21世纪大学计算机专业教材

网络编程与开发技术

殷肖川 刘志宏 姬伟锋 万映辉 编著
李增智 唐亚哲 主审

西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

21世纪大学计算机专业教材

网络编程与开发技术

殷肖川 刘志宏 姬伟锋 万映辉 编著
李增智 唐亚哲 主审

西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS
·西安·

内 容 提 要

本书系统介绍了网络通信软件设计的原理和方法，详细讨论了在 Windows 环境下的各种网络编程接口和网络通信程序设计技术，深入分析了各种设计方法的原理以及异常处理方法，主要内容包括：基于 Net BIOS 的网络编程，基于 TCP/IP 协议的网络编程，进程通信与分布计算，多线程结构的网络编程技术，直接网络编程技术，网络数据包捕获与分析等，附录部分给出了常用网络 API 函数和错误代码。

本书遵循理论与实践相结合的原则，在系统介绍理论的前题下，结合作者实际工作经验，深入讨论了在工程项目中可能遇到的问题和解决问题的方法，并给出了适量的编程实例，以飨读者。各章附有适量习题，便于学生课后练习。本教材可作为高校计算机专业及相关专业研究生、本科生学习网络通信软件设计等相关课程的教科书，也可作为从事计算机网络和数据通信工作的工程技术人员的参考书。

图书在版编目(CIP)数据

网络编程与开发技术/殷肖川，刘志宏，姬伟锋，万映辉编著
—西安：西安交通大学出版社，2003.9
21 世纪大学计算机专业教材
ISBN 7—5605—1750—1

Ⅰ.网… Ⅱ.①殷… ②刘… ③姬… ④万… Ⅲ.计算机网络—程序设计—高等学校—教材 Ⅳ.TP393
中国版本图书馆 CIP 数据核字(2003)第 083318 号

书　　名：网络编程与开发技术
编　　著：殷肖川　刘志宏　姬伟锋　万映辉
策划编辑：贺峰涛　屈晓燕
文字编辑：邹　林
出版发行：西安交通大学出版社
地　　址：西安市兴庆南路 25 号(邮编：710049)
网　　址：http://unit.xjtu.edu.cn/unit/jtupress
电　　话：(029)2668357　2667874(发行部)
　　(029)2668315　2669096(总编部)
电子信箱：eibooks@163.com
印　　刷：陕西宝石兰印务有限责任公司
版　　次：2003 年 9 月第 1 版　2003 年 9 月第 1 次印刷
开　　本：787mm×1092mm　1/16
印　　张：20
印　　数：0 001～3 000
字　　数：483 千字
书　　号：ISBN7-5605-1750-1/TP·355
定　　价：28.00 元

前　言

近年来，计算机网络技术得到了快速发展，日益增长的网络应用需要大量的熟悉网络应用程序设计的人才。同时，只有通过网络编程，才能在更深的层次去理解网络通信协议的工作原理，并在此基础上进行各种网络应用程序的开发。

网络编程是利用网络编程接口来编写网络信息交换的应用程序。本书主要介绍常规网络编程方法、底层网络编程方法、并发程序设计方法以及相关的 Win32 高级编程技术。常规网络编程接口主要包括 NetBIOS 和 Winsock 编程接口；底层网络编程主要讨论了在网络层和数据链路层的编程方法；并发程序设计主要介绍利用多线程结构实现高性能并发网络程序设计的方法。另外，本书还介绍了进程间通信及动态链接库等相关开发技术。

全书共分为 7 章。第 1 章是网络编程基础，介绍计算机网络程序设计的一些基础知识及内容、概念和方法。第 2 章介绍基于 NetBIOS 的网络编程方法，主要介绍基于 NetBIOS 编程的原理、方法和常用命令，详细讨论了 NetBIOS 编程模型、编程技巧及一些基本程序。第 3 章介绍基于 Winsock 的网络编程技术，主要介绍 TCP/IP 协议的原理，Winsock 基本概念与编程接口，Winsock API 接口函数、数据结构及编程方法，并给出了大量的编程实例。第 4 章是进程间通信机制(IPC)，根据作者以往的编程经验，介绍了在实际网络程序设计中通常会使用到的 IPC 方法，重点讨论了利用文件映射的共享内存机制实现进程间通信的原理和方法。第 5 章是多线程结构的网络编程，主要介绍线程的基本概念、线程同步机制及并发程序设计方法，并给出多线程并发网络通信程序实例。第 6 章介绍动态链接库的概念、作用、设计方法以及在实际工程中的应用，并给出了编程实例。第 7 章讲述直接网络编程。由于网络技术的飞速发展，网络复杂性和规模的不断增长，还需要有一些网络工具来分析、诊断和测试网络，并确保网络的安全。而这些网络诊断测试和安全工具通常需要在较低的层次(链路层或网络层)操作网络。本章主要介绍一些能直接对网络底层进行编程的方法。

本书可作为大学本科及以上层次计算机及相关专业的教科书，也可以作为工程技术人员计算机培训教程，参考学时为 50～60 学时。建议第 7 章作为本科层次的选学内容。学习本门课程之前，读者最好已掌握有关 C 语言程序设计、计算机网络、操作系统等方面的知识。因此，建议在大学三年级下学期开设本课程。本书也可作为从事网络研究和软件开发人员的自学教材和技术参考书。

网络编程与开发是一门实践性很强的技术，因此需要读者通过大量的上机练习来理解网络编程概念和程序设计方法。本书在介绍每一部分内容时尽量做到实用易懂，并提供了大量的编程实例，所有程序均在 VC++ 6.0 下调试通过。每章附有一定量实践性的习题可以选

用，建议提供20学时的上机实验课时。

西安交通大学的李增智教授、唐亚哲副教授对本书的全部书稿进行认真、细致的审阅，并提出了许多宝贵的修改意见，在此表示诚挚的感谢！

由于编者水平有限，书中难免存在一些缺点和错误，敬请广大读者批评指正。

编　者

2003年7月1日

目　录

第1章 网络编程基础

1.1 概述

网络编程就是利用网络应用编程接口编写网络应用程序，实现网络应用进程间的信息交互功能。一般来说，应用进程间的通信可以分为两种：同一系统上的应用进程间的通信和不同系统上的应用进程间的通信。同一系统上的应用进程间的通信又称为进程间通信。而不同系统上的进程间的通信，则必须通过网络编程接口访问网络协议提供的服务来实现。事实上，同一系统上的不同应用进程间的通信也可以通过网络编程接口来实现，只是性能上会有些差别。

网络通信离不开网络协议，网络编程接口访问网络协议所提供的服务。不同的网络协议可能提供不同的服务访问接口，同一网络应用编程接口可能提供访问不同网络协议的接口。如著名的网络应用编程接口——Socket API，支持对很多协议的访问，如 TCP(传输控制协议)，UDP(用户数据报协议)，rawIP，数据链路层协议及 UNIX 域协议等。

要学好网络编程及相关开发技术，对于操作系统、网络协议、网络编程模式和方法以及并发程序设计技术要有比较深入的理解，因为网络应用编程与它们是密不可分的。本书介绍了为实现不同系统上的进程间的通信而进行的网络应用编程的原理、接口和方法，详细讨论了在 Windows 环境下的各种网络编程接口和网络通信程序设计与开发技术。

1.2 ISO/OSI 模型

开放系统互连(OSI)模型是国际标准化组织(ISO)从一个很高的层次对网络系统进行的描述。OSI 模型总共包含了 7 层，从最顶部的“应用层”开始，一直到最底部的“物理层”，这 7 个层完整阐述了最基本的网络概念，表 1.1 展示的正是 OSI 模型的样子。在这个模型中，相邻层之间的关系称为接口，而不同网络实体间在相同层之间的关系就是网络通信协议。

表 1.1 OSI 模型

层	功能描述
应用层	为用户提供相应的界面，以便使用提供的连网功能
表示层	完成数据的格式化

续表

层	功能描述
会话层	控制两个主机间的通信链路(开放、操作和关闭)
传输层	提供数据传输服务(可靠或不可靠)
网络层	在两个主机之间提供一套定址/寻址机制,同时负责数据包的路由选择
数据链路层	控制两个主机间的物理通信链路,同时还要负责对数据进行整形,以便在物理媒体上传输
物理层	物理媒体负责以一系列电子信号的形式传出数据

1.3 网络编程接口

从理论上讲,在网络体系结构中的任何一层上都应能提供应用程序设计的编程接口,但是实际上并非如此。在完整的计算机网络系统中,仅提供了基于网络操作系统之上的编程接口。例如 Windows 的 Winsock ,Netware 的 IPX/SPX 及 NetBIOS(Network Basic Input/Output System, 网络基本输入/输出系统)等。在这些接口上进行网络通信程序设计是最常用的方法。图 1.1 给出了应用程序与编程接口的关系。

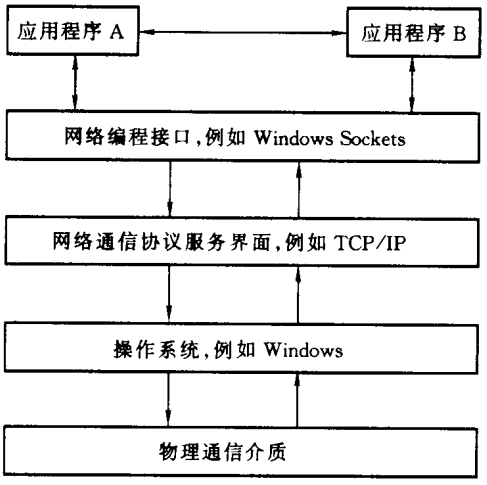

图 1.1 应用程序与网络编程接口关系图

1.3.1 基于 NetBIOS 的网络编程

NetBIOS 是一种标准的应用程序编程接口(API),1983 年由 Sytek 公司专为 IBM 开发成功。NetBIOS 为网络通信定义了一种编程接口。1985 年,IBM 创制了 NetBIOS 扩展用户接口(NetBIOS Extended User Interface, NetBEUI),它同 NetBIOS 接口集成在一起,终于构成了一套完整的协议。NetBIOS 与 TCP/IP 和 IPX/SPX 相比,具有极高的通信效率,尤其在一个小的局域网环境下的实时网络通信,NetBIOS 是首选的网络编程接口。由于 NetBIOS 接口变得愈来愈流行,所以各大厂商也开始在其他如 TCP/IP 和 IPX/SPX 的协议上实施 Net-

BIOS 编程接口。

NetBIOS 同时提供了“面向连接”的会话服务以及“无连接”的数据报服务。NetBIOS 编程接口对应于 OSI 模型的会话层和传输层。值得注意的是 NetBIOS 并非是一种“可路由”协议，假定在客户机和服务器之间需要路由，那么这种协议在两部机器上的通信应用便无法沟通。

1.3.2　基于 Winsock 的网络编程

TCP/IP(Transmission Control Protrocl/Internet Protocol，传输控制协议/网际协议)是发展至今最成功的通信协议之一。它起源于 20 世纪 60 年代末美国政府资助的一个分组交换网络研究项目 ARPRANET，其目的是允许分布在各地的装着完全不同的操作系统的计算机互相通信。TCP/IP 以其开放性的特点，成了 Internet 的基础，通过 Internet 把全世界数以千万的计算机连接在一起，是事实上的工业标准。

Winsock 是 Windows 环境下实现 TCP/IP 网络编程的接口。Winsock 规范并定义了如何使用 API 与 Internet 协议簇，它不仅提供了一套简单的 API，如人们所熟悉的 Berkeley Socket 风格的库函数；也包含了一组针对 Windows 的扩展库函数，以使程序员能充分地利用 Windows 消息驱动机制进行编程。Windows Sockets 规范的本意在于提供给应用程序开发者一套简单的 API，并让各家网络软件供应商共同遵守。尤其要指出的是，所有的 Windows Sockets 实现都支持流套接字和数据报套接字。

1.3.3　直接网络编程

常规网络编程接口一般无法访问到位于底层的网络协议。NetBIOS 主要在会话层和传输层发挥作用，而 Winsock 工作在 TCP/IP 协议的传输层，两者都无法直接对传输层以下的网络协议进行直接操作。随着计算机网络复杂性和规模的不断增长，网络设计和维护人员面临的困难在成倍增加，网络的安全性也正面临着严峻的考验。因此，需要有一些网络工具来分析、诊断和测试网络，以确保网络的安全。而这些网络诊断测试和安全工具不能采用常规的网络编程方法，它们通常需要在较低的层次(链路层或网络层)操作网络。

直接网络编程接口提供在链路层或网络层的编程方法，直接网络编程与普通的网络编程不同，在程序中会涉及底层网络协议及其协议数据单元。因此，需要对链路层帧与网络协议数据单元的结构有一定的了解。本书将介绍 Winpcap 的网络数据包捕获技术、libnet 的网络数据包构造技术以及原始套接字等直接网络编程方法。

1.3.4　基于物理设备的网络编程

基于物理设备的网络编程接口也称为 MAC 层编程接口。这种编程接口主要用于特殊目的的网络程序设计，例如网络数据包截获、网络协议分析、流量统计分析等，或设计自己的安全协议。在这种编程接口上进行网络编程设计，需要对网卡的网络接口控制器(NIC)进行程序编程控制。由于这一接口没有提供现成的程序接口，因此所有功能的实现都必须自行设计。程序设计人员一方面可以根据具体应用设计出适合特殊要求的协议，可以进行更底层的控制，但另一方面也加大了程序设计的难度。

1.4 网络通信方式

1.4.1 面向连接的通信和无连接通信

通常情况下，一个协议提供面向连接（会话）和无连接（数据报）两种通信服务。如 NetBIOS 和 TCP/IP 协议均提供了这两种服务。

在面向连接的服务中，通信双方在进行数据交换之前必须建立一条路径，这样既确定了通信方之间存在路由，又保证了通信双方都是活动的、可以彼此响应的、并且传送数据时是按序传送的，从而保证数据通信的可靠性。一般来说，面向连接服务过程分为三个阶段：连接建立、数据传输和连接释放。面向连接服务比较适合于在一段时间间隔内要向同一目的地发送许多报文的情况。对于发送很短的零星报文，连接建立和释放所带来的开销就显得过大了。面向连接的服务的缺点是建立和维持一个通信信道需要很多开销。此外，为保证数据通信的可靠性而设置的验证机制，会进一步增加额外的开销。

无连接服务却不用确定接收端是否正在收听，发送方随时可以发送数据。无连接协议的特点是速度快、使用灵活，既能实现点到点通信，又能实现多点和广播通信，对非重要的数据传输来说，数据报服务非常有用。但其缺点是不能保证数据可靠到达接收端，同样也不能保证数据的完整性，这些问题必须由应用程序根据需要自行解决。无连接服务类似于邮政服务，发信人把信装入邮箱即可，至于收信人是否能收到这封信或邮局是否按时将信件投递到收信人处，发信人都不得而知。无连接服务特别适合于传送少量零星的报文。

1.4.2 阻塞通信与非阻塞通信

网络通信可以分为阻塞和非阻塞两种模式。在网络编程时，选择通信模式也是一件很重要的事情。对于不同的协议，阻塞通信和非阻塞通信有不同的表现。

以 Winsock 为例，在阻塞模式下，利用 TCP 协议发送一个报文时，如果低层协议没有可用空间来存放用户数据，则应用进程将阻塞，直到协议有可用的空间。而在非阻塞模式下，调用将直接返回而不需等待。在应用进程调用接收函数接收报文时，如果是在阻塞模式下，若没有到达的数据，则调用将一直阻塞直到有数据到达或出错；而在非阻塞模式下，将直接返回而不需等待。

对于面向连接的协议，在连接建立阶段，阻塞与非阻塞也表现不一。在阻塞模式下，如果没有连接请求到达，则等待连接调用将阻塞直到有连接请求到达；但在非阻塞模式下，如果没有连接请求到达，等待连接调用将直接返回。

在连接建立阶段，不管是阻塞模式还是非阻塞模式，发起连接请求的一方总是会使调用它的进程阻塞，阻塞间隔最少等于到达服务器的一次往返时间。

通信模式对应用程序的设计方法也有直接的影响。在非阻塞模式下，应用程序不断地轮询查看是否有数据到达或有连接请求到达。这种轮询方式比其他的技术耗费更多的 CPU 时间，因此要尽量避免使用。而在阻塞模式下，不存在这一问题。但是阻塞模式的缺点是进程或线程在执行 I/O 操作时将被阻塞而不能执行其他的工作，因此在单进程或单线程应用中不能使用这种模式。在多线程应用中比较适合采用阻塞模式，一个线程被阻塞不影响其他线程的工作。

1.4.3　多播通信与广播通信

广播通信是指数据从一个工作站发出，局域网内的其他所有工作站都能收到它。这一特征适用于无连接协议，因为 LAN(局域网)上的所有机器都可获得并处理广播消息。使用广播消息的不利之处是每台机器都必须对该消息进行处理。比如，一用户在 LAN 上广播一条消息，每台机器上的网卡都会收到这条消息，并把它上传到协议栈。然后，协议栈将这条消息在所有的网络应用中循环，看它们是否应该接收这条消息。通常，这个局域网上的多数机器对该消息都不感兴趣，草草地一弃了之。但是各台机器仍需花时间来处理这个数据包，看是否有应用对它感兴趣。结果，高广播通信流会使 LAN 上的机器陷入困境。一般情况下，路由器都不会传送广播包。

多播是指数据从一个工作站发出，这些数据将由一个或多个接收端进行接收。进程加入一个多播通信的方法与采用的底层协议有关。在 IP 协议和 NetBIOS 协议下，多播是广播的一种变形，多播要求对收发数据感兴趣的所有主机加入到一个特定的组，进程希望加入多播组时，网卡上会增添一个过滤器。这样，只有绑定组地址的数据才会被网络硬件捡起，并上传到网络堆栈进行相应处理。

无论是多播还是广播通信，它们都是建立在无连接服务协议之上的，因此数据传输的可靠性无法得到保证。

1.5　Win32 SDK 编程基础

1.5.1　Win32 SDK 的基本概念

在 VC 环境下开发应用程序有 MFC 和 SDK 两种方式。

MFC 是对 Win32 API 进行彻底封装的一个类库，可以采用面向对象的方法进行 Windows 应用程序的开发，开发效率较高，但其生成的程序规模较大，不易理解，而且 MFC 方式隐藏了大量的编程细节，不利于读者对网络通信机制的深入理解。

SDK 是一套软件开发包，内部包含了大量的 Win32 API 函数。借助 Win32 SDK 能够方便灵活地开发网络应用程序。我们后面要讲到的 Winsock API 就是 SDK 的一部分。SDK 程序结构清晰、执行效率高，而且编程更加灵活。鉴于 SDK 方式的特点，为了更方便读者对网络通信机制和编程方法的理解与掌握，本书所列例程均采用 SDK 方式实现。

下面将简单介绍 Windows 消息驱动机制以及 SDK 程序结构。

1.5.2　Windows 消息驱动机制

在 Windows 系统的运行机制中，消息是一个基本而重要的概念，它是指一个报告有关事件发生的通知。消息驱动是围绕着消息的产生与处理展开的，并依靠消息循环机制来实现。

从编程角度来看，一条消息可被视为某一事件的发生，如按下或移动鼠标、按键盘键等。这些事件既可以由用户引发，也可以由应用程序产生，当然 Windows 系统本身也能发出消息。Windows 应用程序的消息来源有以下 4 种：

(1) 输入消息

包括键盘和鼠标的输入。这一类消息首先放在系统消息队列中，然后 Windows 将它们送

入应用程序消息队列，最后再由应用程序来处理。

(2) 控制消息

用来与 Windows 的控制对象(如列表框、按钮、检查框等)进行双向通信。当用户在列表框中改动当前选择或改变了检查框的状态时发出此类消息。这类消息一般不经过应用程序消息队列，而是直接发送到控制对象上去。

(3) 系统消息

对程序化的事件或系统时钟中断作出反应。一些系统消息，如 DDE(Dynamic Data Exchange，动态数据交换)消息要直接通过 Windows 的系统消息队列，而有的则不通过系统消息队列而直接进入应用程序的消息队列，如创建窗口消息。

(4) 用户消息

这是程序员自己定义并在应用程序主动发出的，一般由应用程序的某一部分进行内部处理。

消息产生后，应该送给相应的应用程序，应用程序是如何获得属于自己的消息呢？实际上，Windows 操作系统时时刻刻监视着用户的一举一动，并分析用户的动作与哪一个应用程序相关，然后将该动作以消息的形式发送给当前应用程序。Windows 为每个正在运行的应用程序都维护一个消息队列，应用程序时刻等待着消息的到来，一旦发现它的消息队列中有未处理的消息，就获取并分析该消息，并根据消息内容采用适当的动作来响应用户的操作。所以，应用程序需要不停地从特定的消息队列中获取消息、分析消息并处理消息，直到接收到一条叫做 WM_QUIT 的消息为止。实现这个过程的程序结构就叫“消息循环”。

1.5.3 Win32 SDK 程序结构

在 VC 中，Windows SDK 应用程序有两种工程类型：Win32 Application 和 Win32 Console Application。这两者在程序结构上最大的区别在于 Win32 Application 以 WinMain 函数作为程序的入口点，而 Win32 Console Application 以 main 函数为程序的入口点。对于 main 函数，大家都非常熟悉，这里只重点介绍 WinMain 函数及 Win32 Application 程序结构。

WinMain()函数的原型如下：

```
int WINAPI WinMain
(
  HINSTANCE     hInstance,        //当前实例句柄
  HINSTANCE     hPrevInstance,    //前一个实例句柄
  LPSTR         lpCmdLine,        //指向命令行参数的指针
  int           nCmdShow          //窗口的显示状态
);
```

这里出现的句柄是一个用于标识对象的变量。实例是类中一特定对象类型的一个实例化对象，比如一个特定的进程和线程。

一般情况下，在 WinMain 函数中应完成的操作有：

- 注册窗口类；
- 创建应用程序主窗口；
- 进入应用程序消息循环。

除了 WinMain 函数作为入口点函数外，Windows 程序还有一个很重要的窗口过程函数

WndProc,它的函数原型为:

```
long FAR PASCAL WndProc(HWND hWnd,WORD message,
                        WORD wParam,LONG lParam);
```

WndProc 函数的作用在后面会详细介绍。

下面详细介绍 Windows 程序的一般构成。通常一个 Windows 程序包括以下 7 个部分:

- 注册窗口类;
- 创建窗口;
- 显示和更新窗口;
- 创建消息循环;
- 终止应用程序;
- 窗口过程和窗口函数;
- 处理消息。

(1)注册窗口类

WinMain 既然作为程序的入口,它相当于一个中介人的角色,把应用程序介绍给 Windows 系统,所以第一步是建立和登记应用程序的窗口类。建立窗口类就是用 WNDCLASS 结构定义一个结构变量,如:

```
WNDCLASS wc;
```

然后用自己设计的窗口属性的信息填充结构变量 wc 的各个域。WNDCLASS 结构定义如下:

```
typedef struct _WNDCLASS
{
    UINT        style;          //窗口类风格
    WNDPROC     lpfnWndProc;    //指向窗口过程函数的指针
    int         cbClsExtra;     //窗口类附加数据
    int         cbWndExtra;     //窗口附加数据
    HINSTANCE   hInstance;      //拥有窗口类的实例句柄
    HICON       hIcon;          //最小窗口图标
    HCURSOR     hCursor;        //窗口内使用的光标
    HBRUSH      hbrBackground;  //用来着色窗口背景的刷子
    LPCSTR      lpszMenuName;   //指向菜单资源名的指针
    LPCSTR      lpszClassName;  //指向窗口类名的指针
}WNDCLASS, * PWNDCLASS;
```

对 WNDCLASS 结构的各域一一赋值后,就可以注册窗口类了。注册窗口类使用的函数是 RegisterClass,定义如下:

```
ATOM RegisterClass(CONST WNDCLASS * lpWndClass);
```

注册失败,则函数 RegisterClass 将返回 0,程序不能再进行下去。

(2)创建窗口

注册窗口类后,就可以创建窗口了。创建窗口使用的函数是 CreateWindow,形式如下:

```
HWND CreatWindow
```