

BBS、IM、微博，互联网到移动互联网的交流平台将更加深刻地改变世界。这些社会现象后面的技术本质是如何炼成的？本书为你抽丝剥茧，揭晓答案。

胡东锋 编著

微博



是这样炼成的

从聊天室到Twitter的技术实现

140字 推爆信息世界
Java 网络通信成就技术梦想

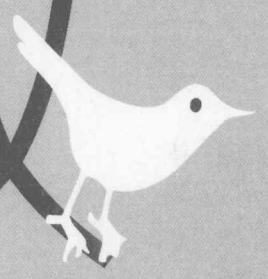


人民邮电出版社
POSTS & TELECOM PRESS

BBS、IM、微博，互联网到移动互联网的交流平台将更加深刻地改变世界。
这些社会现象后面的技术本质是如何炼成的？
本书为你抽丝剥茧，揭晓答案。

胡东锋 编著

微博是这样炼成的



从聊天室到Twitter的技术实现

140字推爆信息世界

Java网络通信成就技术梦想

人民邮电出版社
北京

图书在版编目 (C I P) 数据

微博是这样炼成的：从聊天室到Twitter的技术实现
/ 胡东锋编著. -- 北京：人民邮电出版社，2010.6
ISBN 978-7-115-22485-9

I. ①微… II. ①胡… III. ①移动通信—通信技术
IV. ①TN929.5

中国版本图书馆CIP数据核字(2010)第042690号

内 容 提 要

本书结合项目实践、构架设计、行业知识介绍和学习方法分析为一体，讲解了从原始的命令行聊天室到流行的 Twitter 系统的技术实现和构架分析。本书将带领读者分析完成具有行业代表性的仿 QQ IM 项目、中国移动 CMPP 网关项目、仿 Twitter 系统三大典型项目的构架设计和实现。在具体的项目实践过程中，融入了 TCP/IP 编程、设计模式、UDP 编程、P2P 通信、通信加密技术、Java NIO 技术、JMF 视频通信、RMI 远程调用、Hessian Web Service、Memcached 缓存系统等技术专题。目的是通过由浅入深、项目驱动的实践分析，让读者深入掌握 Java 网络编程和网络通信项目的构架设计，并引导启发读者能自行分析技术表象背后的原理。

本书面向有一定 Java 编程基础的中高级读者和初入行的一线软件开发者，也适合作为高等院校相关专业师生的参考书。

微博是这样炼成的：从聊天室到 Twitter 的技术实现

- ◆ 编 著 胡东锋
责任编辑 蒋 佳
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
中国铁道出版社印刷厂印刷
- ◆ 开本：800×1000 1/16
印张：25
字数：579 千字 2010 年 6 月第 1 版
印数：1～4 000 册 2010 年 6 月北京第 1 次印刷

ISBN 978-7-115-22485-9

定价：59.00 元

读者服务热线：(010)67132692 印装质量热线：(010)67129223
反盗版热线：(010)67171154

自序

希望以下介绍能帮助您理解本书要旨——买回一本不是自己想要的书的痛苦，我比你经历得多。

1. 本书缘起

本书是我在近十年软件开发和培训生涯基础上的处女作。谈不上十年磨一剑，至少也包含了我两年的思考。图书的缘起首先要感谢的是人民邮电出版社的蒋编辑，没有他的梳理规划和鼓励，本书的内容可能永远只是我头脑中的片断。促使我能坚持把这本书写完的动力，来源于我在蓝杰所带领的学生们的期盼和学习热情，感谢你们！当然，写作本书的时间是我的同事陈九龙、熊向军先生长期的加班所换取来的，一句谢谢无法全部代表我的感激。

2. 本书特点

(1) 项目驱动。

本书的技术点主要是网络通信、安全加密和项目设计构架 3 方面。讲解的技术要点有 TCP/IP 编程、设计模式、UDP 编程、P2P 通信、通信加密技术、Java NIO 技术、JMF 视频通信、RMI 远程调用、Hessian Web Service、Memcached 缓存系统等。这些看似难以理解和掌握的技术要点通过循序渐进的案例实践，被一步一步构建到一个完整的项目中。读者只要跟随本书一路走下来，至少能实践完成 3 个典型项目：仿 QQ IM 项目（JavaKe）、中国移动 CMPP 网关项目、仿 Twitter 项目（JTwitter）。同时，读者还可对相关行业业务知识进行详细深入的了解。有实践项目，才有深入理解，才叫做“掌握技术”。

(2) 启发式讲解。

做老师的经历让我明白“老师没有权力直接告诉学生答案”的道理。本书的目的不仅限于告诉读者“可以这样写代码”，而是通过提出需求、分析技术实现、分析缺陷改进、提示原理探究的渐进式讲解路线，充分调动读者实践和思考的积极性。在掌握技术要点、完成实践项目的同时，希望读者明白：可以这样实现、还可以那样实现，两种实现有什么优缺点，各自的原理是什么。

另外，本书中的代码不是仅为“示例”而编写的，从每行代码的注解命名到类与类之间的关系分析，都是合乎规范和经过严谨思考的。“代码就是程序员的小说”，希望读者能注重体会本书的代码质量规范和其中的设计思想。

3. 本书读者对象

如果读者已有几年的 Web 开发工作经验，却开始渐渐厌烦千篇一律的 CRUD 代码操作，编写“优雅、艺术”性代码的机会越来越少，编码就像是在各种现成框架中做填空题一样。本书将为你打开 Java 技术的另一个精彩世界。

如果读者是正在学习 Java 开发的征战者，对线程和 I/O 有些粗浅了解，却被各种名词堆砌的 Web（J2EE）技术搞得眼花缭乱，急切地想去掌握那些能以不变应万变、根本性、原理性的技术“干货”，那么本书就是为你量身打造的。

如果读者还是徘徊在“Hello Word”左右的菜鸟，请慎选。但如果你拥有坚持实践加思考的坚强毅力，能经受住本书的考验，那么你也肯定会变成一只“大鹏”。

书中的不足和错误在所难免，如果读者能不吝指教，我尤为感谢，我的邮箱是 JavaFound@Gmail.com。

胡东锋

2010 年 1 月 7 日 于岳麓山下

目 录

绪论	1
第 1 章 JavaKe 起步：聊天室的实现	5
1.1 从零开始实现公共聊天室	5
1.1.1 网络基础知识	5
1.1.2 一步一步创建简单服务器	8
1.1.3 服务器读写消息实现	11
1.1.4 群聊服务器实现	20
1.1.5 群聊客户端实现	37
1.2 实现 XMPP 通信的 IM 系统	46
1.2.1 初识 XMPP	46
1.2.2 交互流程描述	47
1.2.3 XMPP 消息格式定义	49
1.2.4 服务器端代码的实现	50
1.2.5 客户端代码的实现	57
1.2.6 缺陷分析	65
1.3 程序结构性问题分析	65
1.3.1 整体结构设计的重要性	65
1.3.2 方法定义时细节的处理	66
1.3.3 高耦合的问题	67
1.4 JavaKe 项目需求分析	68
1.4.1 JavaKe 需求分析	68
1.4.2 JavaKe 客户端的功能	68
1.4.3 JavaKe 服务器端的功能	69
1.4.4 JavaKe 系统网络结构	69
第 2 章 JavaKe：典型 IM 系统的实现	70
2.1 大话通信协议	70
2.1.1 理解通信协议的概念	70
2.1.2 定义文件传输协议并实现	71
2.1.3 实现文本/文件传送服务器	73
2.1.4 实现文本/文件传送客户端	76
2.2 JavaKe 通信协议分析	78

2.2.1	通信消息流程	78
2.2.2	通信数据格式协议	81
2.2.3	具体消息体结构定义	82
2.3	关键技术点分析	85
2.3.1	打包解包的概念分析	85
2.3.2	重构打包解包代码	86
2.3.3	应用监听器模型分离界面和通信层	91
2.3.4	“事件监听”模型应用的实现	95
2.3.5	UI 界面与数据模型分离	99
2.3.6	定制 UI 组件示例	107
2.4	JavaKe 系统对象分析	113
2.4.1	对象分析思路	113
2.4.2	用户/分组类定义	116
2.4.3	系统消息对象分析	118
2.4.4	工具类分析	119
2.5	JavaKe 服务器端的实现	129
2.5.1	服务器端分析	129
2.5.2	服务器创建模块的实现	130
2.5.3	服务器通信模块的实现	131
2.5.4	服务器管理模块的实现	135
2.5.5	再谈分析：编程与软件开发的区别	138
2.6	JavaKe 客户端的实现	139
2.6.1	客户端界面分析	139
2.6.2	客户端流程分析	141
2.6.3	客户通信模块类分析	142
2.6.4	客户 UI 界面模块类分析	146
2.7	JavaKe 待完成任务分析	157
2.7.1	功能性完善的问题提出	157
2.7.2	构架性完善的问题提出	158

第 3 章 JavaKe 扩展：连接移动 CMPP 网关 159

3.1	应用软件与移动通信网络的关系	159
-----	----------------	-----

3.2	移动增值业务解析	161
3.2.1	移动增值业务类型说明	161
3.2.2	何谓 SP 服务商	162
3.2.3	中国移动 MISC 平台介绍	162
3.3	CMPP 短信业务理解	164
3.3.1	短信的基本属性	164
3.3.2	移动网络中的短信流程	165
3.3.3	SP 短信服务的基本概念	166
3.3.4	SP 指令匹配理解	167
3.4	中国移动 CMPP 通信解析	167
3.4.1	运营商短信协议介绍	167
3.4.2	客户端与服务器模型	168
3.4.3	异步消息发送模式	168
3.4.4	长连接与短连接	169
3.4.5	CMPP 消息类型	169
3.4.6	CMPP 消息的结构	170
3.4.7	CMPP 部分术语解释	171
3.5	CMPP 短信网关的实现	172
3.5.1	CMPP_CONNECT 及其应答包结构分析	172
3.5.2	CMPP_CONNECT 和其应答包的定义	174
3.5.3	CMPP 应答包打包、解包工具类编写	175
3.5.4	打包过程的说明	179
3.5.5	解包的过程说明	179
3.5.6	字符串数据的读写理解	180
3.5.7	ISMG 服务器端实现	181
3.5.8	MD5 摘要计算原理及应用	184
3.5.9	SP 端网关实现	187
3.5.10	网关程序调试技巧	189
3.5.11	网关关键技术点分析	192
3.6	应用手机终端收发短信	195
3.6.1	AT 命令简介及应用	195
3.6.2	使用 smsLib 发送短信	198

第 4 章	通信高级技术分析	203
4.1	通信的安全保证	203
4.1.1	网络为什么不安全	203
4.1.2	用 Jpcap 窃取数据包	206
4.1.3	对称加密与消息摘要	216
4.1.4	非对称加密机制	222
4.1.5	SSL 安全通道通信示例	232
4.2	JavaKe 远程控制模块实现	241
4.2.1	远程控制原理	241
4.2.2	远程控制关键技术解析	243
4.2.3	被控制端实现	245
4.2.4	控制端实现	250
4.3	基于 UDP 通信的设计	254
4.3.1	UDP 通信示例	254
4.3.2	UDP 可靠传输的控制	258
4.3.3	UDP 组播消息	268
4.3.4	内网穿透实现分析	272
4.4	JavaNIO 通信示例	282
4.4.1	NIO 基本概念	282
4.4.2	NIO 简单聊天室	291
4.4.3	NIO 文件操作	297
4.5	Mina 通信框架应用	299
4.5.1	关于 Mina 框架	299
4.5.2	Mina 简单入门	299
4.5.3	Mina 的体系结构总结	305
4.5.4	使用 Mina 直接传送对象	307
4.5.5	扩展学习	308
4.6	应用 Memcached 实现缓存系统	309
4.6.1	初识 Memcached	309
4.6.2	缓存系统的网络构架	312
4.6.3	为 JavaKe 应用缓存系统	318
4.7	视频通信实现	321

4.7.1	JMF 概述	321
4.7.2	JMF 使用 RTP 协议	328
4.7.3	FMJ 视频聊天室的实现	334
4.8	远程调用技术: RMI vs Hessian	354
4.8.1	分布式计算的基本理解	354
4.8.2	分布式体系结构的模型	356
4.8.3	RMI 应用示例	357
4.8.4	定制 RMI 端口	364
4.8.5	简洁的 Hessian	366
第 5 章	JTwitter 系统实现分析	371
5.1	JTwitter 是什么?	371
5.1.1	概念分析	371
5.1.2	JTwitter 的核心功能需求	373
5.2	客户端程序功能分析	374
5.2.1	登录/注册功能	374
5.2.2	主界面功能	377
5.3	客户端和服务端通信接口设计	380
5.3.1	通信接口设计	380
5.3.2	通信机制设计	385
5.4	数据库结构分析	386
5.5	Web 服务端功能分析	387
5.6	扩展构架分析	388

绪 论

当我们访问某一个 Web 网站时，当我们拨打移动电话或交手机话费时，当我们 QQ 聊天时，当我们在 SNS 网站偷菜时，当我们用微博分享新闻时，当企业领导在 ERP 平台查看报表时……我们会发现自己的生活方式已经发生了变化，我们离不开计算机屏幕后面的网络通信技术。

从程序员的角度看：未来的软件很少能够离开网络通信技术的——就像一棵大树，决定花叶鲜亮的是深埋在地下的根——同样，没有网线，QQ 就不是 QQ 了。不懂，甚至夸张一点说，不精通网络通信的程序员，是无法开发出优秀的应用软件的！除非你想让自己的工作只是浮于表层！

也许你早有耳闻，TCP/IP 是网络通信的精要，但本书不会对 TCP/IP 进行巨细无遗的解释，这方面的书已经随处可见。

本书是想解决这样一个问题：将读者定位在软件开发者的角度，让读者去感受生活中的需求变化，从而在实践中完成网络通信软件的开发和掌握具体技术的应用。

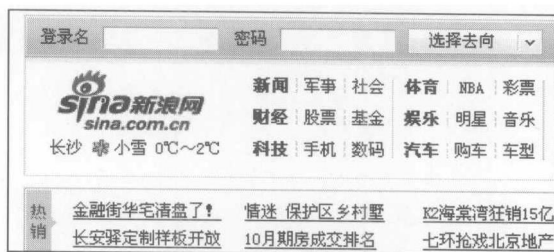
计算机通信只是一种技术，但人们对通信方式的要求却有不同点和侧重点。通过以下几种形式，我们可以看到其发展的路线。

1. 通信方式的发展路线分析

（1）Web 发布。

打开计算机，大多数人的第一个动作是通过浏览器打开一个 Web 页面，浏览器就与 Web 服务器进行了通信，接收 Web 服务器上的内容，显示为一个页面供你阅读。Web 网站的流行造就了国内三大门户网站的兴起。在互联网上，你可以看到别人为你编辑好的内容，通常你看到如图 1 所示的新闻网站就可以将其理解为广播式通信。

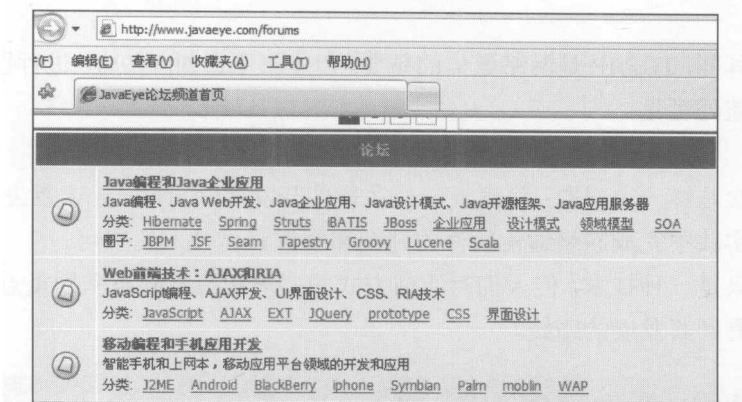
虽然内容丰富，虽然五彩斑斓，但是网站的编辑给你什么，你就只能看到什么，与其说是通信，不如说是灌输。不想只是处于被动的接受信息？于是，BBS 论坛出现了。



▲ 图 1 新浪新闻网站首页

(2) BBS 论坛。

任何人都可以在 BBS 中提出自己的一个话题，一群人围绕着这个话题，可以发表自己的看法。用户不但接收到了信息，还体验到了制造信息的快感。BBS 始终是以“话题”为中心的，发贴多的和发贴质量好的用户就有机会变成“版主”——掌握言论生杀大权。如果你发表不合论坛口味的内容，就会被“关小黑屋”；如果你的长篇大论确实很精彩，更多的也只是为论坛添彩而已。总之，在一个 BBS 论坛中，你很难体会到个体独立的内容的价值。图 2 所示是一个著名的技术论坛 javaEye (www.javaeye.com) 的首页。



▲ 图 2 javaEye 论坛首页

如果你想让自己的网名成为互联网世界响当当的名号，就必须要有自立门户，写自己的 Blog！

(3) Blog 写作。

Blog 这个概念刚兴起时，很多人感觉进入了一个自由新世界——我可以在网络世界里自由地表达自己——如果你想体会到自己的存在，就得拼命地表达（表现）自己。在网络上，你

只需要拥有自己的 Blog 网站，就可以发表自己的思想、生活动态、专业见解。Blog 给每个人在互联网上开辟了独立的空间，或者说独立展示的空间，如图 3 所示。

(4) Twitter 传播。

观念是可以改变的，当很多人愿意时刻将自己所见所思所闻以更快捷的手段公开时——Twitter 出现了。国内最优秀的平台新浪微博，如图 4 所示。



▲ 图 3 javaEye 的个人 Blog 首页



▲ 图 4 新浪微博首页

当 Twitter 流行，网络将会成为真正的信息高速公路。通过 Web 网页看到的 Twitter，可能只是一个简单的页面，你可以登录，并发布自己的消息——这只是 Twitter 的一面而已。一个 Twitter 系统可能集成了所有可以通信的手段：通过 Web 方式，通过桌面 IM 系统，通过移动终端交流……

本书将按照一个什么样的线索，分析并示例 Twitter 系统的实现呢？接下来将揭晓，当然，本书侧重点在分析技术手段，具体地说是 Java 网络通信编程技术的实现。

2. 本书技术实现路线

(1) 命令行聊天室。

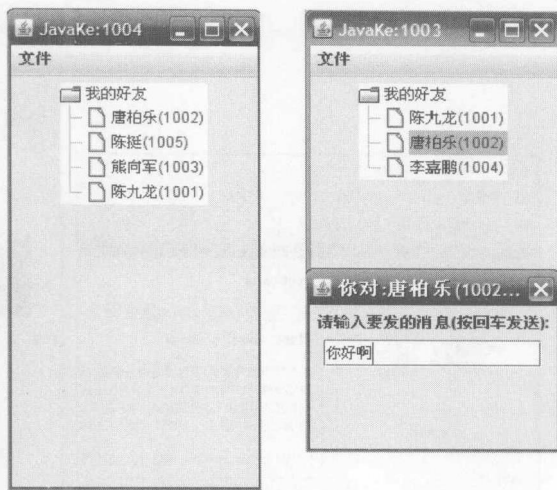
首先从我们最为熟悉的桌面聊天室开始，假设你是一个入门的程序员，一开始不可能开发像 QQ 那样的软件，但一个最为简单和原始的命令行聊天室，只有几百行代码，肯定是可以实现的，如图 5 所示。

(2) 基本完备的仿 QQ 通信系统。

当然，你肯定不满足于此，一定要开发一个与 QQ 媲美的聊天工具！看上去这是一件几乎不可能完成的任务。本书要做到的是帮助你解决开发过程中所需要的技术手段，至于要花费多少时间，只能由你确定。至少你会知道，QQ 这样一个系统是如何开发出来的！本部分将通过自定义通信协议，实现了一个简版的仿 QQ 通信系统，如图 6 所示。



▲ 图 5 原始命令行聊天室



▲ 图 6 仿照 QQ 桌面聊天工具

(3) 与移动通信连接。

一个完备的系统，不可能离开移动通信网络的连接和支持。在分析完仿 QQ 的聊天系统实现后，本书分析了中国移动 CMPP 通信网关连接的典型技术实现和通过 AT 命令驱动手机终端通信的实现方式。本部分的技术点将为你的系统插上无线的翅膀！

(4) 高级通信技术分析。

这毕竟是一本技术书籍，如何实现更加优化、高性能的通信系统是最重点的。分布式系统、高效的缓存系统、视频通信、通信数据加密……这些专题技术将专门放在一章中讲解。在这些技术点的支撑下，你开发的系统才会更加完备！

(5) JTwitter 系统实现分析。

最终，分析的这么多技术手段，无论是 Web 的，还是桌面的，还是移动的，都是希望将其组合到一起，形成一个 Twitter 通信系统。这是本书最后一章的任务。

Twitter 到底是什么？你可以直接翻到本书最后一章去体会，但建议你首先登录 <http://t.sina.com.cn/>，试用国内著名的新浪微博，形象、直观的经验要比看枯燥的技术分析好理解些。

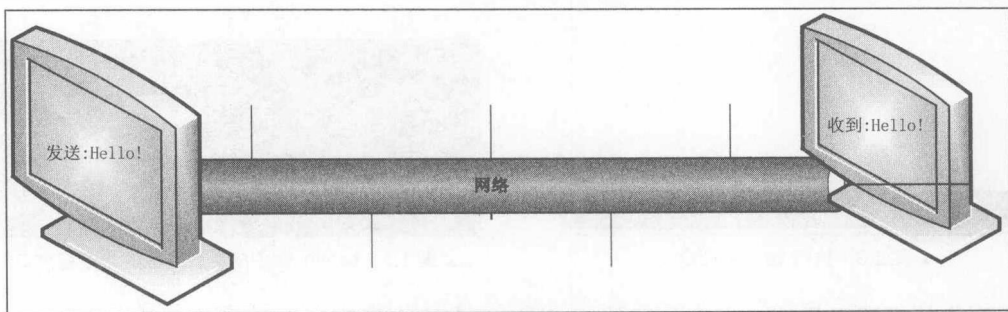
——如果你对这些都感兴趣，并想亲自动手实践，那么，启程吧！

1.1 从零开始实现公共聊天室

1.1.1 网络基础知识

我总是用 QQ 作为网络通信技术的代言人，只因为它是我们再熟悉不过的即时通信工具。

当在 QQ 消息框中输入一段文字按下发送键后，这段文字就会出现在另一台计算机上朋友的 QQ 中——我们从这里开始，分析这个过程这是如何实现的，如图 1.1 所示。



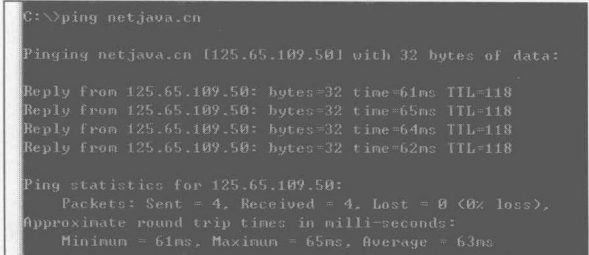
▲ 图 1.1 网络信息发送与接收

开始之前，你至少应该知道，每台机器都有一个 IP 地址，以便另外一台机器可以在网络上找到它，IP 地址在网络中标识了一台机器。但是，当 A 机器与 B 机器通信时，其实是这两台机器上的程序在通信。A 机器上的 QQ_A 如想与 B 机器的上 QQ_B 通信时，QQ_A 还必须知道 QQ_B 在 B 机器的哪个“端口”上等待，就像你到别人家做客，只知道在第几幢是不行的，还得知道是几号——在计算机中，用“端口号”这个数字标识机器上需要通信的某一个程序。

每一台机器都有从 0~65 535 个端口号，其中的每一个数字可供一个程序通信使用。通常情况下 0~1 024 的端口要尽量避免使用——我们称它为“知名端口”。例如打开网页时，连接

的是服务器上的 80 端口，因为它是默认的，所以在浏览器地址栏不需要输入这个端口号。

要测试服务器上是否开放了某个端口，例如首先通过 `ping netjava.cn` 这个主机查看网络是否通畅，如图 1.2 所示。注意，`ping` 命令使用 ICMP 报文，工作在 TCP 层，`ping` 只能证明网络是否通畅，即数据可否传送到指定主机，并不能证明主机上是否开放某个端口。



```
C:\>ping netjava.cn

Pinging netjava.cn [125.65.109.50] with 32 bytes of data:

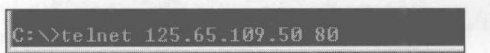
Reply from 125.65.109.50: bytes=32 time=61ms TTL=118
Reply from 125.65.109.50: bytes=32 time=65ms TTL=118
Reply from 125.65.109.50: bytes=32 time=64ms TTL=118
Reply from 125.65.109.50: bytes=32 time=62ms TTL=118

Ping statistics for 125.65.109.50:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 61ms, Maximum = 65ms, Average = 63ms
```

▲ 图 1.2 用 `ping` 命令测试服务器

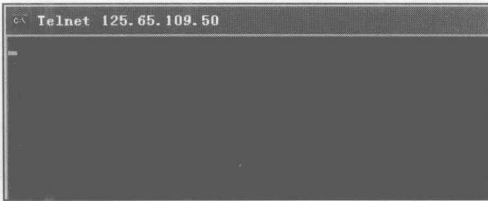
执行 `ping` 命令行，如果网络通畅，就可以使用 `telnet` 命令连接对方的端口，如果能连接上，则证明对方的端口是打开的，如图 1.3 所示。

我们所测试的主机提供的是 `netjava.cn` 网站服务，那么它开放的肯定是默认的 80 端口。请在命令行输入“`telnet 服务器 IP 地址 端口号`”，回车后，你如果看到一个全黑的命令行窗口，如图 1.4 所示。



```
C:\>telnet 125.65.109.50 80
```

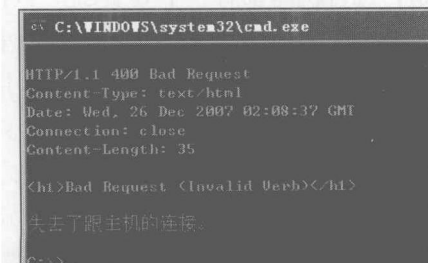
▲ 图 1.3 执行 `telnet` 命令



```
C> Telnet 125.65.109.50
```

▲ 图 1.4 `telnet` 命令成功后全黑的命令行窗口

证明你的机器已经连接上服务器的 80 端口，在这个黑窗口中随便输入几个字符，你会看到如图 1.5 所示的结果。



```
C:\WINDOWS\system32\cmd.exe

HTTP/1.1 400 Bad Request
Content-Type: text/html
Date: Wed, 26 Dec 2007 02:08:37 GMT
Connection: close
Content-Length: 35

<h1>Bad Request (Invalid Verb)</h1>

失去了跟主机的连接。

C:\>
```

▲ 图 1.5 `telnet` 命令连接 Web 服务器的 80 端口

这是因为服务器上的 Web 服务与客户端通信使用的是 HTTP 协议，而对我们随便发送的消息，服务器是不能解析的，所以返回了错误说明，之后断开了与客户端的连接。

事实上，我们计算机里的程序无时无刻不在默默地与外界通信着，当然，也包含计算机中的木马和病毒程序。你现在肯定想知道自己的机器正在与哪些网络服务通信？这很简单：可以在命令行输入 netstat 命令查看，常用的是输入“netstat -an”，回车后，返回结果如图 1.6 所示。

```

C:\WINDOWS\system32\cmd.exe
C:\>netstat -an

Active Connections

Proto Local Address           Foreign Address         State
TCP   0.0.0.0:135              0.0.0.0:0              LISTENING
TCP   0.0.0.0:445              0.0.0.0:0              LISTENING
TCP   0.0.0.0:3306             0.0.0.0:0              LISTENING
TCP   127.0.0.1:1025           0.0.0.0:0              LISTENING
TCP   127.0.0.1:1106           127.0.0.1:1107         ESTABLISHED
TCP   127.0.0.1:1107           127.0.0.1:1106         ESTABLISHED
TCP   127.0.0.1:1108           127.0.0.1:1109         ESTABLISHED
TCP   127.0.0.1:1109           127.0.0.1:1108         ESTABLISHED
TCP   192.168.1.44:139         0.0.0.0:0              LISTENING
TCP   192.168.1.44:1037        219.133.48.109:80       ESTABLISHED
TCP   192.168.1.44:1276        192.168.1.102:139      TIME_WAIT
TCP   192.168.1.44:1277        192.168.1.102:139      TIME_WAIT
TCP   192.168.1.44:1278        63.245.209.21:80       TIME_WAIT
TCP   192.168.1.44:1279        63.245.209.21:80       TIME_WAIT
TCP   192.168.1.44:3502        192.168.1.1:5431       TIME_WAIT
UDP   0.0.0.0:445              *:*                     *:*
UDP   0.0.0.0:500              *:*                     *:*
UDP   0.0.0.0:1027             *:*                     *:*
UDP   0.0.0.0:1028             *:*                     *:*
  
```

▲ 图 1.6 使用 netstat 命令查看本机连接情况

这个命令会打印出你的计算机与其他服务器建立的 TCP 连接或 UDP 连接信息。输出的数据分为 4 列，第 1 列说明连接协议是 TCP 还是 UDP；第 2 列说明连接所使用的本地地址，由一个 IP 和端口组成；第 3 列说明目标机器的地址，也是由一个 IP 和端口组成，其中的 0.0.0.0 和 127.0.0.1 指的是本地地址；最后列是连接状态的说明，由于只有 TCP 协议是面向连接的，所以 Proto 为 TCP 的才有 State 说明。当 State 为 LISTENING 时，表示本地打开了端口，如图 1.7 所示。

TCP	0.0.0.0:3306	0.0.0.0:0	LISTENING
-----	--------------	-----------	-----------

▲ 图 1.7 netstat 命令显示本机一个打开的端口号

图 1.7 表示本地计算机打开了 3306 端口，但没有任何机器与这个端口相连。

图 1.8 表示我们本地计算机上的 1037 端口与 IP 地址为 219.133.48.109 的机器上的 80 端口建立了连接——这是刚才执行 telnet 命令的结果。

TCP	192.168.1.44:1037	219.133.48.109:80	ESTABLISHED
-----	-------------------	-------------------	-------------

▲ 图 1.8 netstat 命令显示的一个已建立的网络连接