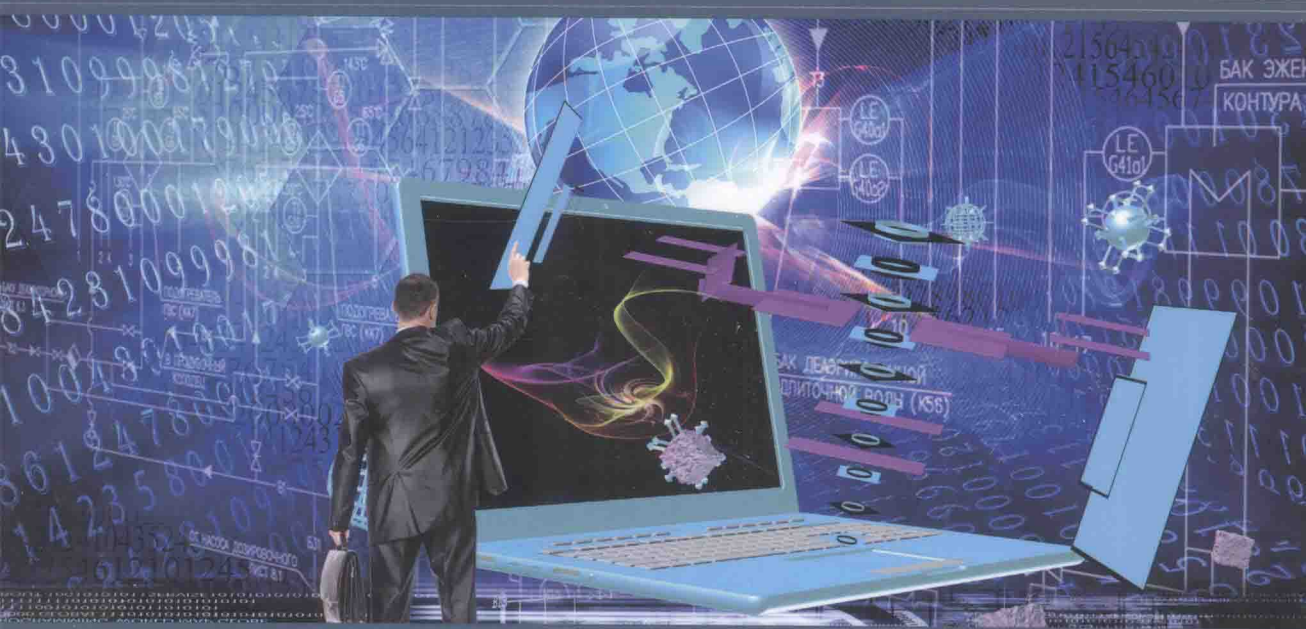
21世纪高等教育网络工程规划教材
21st Century University Planned Textbooks of Network Engineering

网络编程实用教程

（第3版）

Network Application Programming (3rd Edition)

段利国◎主编
刘金江 倪天伟 叶树华◎副主编



中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

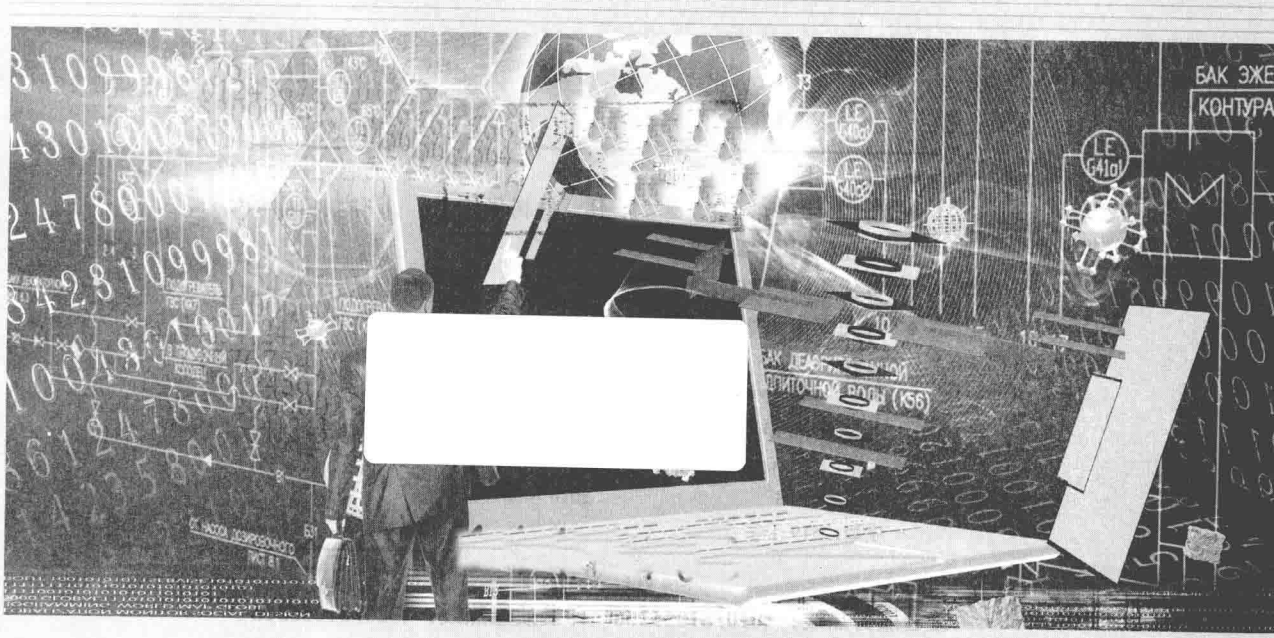
21世纪高等教育网络工程规划教材
21st Century University Planned Textbooks of Network Engineering

网络编程实用教程

（第3版）

Network Application Programming (3rd Edition)

段利国◎主编
刘金江 倪天伟 叶树华◎副主编



人民邮电出版社
北京

图书在版编目（CIP）数据

网络编程实用教程 / 段利国主编. -- 3版. -- 北京：人民邮电出版社，2016.6
21世纪高等教育网络工程规划教材
ISBN 978-7-115-42341-2

Ⅰ. ①网… Ⅱ. ①段… Ⅲ. ①计算机网络－程序设计－高等学校－教材 Ⅳ. ①TP393

中国版本图书馆CIP数据核字(2016)第085718号

内 容 提 要

本书主要介绍基于 TCP/IP 协议栈的套接字网络编程技术。全书分为 10 章，第 1 章介绍网络编程的基本概念及模式，第 2 章介绍套接字网络编程基础，第 3 章介绍 Windows 环境的 WinSock 编程基础，第 4 章介绍 MFC 编程，第 5 章介绍 MFC WinSock 类的编程，第 6 章介绍 WinInet 编程，第 7 章介绍 WinSock 的多线程编程，第 8 章介绍 WinSock 的 I/O 模型，第 9 章介绍 HTTP 及高级编程，第 10 章介绍电子邮件协议与编程。各章后都配有习题，便于读者理解掌握所学内容。

本书配有教学资源包，包括课件和各章实例的源程序，可以从人民邮电出版社教学资源与服务网上免费下载。下载的地址为 http://www.ptpedu.com.cn。

本书可作为高等学校相关专业高年级本科生和研究生的教材，也可供其他技术人员参考。

◆ 主　　编　段利国
副 主 编　刘金江　倪天伟　叶树华
责任编辑　邹文波
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京中新伟业印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：19.75　　2016 年 6 月第 3 版
字数：519 千字　　2016 年 6 月北京第 1 次印刷

定价：52.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广字第 8052 号

第 3 版前言

基于 TCP/IP 协议栈的套接字网络编程技术，是网络编程的核心技术。读者在学习了计算机网络体系结构原理之后，只有掌握套接字编程，才能更深入地了解和运用计算机网络。作者结合自己多年讲授这门课程的体会，在讲义的基础上，又搜集了大量的资料，编写了本书。

第 3 版在充分考虑读者反馈意见的基础上，对第 2 版进行了一些修订。删除了对网络编程帮助不大却反而会增加读者学习负担的内容，如 Linux 系统的网络编程接口、Windows Sockets 的不同版本等内容，使得内容更加简洁；删除了一些前后重复的内容，如套接字编程接口的系统调用等，使得内容更加紧凑；修订了一些词法、语法错误，使得内容更加准确；将编程环境从 VC++ 6.0 迁徙到了 Visual Studio 2015，进一步完善了第 5 章、第 6 章、第 7 章、第 9 章、第 10 章的例程，并在 Windows 10 操作系统、Visual Studio 2015 环境下调试通过。

全书分为 10 章，具体内容如下。

第 1 章介绍网络编程相关的基本概念，目前网络编程的现状，以及网络应用程序的编程模式。

第 2 章介绍套接字网络编程接口的产生和发展，套接字编程的基本概念，以及面向连接与无连接的套接字编程。

第 3 章详细说明 Windows Sockets 规范。

第 4 章介绍 MFC 编程框架，MFC 对象和 Windows 对象的关系，主要的 MFC 类和基类，以及 MFC 的消息驱动机制。

第 5 章介绍 MFC 中的 CAsyncSocket 类和 CSocket 类的使用。

第 6 章介绍 MFC WinInet 类的使用。

第 7 章说明 Win32 操作系统下的多进程多线程机制，VC++ 对多线程网络编程的支持，以及 MFC 多线程编程的步骤。

第 8 章介绍非阻塞套接字工作模式下的 5 种套接字 I/O 模型。

第 9 章介绍 HTTP 和 MFC 中的 CHtmlView 类的使用。

第 10 章介绍电子邮件系统的构成和工作原理、SMTP、纯文本电子邮件信件的格式、多媒体邮件格式扩展（MIME）、邮局协议（POP3），并通过编程实例说明了在网络编程中实现应用层协议的方法。

本书的特点如下。

（1）强调知识点的内在逻辑结构。内容安排由浅入深，循序渐进，以适合教学的顺序全面地介绍了套接字网络编程的理论和应用知识。

（2）特别强调知识与能力的结合，理论与实用并重，各章有大量的编程实例，力图培养学生运用网络编程技术的实践能力，使学生能深入地运用套接字编写各种类型的网络应用程序。

（3）强调掌握网络应用层协议在网络编程中的重要性。网络编程就是网络应用协议的实现，本书力图培养学生迅速掌握网络协议，甚至自己开发网络协议的能力。

（4）强调编程技术与计算机网络体系结构原理的结合。

阅读本书的读者应学习过计算机网络体系结构的原理，以及VC++面向对象编程的知识。

本书建议学时为40学时，各章学时分配如下：

章	学时数	章	学时数
第1章	3	第6章	4
第2章	4	第7章	4
第3章	4	第8章	3
第4章	4	第9章	4
第5章	6	第10章	4

由于编者水平所限，书中难免存在一些缺点和错误，殷切希望广大读者批评指正。编者的邮箱是TYUTDLG@163.com。

编　者

2016年5月

目 录

第 1 章 概述

本章首先介绍网络编程相关的基本概念，重点分析进程通信、Internet 中网间进程的标识方法以及网络协议的特征。接着从网络编程的角度，分析 TCP/IP 协议簇中高效的用户数据报协议（UDP）和可靠的传输控制协议（TCP）的特点。最后详细说明网络应用程序的客户机/服务器交互模式。

透彻地理解这些网络编程相关的基本概念十分重要，能使我们的思路从过去所学的网络的构造原理转移到对网络的应用的层面上来，为理解后续章节的内容打下基础。

1.1 网络编程相关的基本概念

1.1.1 网络编程与进程通信

1. 进程与线程的基本概念

进程是操作系统理论中最重要的概念之一，简单地说，进程是处于运行过程中的程序实例，是操作系统调度和分配资源的基本单位。

一个进程实体由程序代码、数据和进程控制块 3 部分构成。程序代码规定了进程所做的计算；数据是计算的对象；进程控制块是操作系统内核为了控制进程所建立的数据结构，是操作系统用来管理进程的内核对象，也是系统用来存放关于进程的统计信息的地方。系统给进程分配一个地址空间，用来装入进程的所有可执行模块或动态链接库（Dynamic Linking Library，DLL）模块的代码和数据。进程还包含动态分配的内存空间，如线程堆栈和堆分配空间。多个进程可以在操作系统的协调下，在内存中并发地运行。

各种计算机应用程序在运行时，都以进程的形式存在。网络应用程序也不例外。我们在 Windows 操作系统中，有时打开多个 IE 浏览器的窗口访问多个网站，有时运行 Foxmail 电子邮件程序查看自己的邮箱，有时运行网际快车软件下载文件。它们都会在 Windows 的桌面上打开一个窗口；每一个窗口中运行的网络应用程序，都是一个网络应用进程。网络编程就是要开发网络应用程序，所以了解进程的概念是非常重要的。

Windows 系统不但支持多进程，还支持多线程。在 Windows 系统中，进程是分配资源的单位，但不是执行和调度的单位。若要使进程完成某项操作，它必须拥有一个在它的环境中运行的线程，该线程负责执行包含在进程的地址空间中的代码。实际上，单个进程可能包含若干个线程，所有这些线程都“同时”执行进程地址空间中的代码。为此，每个线程都有它自己的一组 CPU 寄存器

和它自己的堆栈。每个进程至少拥有一个线程，来执行进程的地址空间中的代码。如果没有线程来执行进程的地址空间中的代码，那么进程就没有存在的理由了，系统就会自动撤销该进程和它的地址空间。若要使所有这些线程都能运行，操作系统就要为每个线程安排一定的CPU时间。它通过一种循环方式为线程提供时间片（称为量程），造成一种假象，仿佛所有线程都是同时运行的一样。

当创建一个进程时，系统会自动创建它的第一个线程，称为主线程。然后，该线程可以创建其他的线程，而这些线程又能创建更多的线程。

图1.1所示为在单CPU的计算机上，CPU分时地运行各个线程。如果计算机拥有多个CPU，那么操作系统就要使用更复杂的算法来实现CPU上线程负载的平衡。

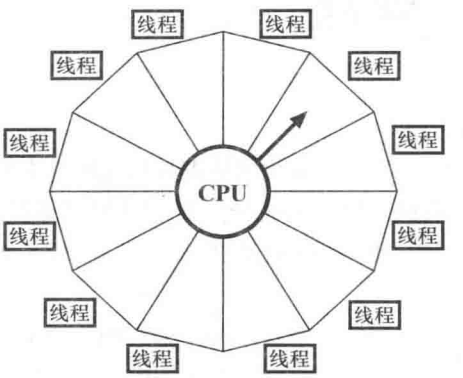


图1.1　单CPU分时地运行进程中的各个线程

Windows操作系统（2000以上的版本）能够在拥有多个CPU的计算机上运行，可以在每个CPU上运行不同的线程，这样，多个线程就真的在同时运行了。Windows 2000操作系统的内核能够在这种类型的系统上进行所有线程的管理和调度，不必在代码中进行任何特定的设置，就能利用多处理器提供的各种优点。Windows 98操作系统只能在单处理器计算机上运行；即使计算机配有多个处理器，Windows 98操作系统每次也只能安排一个线程运行，而其他处理器则处于空闲状态。

2. 网络应用进程在网络体系结构中的位置

从计算机网络体系结构的角度来看，网络应用进程处于网络层次结构的最上层。图1.2所示为网络应用程序在网络体系结构中的位置示意图。

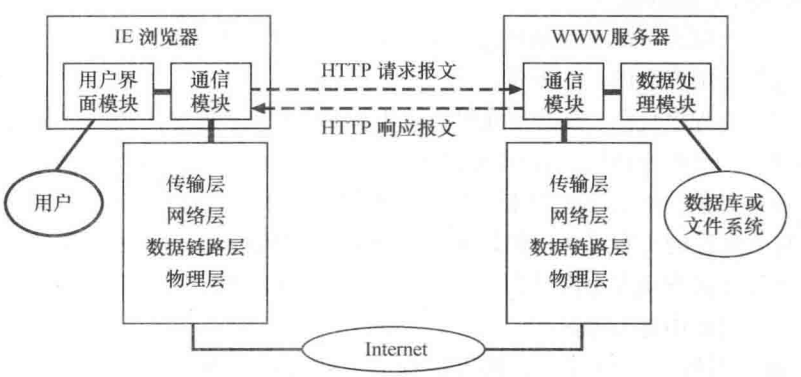


图1.2　网络应用程序在网络体系结构中的位置示意图

从功能上可以将网络应用程序分为两部分。一部分是专门负责网络通信的模块；它们与网络协议栈相连接，借助网络协议栈提供的服务完成网络上数据信息的交换；另一部分是面向用户或者进行其他处理的模块，它们接收用户的命令，或者对借助网络传输过来的数据进行加工。这两部分模块相互配合，来实现网络应用程序的功能。例如，在图1.2中，IE浏览器就分为两部分：用户界面部分接收用户输入的网址，把它转交给通信模块；通信模块按照网址与对方连接，按照HTTP和对方通信，接收服务器发回的网页，然后把它交给浏览器的用户界面部分。用户界面模

块解释网页中的超文本标记，把页面显示给用户。服务器端的 Internet 信息服务（Internet Information Server，IIS）软件，其实也分为两部分，通信模块负责与客户端进行通信，另一部分负责操作服务器端的文件系统或数据库。

要注意网络应用程序这两部分的关系。通信模块是网络分布式应用的基础，其他模块则对网络交换的数据进行加工处理，从而满足用户的种种需求。网络应用程序最终要实现网络资源的共享，共享的基础就是必须能够通过网络轻松地传递各种信息。

由此可见，网络编程首先要解决网间进程通信的问题，然后才能在通信的基础上开发各种应用功能。

3. 实现网间进程通信必须解决的问题

进程通信的概念最初来源于单机系统。由于每个进程都在自己的地址范围内运行，为了保证两个相互通信的进程之间既不互相干扰，又能协调一致地工作，操作系统为进程通信提供了相应的设施。例如，UNIX 系统中的管道（Pipe）、命名管道（Named Pipe）和软中断信号（Signal），UNIX System V 中的消息（Message）、共享存储区（Shared Memory）和信号量（Semaphore）等，但它们都仅限于用在本机进程之间的通信上。

网间进程通信是指网络中不同主机中的应用进程之间的相互通信，当然，可以把同机进程间的通信看作网间进程通信的特例。网间进程通信必须解决以下问题。

（1）网间进程的标识问题。在同一主机中，不同的进程可以用进程号（Process ID）唯一标识。但在网络环境下，各主机独立分配的进程号已经不能唯一地标识一个进程。例如，主机 A 中某进程的进程号是 5，在 B 机中也可以存在 5 号进程，进程号不再唯一了，因此，在网络环境下，仅仅说“5 号进程”就没有意义了。

（2）与网络协议栈连接的问题。网间进程的通信实际是借助网络协议栈实现的。应用进程把数据交给下层的传输层协议实体，调用传输层提供的传输服务，传输层及其下层协议将数据层层向下递交，最后由物理层将数据变为信号，发送到网上，经过各种网络设备的寻径和存储转发，才能到达目的端主机，目的端的网络协议栈再将数据层层上传，最终将数据送交接收端的应用进程，这个过程是非常复杂的。但是对于网络编程来说，必须要有一种非常简单的方法，来与网络协议栈连接。这个问题是通过定义套接字网络编程接口来解决的。

（3）多重协议的识别问题。现行的网络体系结构有很多，如 TCP/IP、IPX/SPX 等，操作系统往往支持众多的网络协议。不同协议的工作方式不同，地址格式也不同，因此网间进程通信还要解决多重协议的识别问题。

（4）不同的通信服务的问题。随着网络应用的不同，网间进程通信所要求的通信服务就会有不同的要求。例如，文件传输服务，传输的文件可能很大，要求传输非常可靠，无差错，无乱序，无丢失，无重复；下载了一个程序，如果丢了几个字节，这个程序可能就不能用了。但对于网上聊天这样的应用，要求就不高。因此，要求网络应用程序能够有选择地使用网络协议栈提供的网络通信服务功能。在 TCP/IP 协议簇中，在传输层有 TCP 和 UDP 这两个协议，TCP 提供可靠的数据流传输服务，UDP 提供不可靠的数据报传输服务。深入了解它们的工作机制，对于网络编程是非常必要的。

以上问题的解决方案将在后续章节中详细讲述。

1.1.2　Internet 中网间进程的标识

1. 传输层在网络通信中的地位

图 1.3 所示为基于 TCP/IP 协议栈的进程之间的通信情况。

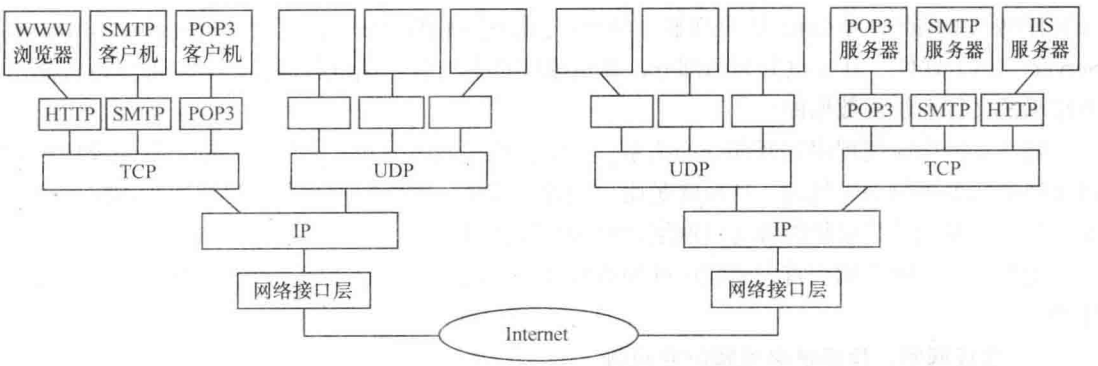


图 1.3　基于 TCP/IP 协议栈的进程间的通信

Internet 是基于 TCP/IP 协议栈的，TCP/IP 协议栈的特点是两头大，中间小。在应用层，有众多的应用进程，分别使用不同的应用层协议；在网络接口层，有多种数据链路层协议，可以和各种物理网相接；在网络层，只有一个 IP 实体。在发送端，所有上层的应用进程的信息都要汇聚到 IP 层；在接收端，下层的信息又从 IP 层分流到不同的应用进程。

网络层的 IP，在 Internet 中起着非常重要的作用。它用 IP 地址统一了 Internet 中各种主机的物理地址，用 IP 数据报统一了各种物理网的帧，实现了异构网的互连。粗略地说，在 Internet 中，每一台主机都有一个唯一的 IP 地址，利用 IP 地址可以唯一地定位 Internet 中的一台计算机，实现计算机之间的通信。但是最终进行网络通信的不是整个计算机，而是计算机中的某个应用进程。每个主机中有许多应用进程，仅有 IP 地址是无法区别一台主机中的多个应用进程的。从这个意义上讲，网络通信的最终地址就不仅仅是主机的 IP 地址了，还必须包括可以描述应用进程的某种标识符。

按照 OSI 七层协议的描述，传输层与网络层在功能上的最大区别是传输层提供进程通信的能力。TCP/IP 提出了传输层协议端口（Protocol Port，端口）的概念，成功地解决了通信进程的标识问题。

传输层，也称传送层，是对计算机网络中通信主机内部进行独立操作的第一层，是支持端到端的进程通信的关键的一层。如图 1.3 所示，应用层的多个进程通过各自的端口复用 TCP 或 UDP，TCP 或 UDP 再复用网络层的 IP，经过通信子网的存储转发，将数据传送到目的端的主机。而在目的端主机中，IP 将数据分发给 TCP 或 UDP，再由 TCP 或 UDP 通过特定的端口传送给相应的进程。对于网络协议栈来说，在发送端是自上而下地复用，在接收端是自下而上地分用，从而实现了网络中应用进程之间的通信。

2. 端口的概念

端口是 TCP/IP 协议簇中，应用层进程与传输层协议实体间的通信接口，在 OSI 七层协议的描述中，将它称为应用层进程与传输层协议实体间的服务访问点（SAP）。应用层进程通过系统调用与某个端口进行绑定，然后就可以通过该端口接收或发送数据，因为应用进程在通信时，必须用到一个端口，它们之间有着一一对应的关系，所以可以用端口来标识通信的网络应用进程。

类似于文件描述符，每个端口都拥有一个叫作端口号（Port Number）的整数型标识符，用于区别不同的端口。由于 TCP/IP 协议簇传输层的两个协议，即 TCP 和 UDP，是完全独立的两个软件模块，因此各自的端口号也相互独立。如 TCP 有一个 255 号端口，UDP 也可以有一个 255 号端口，二者并不冲突。图 1.4 所示为 UDP 数据报和 TCP 报文段的首部格式。

图 1.4　UDP 与 TCP 的报文格式

图 1.4 所示的上半部分是 UDP 的报头格式，下半部分是 TCP 的报头格式。从 TCP 或 UDP 的报头格式来看，端口标识符是一个 16 位的整数，所以，TCP 和 UDP 都可以提供 65535 个端口，供应用层的进程使用，这个数量是不小的。端口与传输层的协议是密不可分的，必须区别是 TCP 的端口，还是 UDP 的端口，两种协议的端口之间没有任何联系。端口是操作系统可分配的一种资源。

从实现的角度讲，端口是一种抽象的软件机制，包括一些数据结构和 I/O 缓冲区。应用程序（即进程）通过系统调用与某端口建立绑定（Binding）关系后，传输层传给该端口的数据都被相应进程接收，相应进程发给传输层的数据都通过该端口输出。在 TCP/IP 的实现中，端口操作类似于一般的 I/O 操作，进程获取一个端口，相当于获取本地唯一的 I/O 文件，可以用一般的读写原语访问它。

3. 端口号的分配机制

端口号的分配是一个重要问题。

假如网络中两主机的两个进程甲、乙要通信，并且甲首先向乙发送信息，那么，甲进程必须知道乙进程的地址，包括网络层地址和传输层的端口号。IP 地址是全局分配的，能保证全网的唯一性，并且在通信之前，甲就能知道乙的 IP 地址；但端口号是由每台主机自己分配的，只有本地意义，无法保证全网唯一，所以甲在通信之前是无法知道乙的端口号的。这个问题如何解决呢？

由于在 Internet 应用程序的开发中，大多都采用客户机/服务器（C/S）的模式，在这种模式下，客户机与服务器的通信总是由客户机首先发起，因此只需要让客户机进程事先知道服务器进程的端口号就行了。另一方面，在 Internet 中，众所周知的为大家所接受的服务是有限的。基于这两方面的考虑，TCP/IP 采用了全局分配（静态分配）和本地分配（动态分配）相结合的方法。对 TCP 或者 UDP，将它们的全部 65535 个端口号分为保留端口号和自由端口号两部分。

保留端口的范围是 0～1023，又称为众所周知的端口或熟知端口（Well-known Port），只占少数，采用全局分配或集中控制的方式，由一个公认的中央机构根据需要进行统一分配，静态地分配给 Internet 上众所周知的服务器进程，并将结果公布于众。由于一种服务使用一种应用层协议，也可以说把保留端口分配给了一些应用层协议。表 1.1 列举了一些应用层协议分配到的保留端口号。

表 1.1　　一些典型的应用层协议分配到的保留端口

TCP 的保留端口		UDP 的保留端口	
FTP	21	DNS	53
HTTP	80	TFTP	69
SMTP	25	SNMP	161
POP3	110	……	

这样，每一个标准的服务器都拥有了一个全网公认的端口号，在不同的服务器类主机上，使用相同应用层协议的服务器的端口号也相同。例如，所有的 WWW 服务器默认的端口号都是 80，FTP 服务器默认的端口号都是 21。

其余的端口号，1024～65535，称为自由端口号，采用本地分配，又称为动态分配的方法，由每台计算机在网络进程通信时，动态地、自由地分配给要进行网络通信的应用层进程。具体地说，应用进程当需要访问传输层服务时，向本地操作系统提出申请，操作系统返回一个本地唯一的端口号，进程再通过合适的系统调用将自己与该端口号联系起来（绑定），然后通过它进行网络通信。

具体来说，TCP 或 UDP 端口的分配规则如下。

端口 0：不使用或者作为特殊的用途。

端口 1～255：保留给特定的服务。TCP 和 UDP 均规定，小于 256 的端口号才能分配给网上众所周知的服务。

端口 256～1023：保留给其他的服务，如路由。

端口 1024～4999：可以用作任意客户的端口。

端口 5000～65535：可以用作用户的服务器端口。

我们可以描述一下在这样的端口分配机制下，客户机进程 C 与服务器进程 S 第一次通信的情景。图 1.5 所示为客户机与服务器第一次通信的情况。

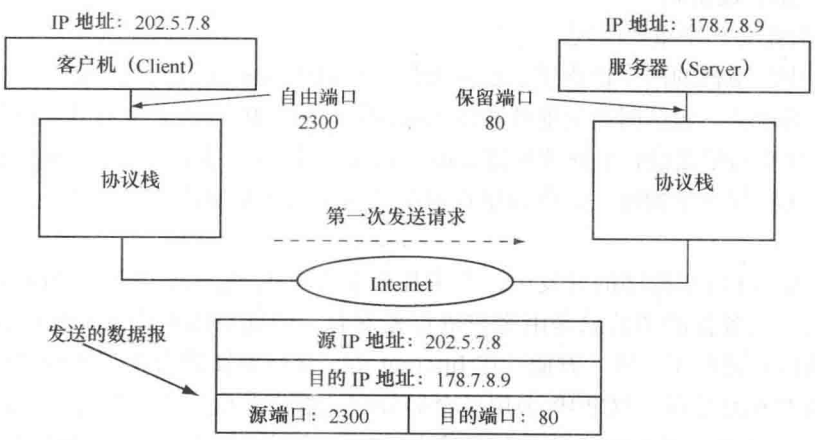


图 1.5　客户机与服务器的第一次通信

Client 进程（简称 C）要与远地 Server 进程（简称 S）通信。C 首先向操作系统申请一个自由端口号，因为每台主机都要进行 TCP/IP 的配置，其中主要的一项就是配置 IP 地址，所以自己的 IP 地址是已知的。C 使用的传输层协议是已经确定的，这样，通信的一端就完全确定了。S 的端口号是保留端口，是众所周知的，C 当然也知道，S 的 IP 地址也是已知的（在客户端输入网址请求访问一个网站的时候，网址当中都包含对方的主机域名），S 采用的传输层协议必须与 C 一致，

这样，通信的另一端也就完全确定了下来，C 就可以向 S 发起通信了。

如此看来，这种端口的分配机制能够保证客户机第一次成功地将信息发送到服务器，但是接着又有另一个问题：服务器进程是要为多个客户机进程服务的，如果当某个客户机第一次成功地连接到服务器后，服务器就接着用这个保留端口继续与该客户机通信，那么其他申请连接的客户机就只能等待了，这就无法实现服务器进程同时为多个客户机服务的要求。但实际的情况是，一个网站的 WWW 服务器，可以同时为千百个人服务，这是怎么回事呢？

原来，在 TCP/IP 的端口号分配机制中，服务器的保留端口是专门用来监听客户端的连接请求的，当服务器从保留端口收到一个客户机的连接请求后，立即创建另外一个线程，并为这个线程分配一个服务器端的自由端口号，然后用这个线程继续与那个客户机进行通信；而服务器的保留端口就又可以接收另一个客户机的连接请求了，这就是所谓“偷梁换柱”的办法。

4. 进程的网络地址的概念

网络通信中通信的两个进程分别是在不同的计算机上。在 Internet 中，两台主机可能位于不同的网络中，这些网络通过网络互连设备（网关、网桥和路由器等）连接。因此要在 Internet 中定位一个应用进程，需要以下三级寻址。

（1）某一主机总是与某个网络相连，必须指定主机所在的特定网络地址，称为网络 ID。

（2）网络上每一台主机应有其唯一的地址，称为主机 ID。

（3）每一主机上的每一应用进程应有在该主机上的唯一标识符。

在 TCP/IP 中，主机 IP 地址就是由网络 ID 和主机 ID 组成的，IPv4 中用 32 位二进制数值表示；应用进程是用 TCP 或 UDP 的 16 位端口号来标识的。

综上所述，在 Internet 中，用一个三元组可以在全局中唯一地标识一个应用层进程：

应用层进程 =（传输层协议，主机的 IP 地址，传输层的端口号）

这样一个三元组，叫作一个半相关（Half-association），它标识了 Internet 中进程间通信的一个端点，也把它称为进程的网络地址。

5. 网络中进程通信的标识

在 Internet 中，一个完整的网间进程通信需要由两个进程组成，两个进程是通信的两个端点，并且只能使用同一种传输层协议。也就是说，不可能通信的一端用 TCP，而另一端用 UDP。因此一个完整的网间通信需要一个五元组在全局中唯一地标识：

（传输层协议，本地机 IP 地址，本地机传输层端口，远地机 IP 地址，远地机传输层端口）

这个五元组称为一个全相关（Association），即两个协议相同的半相关才能组合成一个合适的全相关，或完全指定一对网间通信的进程。

1.1.3　网络协议的特征

在网络分层体系结构中，各层之间是严格单向依赖的，各层次的分工和协作集中体现在相邻层之间的接口上。“服务”是描述相邻层之间关系的抽象概念，是网络中各层向紧邻上层提供的一组服务。下层是服务的提供者，上层是服务的请求者和使用者。服务的表现形式是原语（Primitive）操作，一般以系统调用或库函数的形式提供。系统调用是操作系统内核向网络应用程序或高层协议提供的服务原语。网络中的 n 层总要向 $n+1$ 层提供比 $n-1$ 层更完备的服务，否则 n 层就没有存在的价值。

在 OSI 的术语中，网络层及其以下各层又称为通信子网，只提供点到点通信，没有程序或进程的概念。而传输层实现的是“端到端”通信，引进了网间进程通信的概念，同时也要解决差错

控制、流量控制、报文排序和连接管理等问题，为此，传输层以不同的方式向应用层提供不同的服务。

编程者应了解常用网络传送协议的基本特征，掌握与协议行为类型有关的背景知识，知道特定协议在程序中的行为方式。

1. 面向消息的协议与基于流的协议

（1）面向消息的协议。面向消息的协议以消息为单位在网上传送数据，消息在发送端一条一条地发送，在接收端也只能一条一条地接收，每一条消息是独立的，消息之间存在着边界。例如，在图1.6中，甲工作站向乙工作站发送了3条消息，分别是128、64和32字节；乙作为接收端，尽管缓冲区是256个字节，足以接收甲的3条消息，而且这3条消息已经全部到达了乙的缓冲区，乙仍然必须发出3条读取命令，分别返回128、64和32个字节这3条消息，而不能用一次读取调用来返回这3个数据包。这称为“保护消息边界”（Preserving Message Boundaries）。保护消息边界是指传输协议把数据当作一条独立的消息在网上传输，接收端只能接收独立的消息。也就是说，存在保护消息边界，接收端一次只能接收发送端发出的一个数据包。UDP就是面向消息的。面向消息的协议适于交换结构化数据，网络游戏就是一个好例子。玩家们交换的是一个个带有地图信息的数据包。

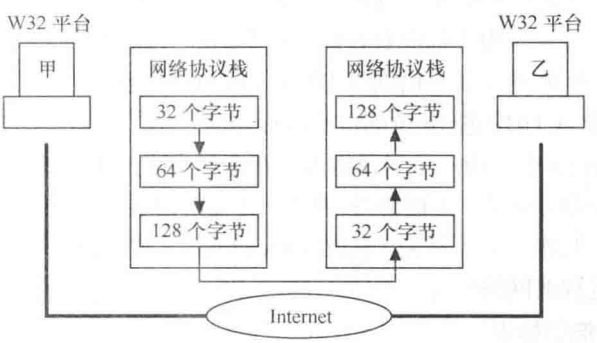


图1.6 保护消息边界的数据报传输服务

（2）基于流的协议。基于流的协议不保护消息边界，将数据当作字节流连续地传输，不管实际消息边界是否存在。如果发送端连续发送数据，接收端有可能在一次接收动作中接收两个或者更多的数据包。在发送端，允许系统将原始消息分解成几条小消息分别发送，或把几条消息积累在一起，形成一个较大的数据包，一次送出。多次发送的数据统一编号，从而把它们联系在一起。接收端会尽量地读取有效数据。只要数据一到达，网络堆栈就开始读取它，并将它缓存下来等候进程处理。在进程读取数据时，系统尽量返回更多的数据。在图1.7中，甲发送了3个数据包：分别是128、64和32个字节，甲的网络堆栈可以把这些数据聚合在一起，分两次发送出去。是否将各个独立的数据包累积在一起，受许多因素的影响，如网络允许的最大传输单元和发送的算法。在接收端，乙的网络堆栈把所有进来的数据包聚集在一起，放入堆栈的缓冲区，等待应用进程读取。进程发出读的命令，并指定了进程的接收缓冲区，如果进程的缓冲区有256个字节，系统马上就会返回全部224（128+64+32）个字节。如果接收端只要求读取20个字节，系统就会只返回20个字节。TCP是基于流的协议。

流传输，把数据当作一串数据流，不认为数据是一个一个的消息。但是有很多人在使用TCP通信时，并不清楚TCP是基于流的传输，当连续发送数据的时候，他们认为TCP会丢包。其实