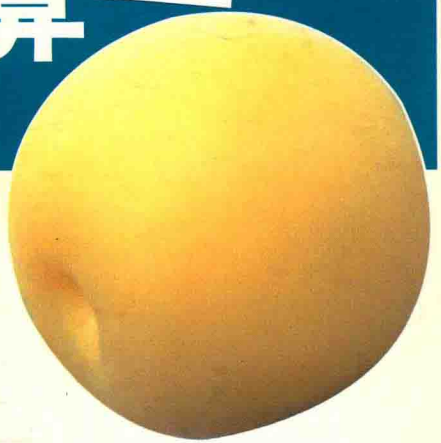
大学计算机教育国外著名教材系列（影印版）

PEARSON Addison Wesley

JAVA NETWORK PROGRAMMING AND DISTRIBUTED COMPUTING

Java网络程序设计与分布式计算



David Reilly
Michael Reilly 著

清华大学出版社

大学计算机教育国外著名教材系列(影印版)

Java Network Programming and Distributed Computing

Java 网络程序设计与分布式计算

David Reilly
Michael Reilly 著

清华大学出版社
北京

图书在版编目(CIP)数据

Java 网络程序设计与分布式计算 = Java Network Programming and Distributed Computing/(美)赖利(Reilly, D.),(美)赖利(Reilly, M.)著. —北京:清华大学出版社,2004.10
(大学计算机教育国外著名教材系列)
ISBN 7-302-09767-4

Ⅰ.J... Ⅱ.①赖...②赖... Ⅲ.JAVA 语言—网络程序设计—高等学校—教材—英文 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2004)第 107001 号

出 版 者: 清华大学出版社
http://www.tup.com.cn
社总机: 010-62770175
地 址: 北京清华大学学研大厦
邮 编: 100084
客户服务: 010-62776969
责任编辑: 龙启铭
印 刷 者: 清华大学印刷厂
装 订 者: 三河市金元装订厂
发 行 者: 新华书店总店北京发行所
开 本: 185×230 印张: 28.75
版 次: 2004 年 10 月第 1 版 2004 年 10 月第 1 次印刷
书 号: ISBN 7-302-09767-4/TP·6744
印 数: 1~4000
定 价: 44.80 元

出 版 说 明

进入21世纪，世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才，谁就能在竞争中取得优势。高等教育，作为培养高素质人才的事业，必然受到高度重视。目前我国高等教育的教材更新较慢，为了加快教材的更新频率，教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从1996年开始，与国外著名出版公司合作，影印出版了“大学计算机教育丛书(影印版)”等一系列引进图书，受到国内读者的欢迎和支持。跨入21世纪，我们本着为我国高等教育教材建设服务的初衷，在已有的基础上，进一步扩大选题内容，改变图书开本尺寸，一如既往地请有关专家挑选适用于我国高等本科及研究生计算机教育的国外经典教材或著名教材，组成本套“大学计算机教育国外著名教材系列(影印版)”，以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材，以利我们把“大学计算机教育国外著名教材系列(影印版)”做得更好，更适合高校师生的需要。

清华大学出版社

PREFACE

Welcome to *Java Network Programming and Distributed Computing*. The goal of this book is to introduce and explain the basic concepts of networking and discuss the practical aspects of Java network programming.

This book will help readers get up to speed with network programming and employ the techniques learned in software development. If you've had some networking experience in another language and want to apply your existing skills to Java, you'll find the book to be an accelerated guide and a comprehensive reference to the networking API. This book does not require you to be a networking guru, however, as Chapters 1–4 provide a gentle introduction to networking theory, Java, and the most basic elements of the Java networking API. In later chapters, the Java API is covered in greater detail, with a discussion supplementing the documentation that Sun Microsystems provides as a reference.

What You'll Learn

In this book, readers will learn how to write applications in Java that make use of network programming. The Java API provides many ways to communicate over the Internet, from sending packets and streams of data to employing higher-level application protocols such as HTTP and distributed computing mechanisms.

Along the way, you'll read about:

- How the Internet works, its architecture and the TCP/IP protocol stack
- The Java programming language, including a refresher course on topics such as exception handling
- Java's input/output system and how it works
- How to write clients and servers using the User Datagram Protocol (UDP) and the Transport Control Protocol (TCP)

- The advantages of multi-threaded applications, which allow network applications to perform multiple tasks concurrently
- How to implement network protocols, including examples of client/server implementations
- The HyperText Transfer Protocol (HTTP) and how to access the World Wide Web using Java
- How to write server-side Java applications for the WWW
- Distributed computing technologies including remote method invocation (RMI) and CORBA
- How to access e-mail using the extensive JavaMail API

What You'll Need

A reasonable familiarity with Java programming is required to get the most out of this book. You'll need to be able to compile and run Java applications and to understand basic concepts such as classes, objects, and the Java API. However, you don't need to be an expert with respect to the more advanced topics covered herein, such as I/O streams and multi-threading. All examples use a text interface, so there's no need to have GUI experience.

You'll also need to install the Java SDK, available for free from Sun Microsystems (http://java.sun.com/j2se/). Java programmers will no doubt already have access to the SDK, but readers should be aware that some examples in this text will require JDK 1.1, and the advanced sections on servlets, RMI and CORBA, and JavaMail will require Java 2.

A minimal amount of additional software is required, and most of the tools for Java programming are available for free and downloadable via the WWW. Chapter 2 includes an overview of Java development tools, but readers can also use their existing code editor. Readers will be advised when examples feature additional Sun Microsystems software.

Companion Web Site

As a companion to the material covered in this book, the book's Web site offers the source code in downloadable form (no need to wear out your fingers!), as well as a list of Frequently Asked Questions about Java Networking, links to networking resources, and additional information about the book. The site can be found at

http://www.davidreilly.com/jnpbook/.

Contacting the Authors

We welcome feedback from readers, be it comments on specific chapters or sections or an evaluation of the book as a whole. In particular, reader input about whether topics were clearly conveyed and sufficiently comprehensive would be appreciated. While we'd love to receive only praise, honest opinions are valued (as well as suggestions about coverage of new networking topics).

Feel free to contact us directly. While we can't guarantee an individual reply, we'll do our best to respond to your query. Please send questions and feedback via e-mail to: jnpbook@davidreilly.com.

David Reilly and Michael Reilly
September 2001

ACKNOWLEDGMENTS

This book would not have been possible without the assistance of our peer reviewers, who contributed greatly to improving its quality and allowing us to deliver a guide to Java network programming that is both clear and comprehensive. Our thanks go to Michael Brundage, Elisabeth Freeman, Bob Kitzberge, Lak Ming Lam, Ian Lance Taylor, and John J. Wegis.

We'd like to make special mention of two reviewers who contributed detailed reviews and offered insightful recommendations: Howard Lee Harkness and D. Jay Newman. Most of all, we would like to thank Amy Fong, whose thoroughness and invaluable suggestions, including questions that the inquisitive reader might have about TCP/IP and Java, helped shape the book that you are reading today.

We'd also like to thank our editorial team at Addison-Wesley, including Karen Gettman, whose initial encouragement and persistence convinced us to take on the project, Mary Hart, Marcy Barnes-Henrie, Melissa Dobson, and Emily Frey. Their support throughout the process of writing, editing, and preparing this book for publication is most heartily appreciated.

CONTENTS

CHAPTER 1

Networking Theory

This chapter provides an overview of the basic concepts of networking and discusses essential topics of networking theory. Readers experienced with networking may choose to skip over some of these preliminary sections, although a refresher course on basic networking concepts will be useful, as later chapters presume a knowledge of this theory on the part of the reader. A solid understanding of the relationship between the various protocols that make up the TCP/IP suite is required for network programming.

1.1 What Is a Network?

Put simply, a network is a collection of devices that share a common communication protocol and a common communication medium (such as network cables, dial-up connections, and wireless links). We use the term *devices* in this definition rather than *computers*, even though most people think of a network as being a collection of computers; certainly the basic concept of a network in most peoples' mind is of an assembly of network servers and desktop machines.

However, to say that networks are merely a collection of computers is to limit the range of hardware that can use them. For example, printers may be shared across a network, allowing more than one machine to gain access to their services. Other types of devices can also be connected to a network; these devices can provide access to information, or offer services that may be controlled remotely. Indeed, there is a growing movement toward connecting non-computing devices to networks. While the technology is still evolving, we're moving toward a network-centric as opposed to a computing-centric model. Services and devices can be distributed across a network rather than being bound to individual machines. In the same way, users can move from machine to machine, logging on as if they were sitting at their own familiar terminal.

One fun and popular example from very early on in the history of networking is the soda machine connected to the Internet, allowing people around the world to see how many cans of a certain flavor of drink were available. While a trivial application, it served to demonstrate the power of networking devices. Indeed, as home networks become easier to use and more affordable, we may even see regular household appliances such as telephones, televisions, and home stereo systems connected to local networks or even to the Internet.

Network and software standards such as Sun's Jini already exist to help devices and hardware talk to each other over networks and to allow instant plug-and-play functionality. Devices and services can be added and removed from the network (as, for example, when you unplug your printer and take it to the next room) without the need for complex administration and configuration. It is anticipated that over the course of the next few years, users will become just as comfortable and familiar with network-centric computing as they are with the Internet.

In addition to devices that provide services are devices that keep the network going. Depending on the complexity of a network and its physical architecture, elements forming it may include network cards, routers, hubs, and gateways. These terms are defined below.

- *Network cards* are hardware devices added to a computer to allow it to talk to a network. The most common network card in use today is the Ethernet card. Network cards usually connect to a network cable, which is the link to the network and the medium through which data is transmitted. However, other media exist, such as dial-up connections through a phone line, and wireless links.
- *Routers* are machines that act as switches. These machines direct packets of data to the next "hop" in their journey across a network.
- *Hubs* provide connections that allow multiple computers to access a network (for example, allowing two desktop machines to access a local area network).
- *Gateways* connect one network to another—for example, a local area network to the Internet. While routers and gateways are similar, a router does not have to bridge multiple networks. In some cases, routers are also gateways.

While it is useful to understand such networking terminology as it is widely used in networking texts and protocol specifications, programmers do not generally need to be concerned with the implementation details of a network and its underlying architecture. However, it is important for programmers to be aware of the various elements making up the network.

1.2 How Do Networks Communicate?

Networks consist of connections between computers and devices. These connections are most commonly physical connections, such as wires and cables, through which electricity is sent. However, many other media exist. For example, it is possible to use infrared and radio as a communication medium for transmitting data wirelessly, or fiber-optic cables that use light rather than electricity.

Such connections carry data between one point in the network and another. This data is represented as bits of information (either "on" or "off," a "zero" or a "one"). Whether through a physical medium such as a cable, through the air, or using light, this raw data is passed across various points in the network called nodes; a node could represent a computer, another type of hardware device such as a printer, or a piece of networking equipment that relays this information onward to other nodes in the network or to an entirely different network. Of course, for data to be successfully delivered to individual nodes, these nodes must be clearly identifiable.

1.2.1 Addressing

Each node in a network is typically represented by an address, just as a street name and number, town or city, and zip code identifies individual homes and offices. The manufacturer of the network interface card (NIC) installed in such devices is responsible for ensuring that no two card addresses are alike, and chooses a suitable addressing scheme. Each card will have this address stored permanently, so that it remains fixed—it cannot be manually assigned or modified, although some operating systems will allow these addresses to be faked in the event of an accidental conflict with another card's address.

Because of the wide variety of NICs, many addressing schemes are used. For example, Ethernet network cards are assigned a unique 48-bit number to distinguish one card from another. Usually, a numerical number is assigned to each card, and manufacturers are allocated batches of numbers. This system must be strictly regulated by industry, of course—two cards with the same address would cause headaches for network administrators. The physical address is referred to by many names (some of which are specific to a certain type of card, while others are general terms), including:

- Hardware address
- Ethernet address
- Media Access Control (MAC) address
- NIC address

These addresses are used to send information to the appropriate node. If two nodes shared the same address, they would be competing for the same information and one would inevitably lose out, or both would receive the same data. Often, machines are known by more than one type of address. A network server may have a physical Ethernet address as well as an Internet Protocol (IP) address that distinguishes it from other hosts on the Internet, or it may have more than one network card.

Within a local area network, machines can use physical addresses to communicate. However, since there are many types of these addresses, they are not appropriate for internetwork communication. As discussed later in this chapter, the IP address is used for this purpose.

1.2.2 Data Transmission Using Packets

Sending individual bits of data from node to node is not very cost effective, as a fair bit of overhead is involved in relaying the necessary address information every time a byte of data is transmitted. Most networks, instead, group data into packets. Packets consist of a header and data segment, as shown in Figure 1-1. The header contains addressing information (such as the sender and the recipient), checksums to ensure that a packet has not been corrupted, as well as other useful information that is needed for transmission across the network. The data segment contains sequences of bytes, comprising the actual data being sent from one node to another. Since the header information is needed only for transmission, applications are interested only in the data segment. Ideally, as much data as possible would be combined into a packet, in order to minimize the overhead of the headers. However, if information needs to be sent quickly, packets may be dispatched when nearly empty. Depending on the type of packet and protocol being used, packets may also be padded out to fit a fixed length of bytes.

When a node on the network is ready to transmit a packet, a direct connection to the destination node is usually not available. Instead, intermediary nodes carry packets from one location to another, and this process is repeated indefinitely until the packet reaches its destination. Due to network conditions (such as congestion or network failures), packets may take arbitrary routes, and sometimes they may be lost in transit or arrive out of sequence. This may seem like a chaotic way of communicating, but as will be seen in later chapters, there are ways to guarantee delivery and sequencing. Indeed, the properties of guaranteed delivery and sequential order are often irrelevant to certain types of applications (such as streaming video and audio, where it is more important to present current video frames and audio segments than to retransmit lost ones). When these properties are necessary, networking software can keep track of lost packets and out-of-sequence data for applications.