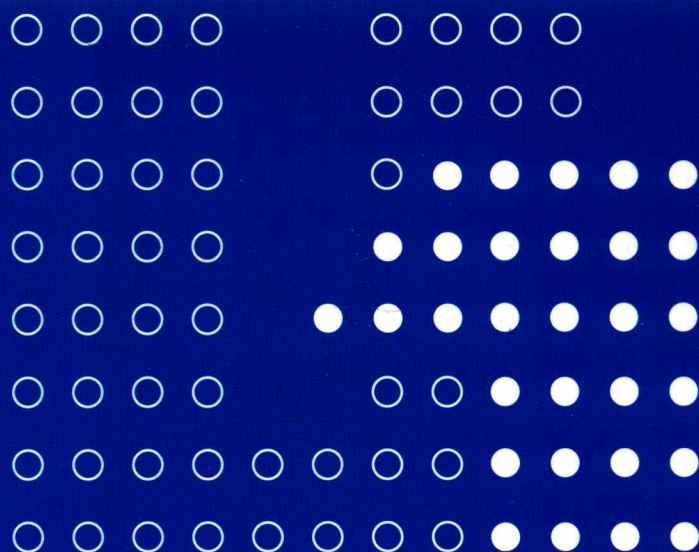
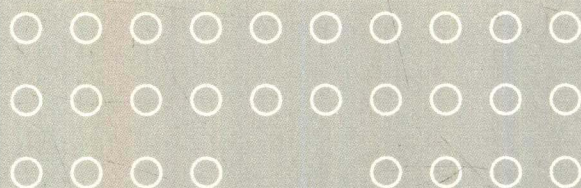


计算机系列教材

TCP/IP

网络编程原理与技术



钟辉 臧晗 董洁 宋凯 孟祥宇 高野 编著

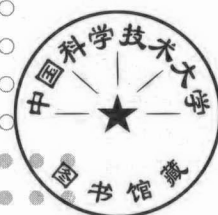
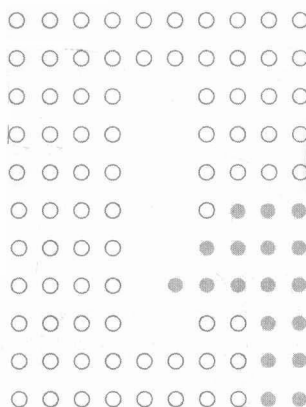
清华大学出版社



计算机系列教材

钟辉 臧晗 董洁 宋凯 孟祥宇 高野 编著

TCP/IP 网络编程原理与技术



清华大学出版社
北京

内 容 简 介

Internet 是世界上最大的计算机互连网络, TCP/IP 是 Internet 上使用最为成熟的协议。本书重点介绍用 TCP/IP 进行编程的主要原理和编程环境, 并举出实例来解释这些编程原理和概念。

网络中最基本的通信基础是客户-服务器模式, 它在计算机通信中占主导地位。本书内容主要包括客户-服务器各部件的功能, 还说明了如何构建客户和服务软件。介绍了客户-服务器模式的基本概念, TCP/IP 协议提供传输数据的基本机制; 如何在 TCP/IP 环境下组织编写应用程序; 计算机网络通信程序的构建方法; 从而进一步了解在网络环境下构建分布式程序。

全书共分 13 章: 第 1 章着重介绍网络编程的目标和准备工作; 第 2 章和第 3 章着重介绍客户-服务器的概念以及并发处理存在的主要问题和应用; 第 4 章和第 5 章基于 Linux 操作系统介绍套接字接口的概念和封装的系统调用函数; 第 6 章介绍客户程序设计和需要注意的细节问题; 第 7 章介绍了各种典型服务器的设计方法, 需要注意的问题和细节; 第 8 章和第 9 章介绍单线程编写并发程序的方法和条件; 第 10~12 章介绍多服务、多协议服务器设计方法和并发管理过程; 第 13 章介绍客户并发设计的方法和使用条件。全书提供了大量应用实例, 每章后均附有习题。

本书适合作为高等院校计算机、软件工程、信息管理等专业高年级本科生、研究生的教材, 也可供对 TCP/IP 比较熟悉并且对网络编程有所了解的开发人员、广大科技工作者和研究人员参考。

本书封面贴有清华大学出版社防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

TCP/IP 网络编程原理与技术/钟辉等编著. —北京: 清华大学出版社, 2019

(计算机系列教材)

ISBN 978-7-302-52602-5

I. ①T… II. ①钟… III. ①计算机网络—通信协议—教材 ②计算机网络—程序设计—教材
IV. ①TN915.04 ②TP393.09

中国版本图书馆 CIP 数据核字(2019)第 044592 号

责任编辑: 白立军

封面设计: 常雪影

责任校对: 梁毅

责任印制: 丛怀宇

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦 A 座 邮 编: 100084

社 总 机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质量反馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

课件下载: <http://www.tup.com.cn>, 010-62795954

印 装 者: 北京国马印刷厂

经 销: 全国新华书店

开 本: 185mm×260mm

印 张: 13

字 数: 306 千字

版 次: 2019 年 8 月第 1 版

印 次: 2019 年 8 月第 1 次印刷

定 价: 39.00 元

产品编号: 082182-01

前 言

随着 Internet 的发展,网络技术已经渗透到人们的生活和工作中。TCP/IP 已经成为最流行的网络协议,且还在演变以满足未来的需要。在速度越来越快的计算机硬件和不断更新的软件发展的背后,TCP/IP 在任何类型的硬件上都很容易实现和编写应用程序。网络作为中枢神经把世界连在一起。也正是因为网络的出现与发展,编写网络应用程序的程序员和工程师也在不断增加。TCP/IP 网络编程看起来非常简单,应用程序接口(application program interface,API)十分易懂。即使初学者也可以使用现代高级语言抽象的客户-服务器程序模板来编写应用程序。但是发现初学者在经历了最初的高效编程之后,在软件设计细节面前开始停滞不前,同时发现他们编写的程序正在遭受性能和健壮性的考验,在灵活使用客户-服务器模式解决应用问题过程中,缺乏对程序运行可靠性、并发控制、容错等方面的解决方法,造成程序运行不稳定甚至错误。网络编程完全不同于常规的单机编程,每个程序运行都要受网络上其他在线程序的控制和干扰。网络编程是一个充满黑暗角落的领域,一些细节有可能会被错误理解。如果停留在高层语言抽象环境里编程,永远不能掌握客户-服务器的实质与精髓。本书对 TCP/IP 网络编程最精细的基本理论和概念进行了分析和介绍,能照亮读者编程中黑暗的角落并帮助其改正错误。

通过本书的介绍,读者会透彻理解网络编程的许多难点。本书详细介绍了客户-服务器编程的所有细节。通过对这些细节的理解,读者将获得相应的知识,即网络协议的内部工作机制如何与应用程序交互。因而对那些以前看起来令人困惑的程序行为就会变得很容易理解,解决问题的办法就会变得很清晰。本书面向套接字的网络编程学习者,只要掌握最基础的 C 语言编程实例和概念,就能掌握最基本的客户-服务器编程细节,更能方便在学习了 C++ 之后,掌握面向对象高级封装环境的网络编程过程。掌握相关的网络协议知识以及操作系统的基础知识将有助于客户-服务器编程。本书针对网络编程的初学者,首先强调基本概念和原理的掌握,加强所有章节示例的可读性,然后才考虑代码的优化问题。本书适合所有希望学习 Linux 的网络编程的读者。Linux 操作系统是早期的网络编程环境,对于该环境的掌握就不难了解 Windows 环境的网络编程,因为 Windows 操作系统的网络编程环境也是来自于 UNIX 操作系统的移植,所以使用任何操作系统都不会有问题。网络编程的特点决定了同时学习两种操作系统平台的网络编程是最有效的学习方法。不必为学习本书的内容而特意掌握 Linux 和 Windows 两种操作系统的操作方法,只需要了解各自的编译方法即可。本书主要介绍了 Linux 平台下的套接字函数和调用方法,所有示例程序都是在 Linux 平台下实现的。

作为一名长期从事计算机网络相关课程教学的教师,作者一直在考虑这样一个问题:怎样用有限的课堂教学时间为学生系统地讲授网络编程的基本概念、基本原理和应用?教材是教学过程中使用的主要资料,是教与学的主要内容依据,所以一本好的教材应科学、合理地覆盖本门课程的知识,具有严谨的总体结构和章节安排,内容应详略得当且能

够突出重点。同时,编写教材的人员也应该注意本门课程与操作系统、C 语言课程之间的联系,解决好课程之间的衔接。作者认为编写网络编程教材应有如下的要求:第一,应具备丰富的实践经验,对自己的专业有深刻的理解;第二,应具备丰富的教学经验,能够把握学生的学习规律,并力求将深奥的理论叙述和讲解简单化;第三,应具备对知识的归纳和总结能力,并具有良好的写作功底,能够将知识阐述得准确、清晰。一本好的教材必须经过精心规划和设计。本书在出版前其内容已经在多年的教学过程中讲授过,作者对内容进行了多次调整和增减,增加了大量易于操作和实验内容。

本书的内容特色是在介绍每一类服务器算法时都增加了配套的示例和讲解过程,并配有源代码和运行结果。让读者可以通过代码运行理解客户-服务器算法的实现过程,了解客户-服务器的结构。尤其是通信代码的交互过程,它是网络编程的难点所在,特意强调在各种应用过程中使用套接字通信交互的理解和使用。其次强调了并发服务器的编写,它也是服务器编写的难点之一。

本书共分为 13 章:第 1~5 章主要介绍网络编程的主要概念——客户-服务器、并发程序、套接字接口及其 API;第 6 章介绍客户编程的基本概念和存在的主要问题,举出实例讲解客户编程的一般方法,隐藏细节的过程库,重点介绍了 TCP 和 UDP 的客户编程;第 7 章介绍服务器编程的基本概念和需要解决的问题,举例说明四大类服务器编写的代码示例,重点说明并发服务器的设计方法;第 8 章和第 9 章介绍单线程实现并发程序的设计及其应用场景;第 10~12 章介绍多服务、多协议服务器设计方法和原理,以及并发性管理;第 13 章介绍客户并发设计的原理和应用场景。

在本书的编写过程中,得到教学团队同事的大力支持和帮助,借此机会向他们表示衷心的感谢!本书由钟辉统筹全书章节内容和框架,并校对所有章节内容。第 1~5 章由董洁、臧晗编写,并进行了代码实验操作验证,总结实验步骤和遇到的问题;第 6 章和第 7 章由钟辉编写,孟祥宇、高野编写实例代码进行实验验证,总结实验代码所遇到的问题;第 8 章和第 9 章由宋凯、孟祥宇编写,高野进行了代码实验验证;第 10~12 章由钟辉、董洁编写,臧晗编写实例代码进行实验验证,并总结出现的问题。

计算机软硬件与互联技术发展迅速,限于作者的学识和时间,本书难免有错误与不妥之处,恳请读者批评指正,作者将万分感谢。

编 者

2018 年 8 月于沈阳

目 录

第 1 章 网络编程准备	1
1.1 TCP/IP 技术的因特网应用	1
1.2 用 TCP/IP 构建分布式环境设计应用程序	2
1.3 用 TCP/IP 构建的标准和非标准应用协议	2
1.4 使用 TCP/IP 标准应用协议的例子	2
1.5 Telnet 连接的例子	3
1.6 使用 Telnet 访问其他服务	4
1.7 TCP/IP 应用协议和软件灵活性	5
1.8 从提供者的角度看服务	5
1.9 本教材内容介绍	6
1.10 小结.....	6
习题.....	6
 第 2 章 客户-服务器模式软件设计概念	8
2.1 客户-服务器的起源	8
2.2 客户-服务器关键问题	9
2.3 客户-服务器术语	9
2.3.1 客户和服务器	10
2.3.2 服务器特权和复杂性	10
2.3.3 标准和非标准客户软件	10
2.3.4 客户的参数化	11
2.3.5 无连接的和面向连接的服务器	11
2.3.6 无状态和有状态服务器	12
2.3.7 无状态文件服务器的例子	13
2.3.8 有状态文件服务器的例子	13
2.3.9 客户标识	14
2.3.10 无状态是一个协议问题.....	15
2.3.11 充当客户的服务器.....	16
2.4 小结.....	16
习题	17

第 3 章 客户-服务器模式软件中的并发处理	18
3.1 引言	18
3.2 网络中的并发	18
3.3 服务器中的并发	19
3.4 并发术语	20
3.4.1 进程概念	20
3.4.2 局部和全局变量的共享	21
3.4.3 过程调用	22
3.5 一个创建并发进程的例子	22
3.5.1 一个顺序执行的 C 实例	22
3.5.2 程序的并发版本	23
3.5.3 时间分片	24
3.5.4 单线程的进程	25
3.5.5 使各进程分离	25
3.6 执行新的代码	26
3.7 上下文切换和协议软件设计	27
3.8 并发和异步 I/O	27
3.9 小结	28
习题	28
第 4 章 网络编程协议的程序接口	29
4.1 引言	29
4.2 不精确指明的协议软件编程接口	29
4.3 接口功能	30
4.4 概念性接口的规范	30
4.5 操作系统调用	31
4.6 网络通信的两种基本方法	31
4.7 Linux 中提供的基本 I/O 功能	32
4.8 将 Linux I/O 用于 TCP/IP	33
4.9 小结	33
习题	33
第 5 章 接口实现——套接字 API	35
5.1 引言	35
5.2 Berkeley 套接字	35
5.3 指明一个协议接口	35
5.4 套接字的抽象	36
5.4.1 套接字描述符和文件描述符	36

5.4.2	针对套接字的系统数据结构	37
5.4.3	主动套接字或被动套接字	37
5.5	指明端点地址	38
5.6	类属地址结构	39
5.7	套接字 API 中的主要系统调用	40
5.7.1	socket 调用	40
5.7.2	connect 调用	40
5.7.3	send 调用	41
5.7.4	recv 调用	41
5.7.5	close 调用	41
5.7.6	bind 调用	41
5.7.7	listen 调用	42
5.7.8	accept 调用	42
5.7.9	在套接字中使用 read 和 write	42
5.7.10	套接字调用小结	42
5.8	用于整数转换的实用例程	43
5.9	在程序中使用套接字调用	44
5.10	套接字调用的参数所使用的符号常量	44
5.11	小结	45
	习题	46
第 6 章	客户软件算法及编程实例	47
6.1	引言	47
6.2	不是研究细节而是学习算法	47
6.3	客户体系结构和要解决的问题	48
6.3.1	标识服务器的位置	48
6.3.2	分析地址参数	49
6.3.3	查找域名	50
6.3.4	由名字查找某个熟知端口	51
6.3.5	端口号和网络字节顺序	51
6.3.6	由名字查找协议	51
6.4	TCP 客户算法	52
6.4.1	分配套接字	52
6.4.2	选择本地协议端口号	53
6.4.3	选择本地 IP 地址中的一个基本问题	53
6.4.4	将 TCP 套接字连接到某个服务器	54
6.4.5	使用 TCP 与服务器通信	54
6.4.6	从 TCP 连接中读取响应	55

6.4.7 关闭 TCP 连接	55
6.5 UDP 客户算法	56
6.5.1 连接的和非连接的 UDP 套接字	57
6.5.2 对 UDP 使用 connect	57
6.5.3 使用 UDP 与服务器通信	57
6.5.4 关闭使用 UDP 的套接字	57
6.5.5 对 UDP 的部分关闭	58
6.5.6 关于 UDP 不可靠性的警告	58
6.6 客户编程实例的重要性	58
6.7 隐藏细节	59
6.8 针对客户程序的过程库	59
6.8.1 connectTCP 的实现	60
6.8.2 connectUDP 的实现	60
6.8.3 构成连接的过程	61
6.9 过程库使用	63
6.10 DAYTIME 服务	63
6.10.1 针对 DAYTIME 的 TCP 客户实现	64
6.10.2 从 TCP 连接中进行读	65
6.11 TIME 服务	66
6.12 访问 TIME 服务	66
6.13 精确时间和网络时延	67
6.14 针对 TIME 服务的 UDP 客户	67
6.15 ECHO 服务	69
6.16 针对 ECHO 服务的 TCP 客户	69
6.17 针对 ECHO 服务的 UDP 客户	71
6.18 小结	72
习题	74
第 7 章 服务器软件算法及编程实例	76
7.1 引言	76
7.2 概念性的服务器算法	76
7.3 并发服务器和循环服务器	76
7.4 面向连接的和无连接的访问	77
7.5 服务器需要考虑的几个问题	77
7.5.1 传输层协议的语义	77
7.5.2 选择传输协议	78
7.5.3 面向连接的服务器	78
7.5.4 无连接的服务器	79

7.5.5	服务器的故障、可靠性和无状态	80
7.5.6	优化无状态服务器	80
7.5.7	请求处理时间	82
7.6	服务器的四种基本类型	82
7.7	循环服务器的算法	83
7.8	循环的、面向连接的服务器的算法	83
7.8.1	用 INADDR_ANY 绑定熟知端口	83
7.8.2	将套接字置于被动模式	84
7.8.3	接受连接并使用这些连接	84
7.9	循环的、无连接的服务器的算法	84
7.10	并发服务器的算法	85
7.11	并发的、无连接的服务器的算法	86
7.12	并发的、面向连接的服务器的算法	86
7.12.1	服务器并发性的实现	87
7.12.2	把单独的程序作为从进程来使用	87
7.13	使用单线程获得表面上的并发性	88
7.14	各服务器类型所适用的场合	89
7.15	服务器类型小结	89
7.16	重要问题——服务器死锁	90
7.17	其他的实现方法	90
7.18	循环的、无连接的服务器设计	91
7.18.1	创建被动套接字	91
7.18.2	进程结构	94
7.18.3	TIME 服务器举例	94
7.18.4	小结	96
7.19	循环的、面向连接的服务器设计	96
7.19.1	分配被动的 TCP 套接字	96
7.19.2	用于 DAYTIME 服务的服务器	97
7.19.3	进程结构	97
7.19.4	DAYTIME 服务器举例	98
7.19.5	关闭连接	100
7.19.6	连接终止和服务器的脆弱性	100
7.19.7	小结	101
7.20	并发的、面向连接的服务器设计	101
7.20.1	并发 ECHO	102
7.20.2	循环与并发实现的比较	102
7.20.3	进程结构	102
7.20.4	并发 ECHO 服务器举例	103

7.20.5 清除游离(errant)进程	106
7.20.6 小结	106
7.21 小结	107
习题	107
第 8 章 使用线程模型实现并发	110
8.1 引言	110
8.2 Linux 线程概述	110
8.3 线程和进程的关系	111
8.3.1 描述符、延迟和退出	111
8.3.2 线程退出	112
8.4 线程协调和同步	112
8.4.1 互斥	112
8.4.2 信号量	113
8.4.3 条件变量	113
8.5 使用线程的服务器实例	114
8.6 监控	117
8.7 小结	118
习题	118
第 9 章 单线程并发服务器设计	119
9.1 引言	119
9.2 服务器中的数据驱动处理	119
9.3 用单线程进行数据驱动处理	120
9.4 单线程服务器的线程结构	120
9.5 单线程 ECHO 服务器举例	121
9.6 小结	123
习题	124
第 10 章 多协议服务器设计	125
10.1 引言	125
10.2 减少服务器数量的动机	125
10.3 多协议服务器的设计	125
10.4 进程结构	126
10.5 多协议 DAYTIME 服务器的例子	126
10.6 共享代码的概念	129
10.7 并发多协议服务器	130
10.8 小结	130

习题	130
第 11 章 多服务服务器设计	132
11.1 引言	132
11.2 合并服务器	132
11.3 循环的、无连接的、多服务服务器设计	132
11.4 循环的、面向连接的、多服务服务器设计	133
11.5 并发的、面向连接的、多服务服务器设计	134
11.6 并发的、单线程的、多服务服务器的实现	135
11.7 从多服务服务器调用单独的程序	135
11.8 多服务、多协议服务器设计	136
11.9 多服务服务器的例子	137
11.10 静态的和动态的服务器配置	143
11.11 UNIX 超级服务器——inetd	144
11.12 inetd 服务器的例子	146
11.13 小结	147
习题	148
第 12 章 服务器并发性管理	149
12.1 引言	149
12.2 在循环设计和并发设计间选择	149
12.3 并发等级	150
12.4 需求驱动的并发	150
12.5 并发的代价	150
12.6 额外开销和时延	151
12.7 小时延问题	151
12.8 从线程/进程的预分配	152
12.8.1 Linux 中的预分配	153
12.8.2 面向连接的服务器中的预分配	153
12.8.3 互斥、文件锁定和 accept 并发调用	154
12.8.4 无连接的服务器中的预分配	155
12.8.5 预分配、突发通信量和 NFS	155
12.8.6 多处理器上的预分配	156
12.9 延迟的从线程/进程分配	156
12.10 两种技术统一的基础	157
12.11 技术的结合	157
12.12 小结	158
习题	158

第 13 章 客户软件并发设计	160
13.1 引言	160
13.2 并发的优点	160
13.3 运用控制的动机	160
13.4 与多个服务器的并发联系	161
13.5 实现并发客户	162
13.6 单线程实现	162
13.7 使用 ECHO 的并发客户的例子	163
13.8 并发客户的执行	167
13.9 例子代码中的并发性	169
13.10 小结	169
习题	169
附录 A 系统调用与套接字使用的库例程	170
参考文献	192

第 1 章 网络编程准备

网络编程是一个知识面很宽广的领域,采用网络技术在两台或更多台计算机之间进行数据通信时,存在很多障碍,这些障碍遍及简单的串行连接到复杂的网络体系结构。如今人们都清楚地认识到 TCP/IP 是构建网络的首选技术。这很大程度上是因为互联网的发展和其最盛行的 Web 互联网应用程序:万维网(World Wide Web)。Web 使用了浏览器和服务器应用程序,以及 HTTP,但实际上它并不是一个程序,也不是一个协议。Web 是互联网上最广泛的可视化网络技术的应用程序。

其实在 Web 诞生之前,TCP/IP 也是一种非常流行的网络构建方法。因为 TCP/IP 是一个开放的标准并且可以在不同的硬件之间进行互相连接,因此越来越多的人使用 TCP/IP 来构建网络和网络应用程序。到 20 世纪 90 年代初,TCP/IP 已经成为网络的主要技术和事实标准,而且这种技术趋势将在未来的很长一段时间内持续下去。如果希望了解和掌握网络编程技术,就必须先掌握一些背景知识,其对全面理解网络编程技术有很大帮助。本书首先介绍初学者通常会遇到的几个问题,引导大家掌握网络编程技术的精髓和方法。这些问题很大一部分是一些对概念的认识不清楚和误解造成的,其中包括对 TCP/IP 的概念及其应用程序接口(application program interface,API)的不完全理解。

1.1 TCP/IP 技术的因特网应用

TCP/IP 已经成为业界最成熟的互联网协议,且使与厂商无关的计算机通信成为可能。而客户-服务器模式在计算机通信中占主导地位,它是网络中最基本的通信基础。理解客户-服务器模式各部件的功能,掌握 TCP/IP 提供传输数据的基本机制,才能掌握如何构建客户和服务器软件,如何在 TCP/IP 环境下编写网络应用程序,掌握计算机网络通信程序的基本构造。

互联网技术已经十分普及,目前全世界范围内的大多数公用和专用网络都选用了 TCP/IP。在欧洲、印度洋、南美洲和太平洋周边的一些国家,TCP/IP 的使用正迅速增长。TCP/IP 提供了将计算机互联起来的技术,通过 TCP/IP 互联网以获得多种可用的应用。有些应用很明显是使用 TCP/IP 的,而有些应用却不太明显。我们最熟悉的是与整个因特网和万维网有关的应用,如浏览网页、聊天室;还有通常被称为万维网广播(webcasting)的流式处理(streaming)。企业或公司以其他的方式来使用 TCP/IP 技术,例如,一家公司可使用 TCP/IP 监视和控制海上石油平台,而另一家公司可使用 TCP/IP 控制库存数量。一些连锁经营的宾馆在它们的预订系统中使用了 TCP/IP,每次预订信息通过一个专用的 TCP/IP 互联网进行通信。另外,许多大规模的网络使用 TCP/IP 应用来监视和控制网络设备。除此之外,新的应用也正不断涌现。

1.2 用 TCP/IP 构建分布式环境设计应用程序

网络技术已日渐成为所有软件的一部分,程序员必须掌握这样的基础知识:设计和实现分布式应用程序所用到的原则和技术。我们将会看到,分布式计算的一个主要目标就是透明性,即所产生的分布式程序的行为应尽可能与同样程序的非分布式版本一样。因此,分布式计算的目标就是提供一个环境,该环境隐藏了计算机和服务的地理位置,使它们看上去就像是在本地一样。例如,传统的数据库系统将数据同应用程序存储在上一台计算机上,而分布式数据库系统允许用户访问存储在不同地点的不同计算机上的数据。若应用程序设计得好,用户是感觉不出来数据是本地的还是远程的。

1.3 用 TCP/IP 构建的标准和非标准应用协议

TCP/IP 协议族包含许多应用协议,而且新的应用协议每日都在出现。事实上,只要一个程序员设计了两个使用 TCP/IP 通信的程序,这个程序员就已经发明了一种新的应用协议。当然,有些应用协议已经被记录和标准化,并被采纳为正式的 TCP/IP 协议族的一部分,我们称这种协议为标准应用协议。应用程序员发明的其他供私人使用的协议,则称为非标准应用协议。

只要有可能,多数网络管理员都会选择使用标准的应用协议。当现有的协议足够用时,就不必再去发明一种新的应用协议。例如,TCP/IP 包含了像文件传送、远程登录和电子邮件(E-mail)等服务的标准应用协议。因此,对这些服务,程序员应使用标准协议。一般来讲,标准应用协议用于那些公共的应用环境,如 E-mail、WWW、Telnet 和 FTP 等,它们类似于操作系统下运行的一些通用应用程序命令,供所有用户使用。非标准应用协议是具有特殊用途的部分用户使用的,如酒店管理系统、物流管理系统、企业 ERP 系统等。

1.4 使用 TCP/IP 标准应用协议的例子

尽管一个给定的远程登录会话仅仅以人可以输入的速度产生数据,并以人可以阅读的速率接收数据。但在已连接的因特网中,远程登录的通信量名列前十名。许多用户依赖远程登录作为工作环境的一部分;不必直接连接到用于完成大部分计算的那些计算机上。

TCP/IP 协议族包含一个用于远程登录的标准应用协议,称为 Telnet。Telnet 协议定义了应用程序为登录到远程计算机而必须发送给该计算机的数据的格式,以及远程计算机发回的报文的格式。它定义了字符数据在传输时应如何被编码,以及为控制会话或放弃一个远程登录应发送的特殊报文。

Telnet 协议如何对数据进行编码的内部细节,对大多数用户是不相关的。一个用户可以调用接入远程计算机的软件,而无须知道或关心它的实现。实际上,使用远程服务常

常和使用本地服务一样容易。例如,运行 TCP/IP 的计算机系统往往含有一个命令,用户调用此命令来运行 Telnet 软件。在很多系统中,该命令名为 telnet。为了调用它,用户输入:

```
telnet machine
```

参数 machine 表示期望远程登录接入的计算机的域名。因此,为构成一个到计算机 example.com 的 Telnet 连接,用户输入:

```
telnet example.com
```

从用户的观点看,运行 telnet 将连接远程计算机并使用户窗口中的命令可在远程计算机上执行。一旦建立了连接,telnet 应用就提供一个双向通信信道。只要用户一直打开窗口,telnet 应用程序就会把用户输入的每个字符都发送给远程计算机,并把远程计算机发出的每个字符显示在用户的显示器上。

通常在 telnet 用户连接到一个远程系统后,该远程系统会要求用户输入登录标识符和口令进行验证。这个提供给远程用户的提示信息与提供给本地用户的提示符信息是一样的。因此,Telnet 提供给每个远程用户一种正在使用直连终端的错觉。

1.5 Telnet 连接的例子

作为一个例子,考虑一下当某位用户调用 telnet 并连接到计算机 purdue.edu 时,会发生什么:

```
telnet purdue.edu
Trying...
Connected to purdue.edu
Escape character is '^]'.
SunOS 5.6
Login:
```

当 telnet 程序将计算机名转换为 IP 地址并试图与该地址建立一个有效的 TCP 连接时,就出现了最开始的输出消息“Trying...”。一旦建立了连接,telnet 便打印出第 2 行和第 3 行,它们通知用户连接尝试已经成功并指出一个特殊字符,在需要时,用户可以输入该字符临时从 telnet 应用程序中退出(例如,如果发生了故障,用户就要放弃连接)。符号^]表示用户必须按住 Ctrl+]键。

输出的最后几行来自远程计算机。它们表明其操作系统是 SunOS 5.6 版,还提供了一个标准的登录提示符。光标停在“Login:”消息后,等待用户输入一个合法的登录标识符。用户必须在远程计算机上有一个账户,Telnet 会话才能继续下去。在用户输入一个合法的登录标识符后,远程计算机提示用户输入口令。只有在登录标识符和口令都有效的情况下,才允许用户接入。

1.6 使用 Telnet 访问其他服务

TCP/IP 使用协议端口号来标识一台特定计算机上的应用服务。实现某个特定服务的软件在某个预先确定的(熟知)协议端口上等待请求。例如,用 Telnet 协议访问的远程登录服务,它被分配的端口目的是 23。默认情况下,当用户调用 telnet 程序时,程序会连接指定计算机上的端口 23。

有趣的是,Telnet 协议可以用来访问不同于标准的远程登录服务的其他服务。为此,用户必须指定所期望访问服务的协议端口号。当在命令行上调用 telnet 时,大多数系统会提供可选的第二个参数,以便允许用户指定其他的协议端口号;一些 telnet 版本提供一个图形接口让用户从一个菜单项中选择端口号。不管哪种情况,若用户没有提供第二个参数,telnet 就使用端口 23。但是,若用户指明了一个协议端口号,telnet 就连接该端口号。例如,如果用户输入:

```
telnet purdue.edu 13
```

telnet 程序将建立与计算机 purdue.edu 上的协议端口 13 的一个连接。

端口 13 对应的服务并不是常规的远程登录服务,而是将提供 DAYTIME 服务,报告远程计算机上的本地日期和时间。在 telnet 首次连接端口 13 时,远程计算机不会返回任何提示。但是,telnet 程序会产生 3 行输出:

```
telnet purdue.edu 13
Trying...
Connected to purdue.edu
Escape character is '^]'.
```

而远程计算机上的服务只产生了 1 行含日期和时间的输出。在它产生输出后,远程计算机将关闭连接。

```
Sun Jan 16 21:52:08 2000
Connection closed by foreign host.
```

Finger 服务可从端口 79 上获得,它是一种使用 Telnet 协议访问的交互式服务。Finger 服务提供有关已登录到计算机中的所有用户信息或某个指定用户的信息。输入可以是一个空行(提示要列出计算机上的所有用户)加回车符,或一个用户名(提示只要提供指定用户的信息)加一个回车符。

虽然 Finger 服务是交互式的,但它不会自动产生提示信息,而是等待用户输入一个请求再加以响应。例如,为在计算机 purdue.edu 上的端口 79 处对 John Rice 执行 Finger 操作,可在调用 telnet 后,等待连接,然后输入用户名。服务会提供该用户的信息加以响应。下面所示的输出说明了这一交互过程:

```
telnet purdue.edu 79
Trying ...
```