and servers; the code in this chapter can serve as a starting point for a variety of exercises. Chapter 3 covers the basics of message construction and parsing. The reader who digests the first three chapters should in principle be able to implement a client and server for a given (simple) application protocol. Chapters 4 and 5 then deal with increasingly sophisticated techniques for building scalable and robust clients and servers, with Chapter 5 focusing on the facilities introduced by the “New I/O” packages. Finally, in keeping with our goal of illustrating principles through programming, Chapter 6 discusses the relationship between the programming constructs and the underlying protocol implementations in somewhat more detail.

Our general approach introduces programming concepts through simple program examples accompanied by line-by-line commentary that describes the purpose of every part of the program. This lets you see the important objects and methods as they are used in context. As you look at the code, you should be able to understand the purpose of each and every line.

Our examples do not take advantage of all library facilities in Java. Some of these facilities, in particular serialization, effectively require that all communicating peers be implemented in Java. Also, to introduce examples as soon as possible, we wanted to avoid bringing in a thicket of methods and classes that have to be sorted out later. We have tried to keep it simple, especially in the early chapters.

What This Book Is Not

To keep the price of this book within a reasonable range for a supplementary text, we have had to limit its scope and maintain a tight focus on the goals outlined above. We omitted
many topics and directions, so it is probably worth mentioning some of the things this book is not:

- It is not an introduction to the Java language. We focus specifically on TCP/IP socket programming. We expect that the reader is already acquainted with the language features and basic Java libraries—including those (like generics) introduced in later releases—and knows how to develop programs in Java.

- It is not a book on protocols. Reading this book will not make you an expert on IP, TCP, FTP, HTTP, or any other existing protocol (except maybe the echo protocol). Our focus is on the interface to the TCP/IP services provided by the socket abstraction. It will help if you start with some idea about the general workings of TCP and IP, but Chapter 1 may be an adequate substitute.

- It is not a guide to all of Java's rich collection of libraries that are designed to hide communication details (e.g., `HTTPConnection`) and make the programmer's life easier. Since we are teaching the fundamentals of how to do, not how to avoid doing, protocol development, we do not cover these parts of the API. We want readers to understand protocols in terms of what goes on the wire, so we mostly use simple byte streams and deal with character encodings explicitly. As a consequence, this text does not deal with `URL`, `URLConnection`, and so on. We believe that once you understand the principles, using these convenience classes will be straightforward.

- It is not a book on object-oriented design. Our focus is on the important principles of TCP/IP socket programming, and our examples are intended to illustrate them concisely. As far as possible, we try to adhere to object-oriented design principles; however, when doing so adds complexity that obfuscates the socket principles or bloats the code, we sacrifice design for clarity. This text does not cover design patterns for networking. (Though we would like to think that it provides some of the background necessary for understanding such patterns!)

- It is not a book on writing production-quality code. Again, although we strive for a minimum level of robustness, the primary goal of our code examples is education. In order to avoid obscuring the principles with large amounts of error-handling code, we have sacrificed some robustness for brevity and clarity.

- It is not a book on doing your own native sockets implementation in Java. We focus exclusively on TCP/IP sockets as provided by the standard Java distribution and do not cover the various socket implementation wrapper classes (e.g., `SocketImpl`).

- To avoid cluttering the examples with extraneous (nonsocket-related programming) code, we have made them command-line based. While the book's Web site, books.elsevier.com/companions/9780123742551 contains a few examples of GUI-enhanced network applications, we do not include or explain them in this text.

- It is not a book on Java applets. Applets use the same Java networking API so the communication code should be very similar; however, there are severe security restrictions on
the kinds of communication an applet can perform. We provide a very limited discussion of these restrictions and a single applet/application example on the Web site; however, a complete description of applet networking is beyond the scope of this text.

Acknowledgments

We would like to thank all the people who helped make this book a reality. Despite the book’s brevity, many hours went into reviewing the original proposal and the draft, and the reviewers' input significantly shaped the final result.

Thanks to: Michel Barbeau, Chris Edmondson-Yurkanan, Ted Herman, Dave Hollinger, Jim Leone, Dan Schmidt, Erick Wagner, EDS; CSI4321 classes at Baylor University, and CS 471 classes at the University of Kentucky. Any errors that remain are, of course, our responsibility.

This book will not make you an expert—that takes years of experience. However, we hope it will be useful as a resource, even to those who already know quite a bit about using sockets in Java. Both of us enjoyed writing it and learned quite a bit along the way.

Feedback

We invite your suggestions for the improvement of any aspect of this book. If you find an error, please let us know. We will maintain an errata list at the book’s Web site. You can send feedback via the book’s Web page, books.elsevier.com/companions/9780123742551, or you can email us at the addresses below:

Kenneth L. Calvert—calvert@uky.edu
Michael J. Donahoo—Jeff_Donahoo@baylor.edu