“十二五”国家重点图书出版规划项目

分布式网络系统与Multi-Agent系统编程框架

姜 维 庞秀丽 著

哈爾濱工業大學出版社
HARBIN INSTITUTE OF TECHNOLOGY PRESS

“十二五”国家重点图书出版规划项目

分布式网络系统与 Multi-Agent 系统编程框架

姜 维 庞秀丽 著

哈爾濱工業大學出版社

内 容 简 介

本书对典型的分布式网络系统和 Multi-Agent 系统的分析设计需求进行归纳总结，阐述了分布式网络编程框架和 Multi-Agent 编程框架的体系结构、设计方法和技术原理。书中相关理论和技术可以直接用于解决具体应用系统研制中的问题，同时配套软件也能对具体应用系统的研发提供直接支持。书中内容分为两个部分：分布式网络编程与 Multi-Agent 编程。通过对两类系统制定编程规范、阐述技术实现原理，提出了分布式网络编程框架与 Multi-Agent 系统编程框架，实现了理论方法与编程实践的相结合。

本书可作为网络系统研发者和工程技术人员的参考用书，也可作为相关专业的研究生和高年级本科生的教学用书。

图书在版编目(CIP)数据

分布式网络系统与 Multi-Agent 系统编程框架/姜维，庞秀丽著. —哈尔滨：哈尔滨工业大学出版社，2014.11

ISBN 978-7-5603-4984-8

Ⅰ.①分… Ⅱ.①姜…②庞… Ⅲ.①分布式计算机-网络系统-高等学校-教材 ②软件工具-程序设计-高等学校-教材 Ⅳ.①TP338.8

中国版本图书馆 CIP 数据核字(2014)第 257411 号

责任编辑 杨秀华
封面设计 刘长友
出版发行 哈尔滨工业大学出版社
社 址 哈尔滨市南岗区复华四道街 10 号 邮编 150006
传 真 0451-86414749
网 址 http://hitpress.hit.edu.cn
印 刷 哈尔滨工业大学印刷厂
开 本 787mm×1092mm 1/16 印张 17 字数 421 千字
版 次 2015 年 1 月第 1 版 2015 年 1 月第 1 次印刷
书 号 ISBN 978-7-5603-4984-8
定 价 48.00 元

前　言

基于计算机网络实现的分布式系统与网络系统可更好地服务于人们，如火车订票系统、飞机订票系统、电子商务、日常网络办公系统、在线语音通信系统、网络教学系统等；再比如在科研中分布式仿真系统、多机分布式计算系统、建筑物应急疏散仿真系统、机器人足球系统、卫星任务规划系统等。凡是需要在计算机网络支持下的许多计算机参与的系统，都需要利用到计算机网络编程技术。网络编程通过直接控制和利用计算机网络底层通信，可更好地实现程序间的交互，通过协作可完成单机难以完成的计算功能。

本项目受国家自然科学基金（No. 71271066，No. 71202168）和总装备部项目支持，本书基于多年来在科研项目中有关计算机网络编程的研究成果与实际工程项目中具体系统研发的成果积累，阐述了有关分布式网络编程与 Multi-Agent 系统编程的设计方法与实现原理。通过以两个编程框架为主线，书中阐述了一般性分布式网络系统的交互调用方法与系统框架设计，也阐述了 Multi-Agent 的编程框架设计。所阐述的理论、方法、技术包括：

第一，阐述了分布式网络编程与 Multi-Agent 系统的分层体系结构，制定了每层内对象的服务规范，以便有整体程序设计的概念。

第二，阐述了具体技术的实现原理。包括远程过程调用 RPC，事件触发机制 EVENT，Agent 的交互行为、自治行为、控制行为等。

第三，从系统视角方面阐述了客户管理、共享对象机制、分布式共享文件系统等。

第四，提供了多任务编程机制，包括事件触发机制、消息触发机制等。

第五，提供了具体应用事例和具体应用案例。

在本书撰写过程中，不仅提供了设计方法和技术原理，也特别注意所阐述内容的可编程实现，配套框架软件库对应第 4 版提供的 ART 框架和 ART-Agent 框架，前者用于分布式网络编程，后者用于 Multi-Agent 系统编程。底层代码完全通过 C++实现，同时又提供了多种编程接口，包括 C++，Java，C，Delphi 语言等。由于各操作系统，如 Windows，Unix 和 Linux 等有相似的机制，如信号量、socket 接口等，故可实现跨操作系统平台，跨编程语言的软件研发。因此，不仅希望本书能在理论方法上有借鉴意义，同时也希望能够有助于：

第一，学习上，便于学习掌握分布式网络编程的基本方法，也可通过框架提供的程序设计模式，深入掌握一般性分布式网络程序设计的架构，并能够构建自己的分布式网络程序；

第二，科研上，基于本书提供的编程框架，科研者可以直接通过接口调用，实现底层功能，可将更多精力集中在具体应用问题的研究上，如 Multi-Agent 系统框架，只需在本书提供的 Agent 的框架模型基础上对具体应用问题建立 Agent 模型就可实现；

第三，工程上，可以直接应用基本配套软件库开发专业级分布式网络系统与 Multi-Agent 系统。配套辅助开发工具帮助快速、便捷地建立自己的系统。

技术在持续发展，框架软件本身也在改进。书中存在错误和不足之处，敬请各位专家与学者批评指正。

配套网站上有软件库、教学幻灯片、演示事例、书籍勘误表、最新技术文档、常见问题、在线研讨、作者联系方式及联系电话等。网址：http://www.jiangw.cn，http://www.jiangw.com，http://www.orsci.cn，http://www.orsci.com。

著　者

2014年9月

目　录

第1章　分布式网络编程准备工作

1.1　分布式网络系统编程

分布式系统描述的是一些独立的计算机组网共同运行，形成的一个整体系统，而对用户来说，整个系统就像一台计算机一样，其中包含两个含义：从硬件角度来看，每台计算机都是自主的；从软件角度来看，用户将整个系统视为一台计算机。分布式系统的一个很重要特征就是逻辑透明性，它包括位置透明性、迁移透明性、复制透明性、并发透明性与并行透明性等。

网络系统，相比分布式系统来说，各计算机端软件的耦合程度并未达到分布式系统的紧密级别，即分散在各计算机的软件间耦合程度相对松散。分布式系统通常要求各计算机运行相同的操作系统，以便实现进程的自动迁移等操作，而网络系统并不要求各计算机运行的操作系统是一致的，但为了能够相互通信，往往要求支持相同的网络协议，例如 TCP/IP 协议。

1.1.1　网络环境下的各程序端交互

虽然分布式系统在设计上对各软件间耦合度要求相对高些，但就各计算机程序的交互方式来看，它与一些网络应用系统编程有较高的相似性，都需要各程序之间进行通信、交互调用与协同工作，如图 1.1 所示。

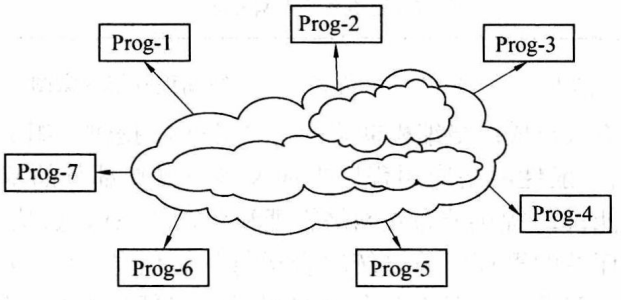

图 1.1　分布式网络程序各程序端基于网络进行交互式通信

现实生活中，有许许多多的多机协同工作实例。例如，一个服务网站因为用户众多，服务端可利用一组组网的计算机联合工作，通过将大量远程用户的任务分散在各个计算机上协作执行；一个具有海量数据的计算任务分散在多台计算机共同执行；一个大型分布式仿真系统，由十余台计算机联网仿真，其中的数据生成、数据加工、管理控制、数据存储、图表展示等都分散在不同计算机上；火车售票程序因旅客分散在不同地域而采用分布式系统等。

分布式程序和网络程序都需要利用计算机网络传递数据，都需要交互控制，一些复杂的网络程序与分布式程序设计接近，两者在网络通信、交互式调用等网络编程方面具有较

高的一致性。两类系统中,每一个计算机上的程序都需要与其他计算机上的程序进行交互,以共同完成整个系统的工作任务。鉴于本书更多关注一般性分布式程序与网络应用程序的编程架构,因此本书不去严格区分两类系统。在一般性的网络程序交互中,远程过程调用(Remote Procedure Call,简称 RPC)是一种典型且常用的交互式调用方法。RPC 实现了一台计算机上的进程对另一台计算机上的进程提供的函数服务进行调用,编程方式上类似于编写程序通过调用本地函数提供服务,只是该服务程序是在另外一台计算机上执行的。RPC 调用为分布式网络程序的编写提供了基础,它是在 TCP 协议仅提供可靠的数据流服务的基础上,通过增加一定的语义功能,从而实现一台计算机利用另外一台计算机提供函数服务的功能。

1.1.2　分布式网络编程的中间件技术

分布式程序与网络程序有许多现实需求,在计算机网络技术与计算机网络普及已取得重大成就的情况下,如何高效、快速地编写新的分布式网络程序有较高的现实意义和实际应用价值。通过一些已做工程项目和已有研究工作发现,一些分布式网络应用程序在计算机网络通信与网络相关的编程模式方面存在较高的相似性,如 RPC 是许多分布式程序的基础调用模式。此外,还可以提供分布式变量共享与分布式文件系统等。于是本书试图分析分布式网络应用程序编写时所需要的一些交互调用和主要功能,并通过构建插件对象(Plugin)的形式提供一个分布式网络编程框架,以期有助于一般性的网络程序编写,如图 1.2所示。

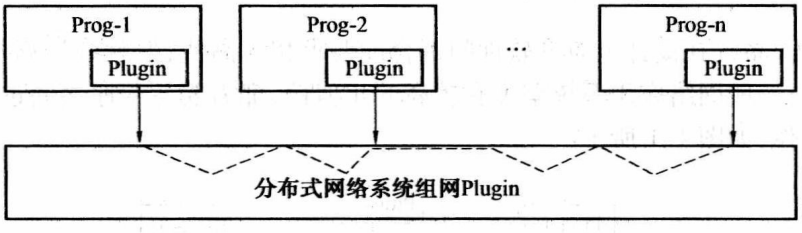

图 1.2　插件(Plugin)方式设计网络编程框架示意图

图 1.2 中以插件方式形成网络编程框架的初衷是:(1)提供一组交互调用操作,试图透明化网络通信过程,将一般性网络应用程序中涉及网络操作部分的基本调用抽取出来,形成编程框架的调用功能;(2)提供分布式网络管理和系统状态信息,从分布式网络程序的系统角度提供各分布程序端的管理以及系统综合运行状态信息,如连接注册管理、运行状态与分布式时间管理等;(3)提供事件触发与消息触发的多任务编程机制,形成编程框架,如借鉴 Windows 的消息机制编程方法、Delphi 编程语言中的事件机制编程方法等。图 1.2 中各个网络程序通过嵌入插件(Plugin)对象,使框架的应用者更能集中精力于具体应用问题的分析与设计上,而有关网络相关的调用和分布式系统管理则由框架系统来提供。

框架系统是以插件对象的形式来对具体分布式网络系统的研制提供支持。框架系统中提供了典型分布式网络编程的交互式调用与分布式系统管理和运行信息。例如,ART 框架提供了远程过程调用、远程消息(RSM)调用、事件触发机制、分布式共享对象、分布式文件系统、客户管理与客户状态等;Multi-Agent 框架系统提供了 Agent 对象交互行为和自治行为方法、Agent 之间的交互调用、Agent 与环境调用、Agent 环境的消息触发机制、分布式文件系统等。由于各分布端采用 TCP 协议来传输通信数据,而目前较多操作系统都直接支持

TCP 协议，所以，第一，通过将框架系统的技术在多个操作系统中实现，如 Windows，Unix，Linux 等操作系统，可以实现跨平台的分布式网络编程；第二，编程框架通过提供多种编程语言接口，包括 C++，C，Java，Delphi，可以实现同一种语言编程或者多种语言编程，实现跨语言编写分布式网络编程。随着技术发展与实际分布式网络程序设计的需要，一些新的功能将补充到框架系统中，实现对框架系统功能的进一步扩展。

1.2　Socket 通信技术及可靠传输问题

1.2.1　TCP 实现虚拟网络抽象

在各种网络协议中，TCP/IP 互联网协议（Internet Protocol）是一个被广泛使用的网络连接协议，已应用在 Internet 和大量的局域网络中。TCP/IP 是一个协议族，包含一套协议，通过硬件路由器的支持，它能够将众多物理网络进行互连，提供了一个单一的、虚拟的网络系统，如图 1.3 所示。例如，Internet 网络实际上是由许许多多的物理网络通过路由器硬件连接而成的，TCP/IP 为各个物理网络的连接提供软件协议支持，最终 Internet 形成一个大的抽象网络。所谓网络协议（Network Protocol）是指规定消息的格式及每条消息所需要的适当动作的一套规则。每个硬件之间进行数据传送时，彼此需要一套一致的语义约定，以确保硬件在通信数据传输这一问题的协调上有着统一的规范。

TCP/IP 提供了一套协议，用于解决网络通信中的各个层面上的问题，按照 TCP/IP 分层模型，它包括五层：物理层、网络接口层、互联网层、传输层和应用层。其中，互联网层通过网络接口层和路由器的支持，实现了 IP 数据报的尽力传输，而传输层基于互联网层提供的服务，实现了从一台计算机的一个端口到另一台计算机的一个端口的传输服务。互联网层实现了路由选择，经过互联网层的功能封装，传输层已经看不到路由器的存在，传输层可以只专注于从一台计算机的一个端口到另一台计算机的一个端口的数据传输，如图 1.3 中的 H1 的一个端口到 H2 的一个端口的数据传输。

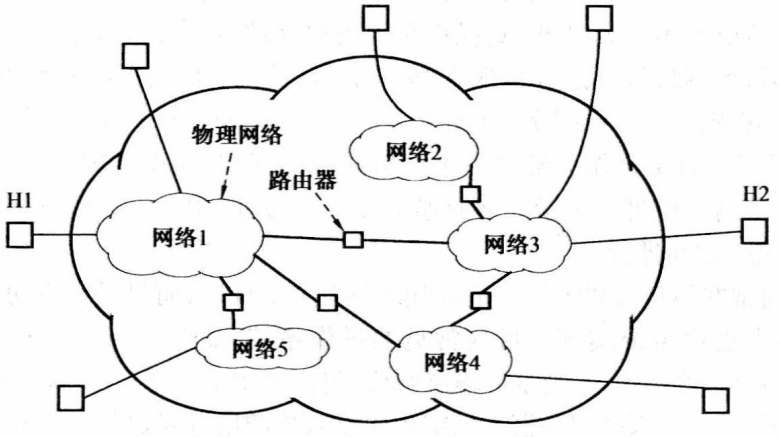

图 1.3　TCP/IP 实现网络互连提供单一网络的逻辑抽象

大多基于 TCP/IP 协议的网络程序编写是基于传输层的服务接口，并不直接与网络硬件打交道。传输层包括两个重要的协议：TCP 和 UDP。其中，传输控制协议 TCP（Transmission Control Protocol）为面向连接的协议提供了可靠的传输服务，即保证在连接终

止前，接收方所接收的数据（字节流）与发送方发送的数据完全一致，不会发生数据的重复、丢失、乱序和延迟等问题，而且 TCP 协议还处理了流量控制和拥塞控制等问题。相比 TCP，用户数据报协议 UDP（User Datagram Protocol）面向无连接，提供面向事务的简单不可靠信息的传送服务①。当基于 TCP 协议编写网络程序时，数据的可靠性传输问题已由 TCP 协议解决。

1.2.2　TCP 提供端到端的可靠数据流传输

互联网提供了一个通用的通信框架，关于通信的使用方式则留给具体应用者。例如，传输层只专注于数据传输，而有关数据本身的语义解释，则留给了具体应用程序。在通常的网络编程中，一个应用必须主动地启动交互，另一个则被动地等待，这种典型的编程模式称为“客户/服务器”交互模式（Client-server Paradigm of Interaction）。其中，客户（client）程序主动与服务器建立通信连接，而服务器（server）程序被动地等待通信。

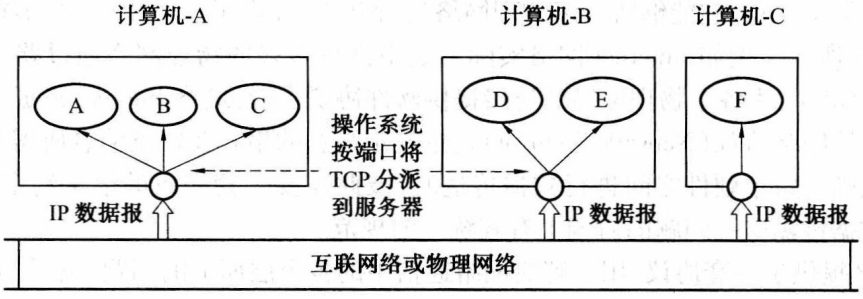

图 1.4　TCP 利用端口（port）区分不同的服务

为了区分不同的连接服务，在 IP 层利用 IP 地址指明目标计算机的基础上，传输层使用端口号（port）进一步区分不同的服务。在 server 启动监听时，需要指明一个端口号；在 client 建立连接时，或者预先指定一个端口号或者由操作系统为其分配唯一的端口号。在 IP 地址指明计算机的基础上，端口号标识 TCP 连接是唯一的、可区分其他 TCP 的连接。如图 1.4 所示，若 A 与 D，B 与 E，C 与 F 分别建立连接，则 D 在与 A 传输时，TCP 数据报被封装到 IP 数据报中，按 IP 地址传输到计算机-A，操作系统再根据端口号来区分是应该传输到哪个服务点，经判别因目标端口号就是 A 的服务端口号，于是 TCP 数据报传送给 A。可看出端口用于区分同一计算机中的不同服务，如图 1.4 的计算机-A 中区分 A，B，C。上述表明，“IP 地址：port 端口号”可以组合作为确定一个端口连接的标识，其中的 IP 地址用于指明计算机，而 port 端口号用于指明该计算机中的连接服务。通过“IP：port”作为标识，TCP 保证了传输可在“端到端”之间进行。

通信协议通常并不定义协议软件的应用程序接口（API），而只是定义协议软件应当提供的操作，具体由操作系统实现。目前常见的操作系统，如 Windows，Unix 和 Linux，支持 Socket API（套接字 API）。它实现了传输层协议，包括 TCP 和 UDP。socket 包括多个函数完成通信协议调用，其客户端和服务器端的编程主要函数调用流程如图 1.5 所示②。

图 1.5 的客户/服务器操作主要过程如下：

---

① TCP 和 UDP 各有其特点，适用于不同应用场合，在需要数据可靠传输时往往采用 TCP 协议编程。

② 关于 socket 编程，请参看《计算机网络》相关教材。

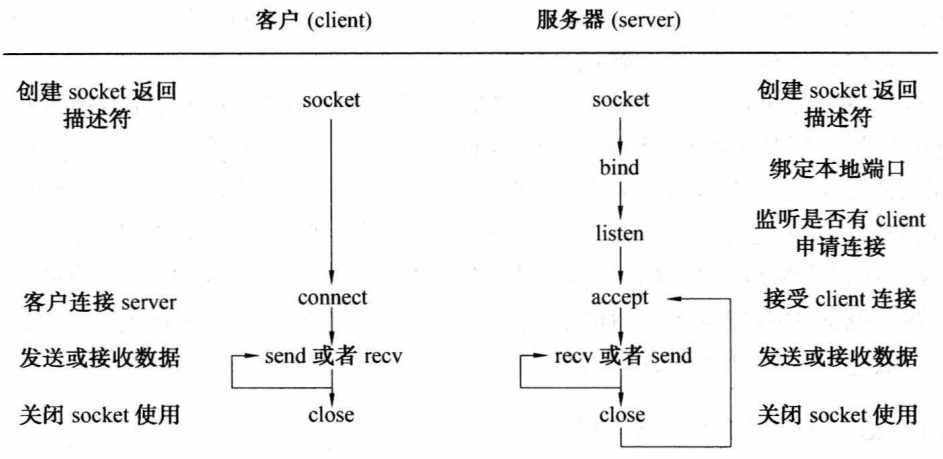

图 1.5　socket 进行客户/服务器编程的主要流程

(1)服务器(server)被动监听连接,故其服务端口一般预先设定①,当创建 socket 后,通过 bind 设定服务 IP 地址和服务 port,启动监听 listen 后,开始监听是否存在客户(client)进行 TCP 连接操作(connect 操作)。

(2)客户端主动连接服务器,当其创建一个 socket 后,通过调用 connect 连接服务器。客户端的端口一般采用自动分配方法。

(3)当客户端调用 connect 后,服务器的 listen 函数返回一个新 TCP 连接请求,如果服务器允许接受连接,则调用 accept 生成一个与客户端 socket 连接的数据 socket,服务器端新生成的数据 socket 与客户端的 socket 形成一对一的连接。这种连接已由 TCP 按照三次握手协议建立,保证了在该连接中数据传输的可靠性。

(4)当客户端与服务器连接后,两者都可以在连接的 socket 上通过调用 send 函数发送数据以及调用 recv 接收数据。

(5)通信结束后,需调用 close 友好地关闭连接操作,按照 TCP 协议,close 会按照三次握手结束 TCP 连接操作。

值得说明的是,因为通常一个 server 允许多个 client 进行连接,为了更好地服务,往往 server 采用多线程(或多进程)②工作方式,可以并发地为多个客户端提供服务。

1.3　P2P 通信模型与实现原理

P2P(Peer to Peer)对等网络模型,其常用于指代一种编程模式,也常用于指代网络文件下载的传输方法。在“客户/服务器”(C/S)编程模式中,客户数量较多,而且由单一服务器提供服务,虽然服务器端可以通过机群等方式提高了服务器的能力,但总体上属于众多的客户端连接到同一个服务器上,这面临两个问题:第一,服务器的稳定性成为网络运行的关键,因为只有一个服务器,如果其运行崩溃,则影响所有用户的连接;第二,因只有一个服务器提供服务,服务器负载可能过大。与“客户/服务器”模式相比较,P2P 模式的每个机器地位大致对等,任意一个节点都既可以作为服务器又可以作为客户端,实现了任意两端之间

① 服务器的端口号也允许由操作系统动态分配,但考虑到其用于客户的连接,所以通常预先指定一个值。

② 关于多线程、多进程请参看本章第 1.5 节内容,或参看《操作系统》或《多线程编程》相关教材。

的直接通信。因此,服务的提供可以通过其他多个节点同时进行,从而易于克服 C/S 模式中单一服务器面临的稳定性和负载过大问题。

P2P 的这种编程模式已被应用于一些网络数据下载软件与一些视频点播软件,并且取得了很好的效果。P2P 下载的核心是数据存储在客户本地,通过存储信息(名称、地址、分块)的查询,让终端之间直接进行数据传递。在"客户/服务器"模式中,如果数据传输任务较多,则服务器的负担往往较重。例如,较多客户下载一些大型文件数据,而在 P2P 模式下,一些数据在客户端也有文件数据或副本,那么下载工作可以从其他客户端进行,甚至服务器只存储数据的索引与链接,不存储数据,没有服务容量的压力,这样使得网络上的数据流量更加分散化,提高了网络数据传输效率。

1.3.1　基于 TCP 的 P2P 模型设计

对于 P2P,本书更关注其数据传输层面上的对等编程模式与实现技术①。一般的 P2P 是基于 TCP 或者 UDP 实现的,考虑到 TCP 传输可靠,其面向连接的特点有利于网关②上保持公共 IP 和局域网 IP 间的数据连接,本书阐述了一种 TCP 实现 P2P 数据传输技术的方案。从数据通信角度来看,建立 P2P 要求每一个端都可以主动连接网络中的其他端,同时又允许其他端连接本地端程序。站在 TCP 角度来看待这样的技术需求,就像每一个端既是客户端 client 又是服务器 server,于是每个 P2P 端以一个 socket server 与若干 socket client 相结合的方式构建,如图 1.6 所示。

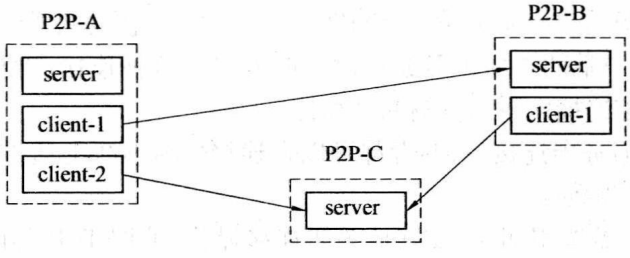

图 1.6　基于 TCP 构造 P2P 对等连接示意图

图 1.6 中 P2P 对象③的一个节点 A 连接节点 B 和节点 C 是通过两个 TCP socket 中的 client 连接到 B 和 C 节点的 TCP socket 的 server 实现的。节点 B 连接到节点 C 是通过 B 的 client 连接到 C 的 server。通过 P2P 对象封装,当 A 发送给 B 数据时,则采用 A 中相应连接 B 的 client 进行传输,当发送给 C 数据时,则使用 A 中相应连接 C 的 client 进行传输。一旦 A 与 C 建立了连接,就相当于 C 与 A 建立了连接,此时如果 C 给 A 发送数据,则通过 C 的 server 进行传输。

为实现图 1.6 中的技术,在 P2P 对象实现时,存在两个重要登记表:一个是"client 登记

---

① 现有 P2P 模式在下载软件、视频直播、网络电话、网络聊天和网格计算等领域有着众多的应用。在大型 P2P 网络中还对节点的作用进行划分,包括管理节点、索引节点、搜索节点和一般用户节点等。本书用于一般分布式网络程序编写,因此更关注在通信层面上的 P2P 设计与实现技术。

② 网关一般由一台计算机或专用硬件实现,它能够完成地址映射,使得局域网 IP 地址可以访问 Internet 网。如果网关不保持 TCP 这种地址映射,则会发生 Internet 网的计算机无法将数据传给局域网 IP 的计算机。TCP 这种数据穿透能力不易由 UDP 实现。

③ 本书将 P2P 数据通信模型封装为一个类,名称为 TP2P。

表”,它用于记载主动对外连接其他 P2P 节点的 client 列表;另一个是“节点连接登记表”,它用于记载本节点与其他 P2P 节点的连接情况,即记载是经由 server 连接还是经由 client 连接的,如果是 client 连接还需要记载对应的 client 对象。例如,表 1.1 为图 1.6 各节点对应的节点连接登记表。一个 P2P 节点的 client 登记表允许是空,如图 1.6 中的 C 节点,此时意味着它没有主动对外进行连接。对于 server 的连接也可以是空,如图 1.6 中的 A 节点,此时意味着没有其他节点主动连接该 P2P 节点。

表 1.1　P2P 节点对象连接表举例

本节点	连接节点	连接对象
A	B	client-1
	C	client-2
B	A	server
	C	client-1
C	A	server
	B	server

在图 1.6 中,如果 A 对 B 已经建立了连接,则 B 的连接登记表中登记了 A 对 B 的连接。当需要给 A 传输数据时,使用已有连接就可以了,但是如果恰好存在这样的一种情况:在 A 与 B 没有连接时,若瞬间发生 A 与 B 主动连接、恰好 B 也与 A 进行主动连接,那么可能同时建立了两个连接,如图 1.7 所示。虽然这种情况小概率发生,但仍有可能存在,因此必须谨慎处理,对此还不能随意删除一个连接,如果这样做也必须两者进行有效协商。由于是小概率事件,一种稳妥的方案是同时保留这两个连接,如图 1.7 所示,同时存在连接-1 和连接-2。虽然两个连接可能同时发生,但由于对连接登记表的登记操作应用了互斥锁,因此可以保证 P2P 节点的连接登记表只登记第一次连接所对应的对象①,这样本 P2P 节点向对方 P2P 节点传输数据时始终采用同一个 TCP 连接,故能够保证数据传输的可靠性。换个角度来看,对于接收者节点来说,因为对方始终使用同一个 TCP 连接向本节点发送数据,因此接收的数据仍然是按照发送顺序可靠传输的。由于可能存在双连接,在与一个 P2P 节点断开时,若同时存在两个连接边,P2P 节点需要同时 close 两个连接,以进行彻底关闭。

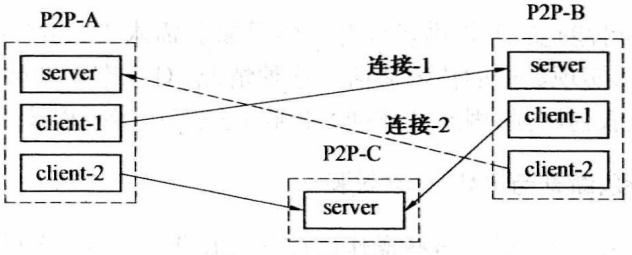

图 1.7　P2P 对等连接中节点间存在两个连接边的情况

① 虽然只登记第一个连接,但也同时记载存在另外一个连接,以便在与对方断开连接时,能够同时关闭两个连接。值得思考的是,为什么记载第一个连接,而不是记载最后一个连接?这是考虑到可能会在两个连接之间本节点存在数据发送情况,因如果已经存在数据发送,则不能再保证两个连接数据传输的顺序性。

由于每个 P2P 节点都有一个 server，所以都可以被动连接，又因其可动态申请 client，所以又可以主动连接，形成 P2P 各节点之间的自由连接。那么，如何标识一个 P2P 节点呢？采用 server 的 IP 地址和 port 端口号作为 P2P 节点标识，其原因如下：第一，P2P 节点只有一个 server 且其 IP 地址和 port 端口号在 P2P 节点服务期间一直保持不变；第二，server 的 IP 地址和 port 端口号就是代表着服务，用于其他节点的连接，具有典型标识作用。

如果采用 server 的“IP:port”作为节点标识，那么如何使相连接的节点知道对方标识呢？设想图 1.7 中 P2P-A 的 client-2 连接 P2P-C 的 server，此时 P2P-A 因为是主动连接，所以知道 P2P-C 的标识就是即将连接的 server 的“IP:port”，但是 P2P-C 如何知道 P2P-A 的标识？方案是采用通信，约定当一个节点主动连接另一个节点后，则主动连接节点必须立即发送本节点的标识。所谓立即发送，既强调最初的数据就是本节点标识，又强调了尽可能快速告诉对方本节点的标识。被动连接节点只有收到对方的标识后，才建立“连接登记表”项。分析一下，即使主动连接者立即发送标识，也会存在延迟，此时，第一，不会影响主动连接者的数据发送，因为其首次发送一定是标识，然后才是真正数据，而发送标识后，对方就建立了“连接登记表”项；第二，对被动连接者影响较小，因为应用程序设计往往是单方面主动连接，且要求主动连接者立即发送标识，所以创建“连接登记表”的延迟不大，但即使存在瞬间发生被动连接者也需要与主动连接者进行数据传输，由于此时“连接登记表”尚未建立连接项，故可认为尚未连接，于是需要建立第二条连接边，所以这样也不影响数据的传递。需要强调的是，对“连接登记表”的操作应该互斥加锁，确保只登记首条连接，但倘若存在双边连接的小概率事件，也应该作以登记，以便释放时能释放两条边。

通过上述基于 TCP 进行 P2P 的节点构造和 P2P 的对象封装，可以形成具有对等互联的 P2P 传输模式，依据需要可以进行多个 P2P 节点之间的连接，如图 1.8 所示。

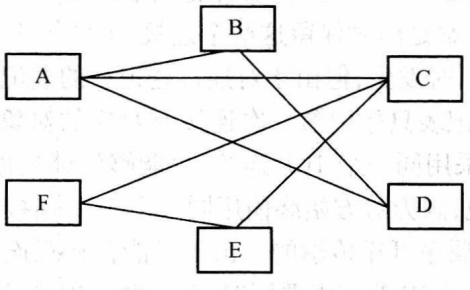

图 1.8　P2P 网络依据需要可以进行多个节点间的连接

P2P 的各节点既可能是服务提供者也有可能是服务需求者，每个节点都可能与其他多个节点进行连接，然后实现数据的传输通信。这种情况，对于有多个数据来源的节点来说，其数据接收的并发性较为明显，因此其数据的接收往往采用多线程技术实现。

1.3.2　节点连接、断开与 P2P 对象封装

下面阐述 P2P 节点主要操作，这些操作已被封装在 TP2P 类中实现。

节点启动操作（Open 操作）：启动 server 提供被动连接服务。

节点连接流程（Connect 操作）：如果需要主动连接另一 P2P 节点，则创建一个 client，登记在“client 登记表”中，并用该 client 连接另一节点的 server，连接后需要立即主动发送本地标识、然后建立“连接登记表”项；如果是另一节点连接本节点的 server，则尚需等待对方发送过来其节点标识，当收到对方节点标识后建立“连接登记表”项。

节点断开流程（Disconnect 操作）：查找“连接登记表”找到需要断开节点的连接，关闭该

TCP 连接①,依据“连接登记表”判断是 server 还是 client 连接,如果是 client 连接,则需要释放 client 对象;如果存在双连接边,则需要释放两个边。

节点关闭操作(Close 操作):确保与所有其他节点断开,关闭 server。

图 1.9 展示了一个 TP2P 类的属性、方法和触发事件,该类封装了 P2P 通信模型的操作。

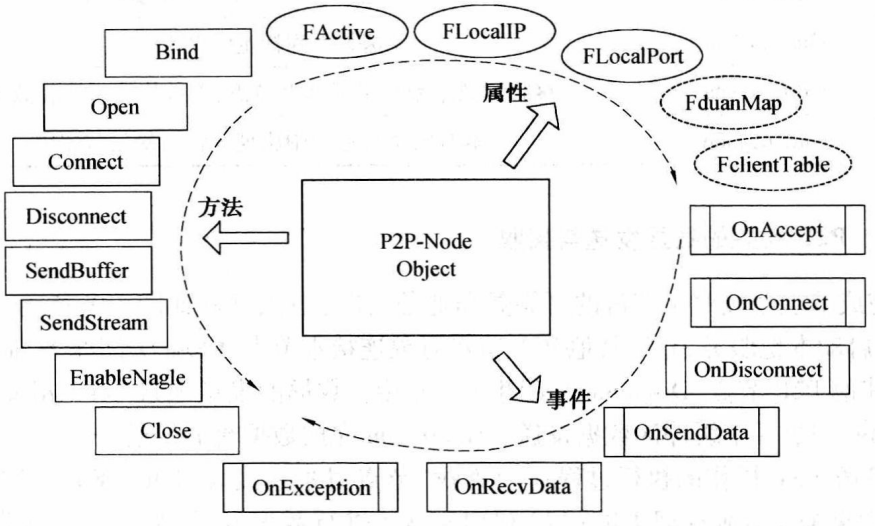

图 1.9　P2P 节点对象的属性、方法和事件

表 1.2 描述了图 1.9 中对象的属性、方法和事件的主要功能和作用。

表 1.2　P2P 节点对象的属性、方法和事件的主要作用

类别	名称	主要作用
属性	FActive	标识是否当前对象已经启动服务
	FLocalIP	标识本地服务 IP 地址
	FLocalPort	标识本地服务 Port 值
	FclientTable	client 连接登记表
	FduanMap	节点连接登记表
方法	Bind	绑定本地服务 IP 和 Port
	Open	启动本地服务器,监听外界客户连接请求
	Close	断开本地服务器
	Connect	主动连接其他 P2P 节点
	Disconnect	主动断开其他 P2P 节点
	SendBuffer	发送数据到指定的 P2P 节点
	SendStream	发送数据流到指定的 P2P 节点
	EnableNagle	打开或者关闭 Nagle 算法

① 调用 socket 的 close 方法关闭一个 TCP 连接。

续表 1.2

类别	名称	主要作用
事件	OnAccept	监听到存在其他 P2P 节点连接请求,允许设置阻止连接
	OnConnect	与其他 P2P 节点建立了连接事件
	OnDisconnect	与其他 P2P 节点断开了连接事件
	OnSendData	发送数据到指定的 P2P 节点
	OnRecvData	接收到数据来自于某 P2P 节点,参数指明来源和数据等
	OnException	本对象执行过程中出现异常时触发该事件

1.3.3 P2P 模型的数据发送与接收

P2P 完成节点间按照字节流的可靠数据通信,任意节点互联如图 1.8 所示。节点对象通过 Open 启动本地服务、允许其他 P2P 节点对象连接本节点、Close 关闭服务,通过 Connect 主动连接其他 P2P 节点、Disconnect 关闭一个连接。数据的发送通过 Send 相关方法完成,如 SendBuffer 完成一个缓冲区数据发送、SendStream 完成数据流的发送。

数据发送 Send 操作的执行过程:(1)查找“节点对象连接表”FduanMap,找对对应的连接 socket,如果未连接则返回失败;(2)利用 socket 进行数据发送,为避免多线程并行利用 socket 发送数据存在冲突,如终端 SendStream 的多次发送,发送期间需要利用互斥锁锁定每次发送操作。

数据接收 Recv 的过程分为单线程模式和多线程模式。其中,单线程模式是指由 P2P 的调用者主动调用 RecvBuffer 方法进行接收数据,此时如果接收缓冲区存在数据需要接收,RecvBuffer 返回接收的数据来源、接收的数据,如果接收缓冲区中不存在数据需要接收,则立即返回。函数返回值大于 0 代表返回的结果数据数量、0 代表不存在接收数据、-1 代表连接断开或读取的端口不存在等错误。

多线程接收模式是指本 P2P 对象启动监听线程,并通过触发 P2P 对象的 OnRecvData 事件返回接收数据的工作模式。当面临多个客户进行连接时,多线程模式往往能更好地提供服务。图 1.10 展示了 P2P 节点中的多线程监听模式,主要包括两类监听线程:第一类为监听线程,它对 server 中的连接请求 socket 上调用的 listen 方法进行监听,如果存在其他客户的连接请求,则触发 OnAccept 事件;第二类为数据接收线程,用于监听每一个与其他 P2P 节点连接 socket 的数据缓冲区,当存在可用接收数据时,触发 OnRecvData 事件。

图 1.10 中的 socket-1 到 socket-n 代表着与其他 P2P 节点连接的 socket。如图 1.7 所示,当本节点采用 client 模式连接其他节点,则 socket 直接对应着通信 socket,但若采用 server 被动连接模式,则当进行 accept 方法时会返回一个新建连接的 socket,其代表着与对方节点连接的数据 socket。因此,在图 1.10 中的 socket-1 ~ socket-n 都代表着实际数据通信 socket。

图 1.10 中的每个数据通信 socket 都采用一个阻塞接收线程提供数据监听服务,默认情况下,OnRecvData 事件是并发的:对每个源 P2P 节点来说并发接收数据,而对于同一个源 P2P 连接节点,因为只有一个数据监听线程,所以它是串行接收的。图 1.10 中也可以通过并发事件开关修改默认的并发性,来强制所有数据接收都是串行的。由于每个源 P2P 节点