摘 要

蓝牙技术是近年来兴起的一项以电缆替代和组建小范围网络为目标的短距无线通信技术，其协议和应用规范于 1999 年 7 月一经提出就引起了业界广泛关注。

在蓝牙技术所描绘的广阔应用前景中，桌面系统和嵌入式系统是相辅相成的两个主流方向。考虑到 Windows 系列产品在桌面操作系统领域的统治地位，以及蓝牙技术设计用来替代串行电缆的最初动因，基于 Windows 平台、综合了诸串口应用的集成系统的研究具有其技术先进性和巨大的现实意义，本文结合笔者在论文期间参加的国际认证与测试公司（IVT）的蓝牙桌面系统开发项目，对蓝牙通信系统桌面集成解决方案进行了研究与设计。

论文首先认真研究了蓝牙协议和应用规范，深入分析了串口仿真应用及其之上的各项应用。然后以命令行方式分别实现了基于仿真串口的三个应用：拨号网络，传真和局域网访问。

在此基础上，本文根据软件工程的要求，运用面向对象的分析和设计方法进行了集成系统的开发工作；工作分为两个阶段：

第一阶段，利用面向对象的思想和 UML 语言对集成系统进行了需求分析，并依据分析结果构建了集成系统软件模型。在模型设计中，我们将集成系统从顶层分解为 UI(User Interface)与 BOL(Business Object Layer)两个包，重点完成了 BOL 包逐层细化的设计过程。贯穿与分析与设计全过程，我们采用了结构化的设计方案以减少 UI 与 BOL 之间，以及 BOL 包内各功能模块之间的耦合度，增强各模块的内聚性和可重用度。

第二阶段，完成了集成系统模型中 BOL 包大部分和 UI 包部分模块的实现工作。主要包括：BOL 中 Profile Server 包、Profile Client 包、Bluetooth Device 包和 Virtual Device 包以及 UI 中设备管理、传输层配置和 PNP 硬件检测模块。

此外为配合核心程序运行，文中还设计并实现了包括安装程序、手工卸载程序以及虚拟设备管理模块在内的集成系统支撑模块。

Abstract

Bluetooth is a new short-range wireless communication technology developing rapidly in recent years. Aiming at cable replacement and personal area networking construction, Bluetooth is attracting more and more interest since it was first released in July 1999.

The wide range of Bluetooth applications fall into two major categories: desktop applications and embedded applications. The project in this dissertation is focusing on the former one. Considering the dominant status of MS Windows series products in the field of desktop OS and the original motivation of Bluetooth technology to replace the serial cable with wireless connection, our main goal is to research and develop a Bluetooth system integrating the serial port based profiles in Windows environment.

In this dissertation, we first introduced Bluetooth protocol and profile specifications. Particular concerns are paid to serial port emulation profile and profiles based on it. Then three virtual serial port based profiles—Dial-up networking, Fax and LAN access—are implemented as command line executables.

According to the three implementations and following the software engineering principles, this thesis developed an integrated system applying to Windows environment. The development progress consists of two steps:

In the first step, the author made system requirements analysis using object-oriented method and Universal Modeling Language (UML). Based on the result of analysis, static and dynamic software model of the integrated system was developed. The model was designed in a "scale-up" manner, which means organizing the description over several layers of abstraction, so that we can at any time focus on as big or as small a part of the system while retaining the overall picture. The system was abstracted into two packages on the top level: User Interface (UI) and Business Object Layer (BOL). As BOL was the key package of integrated system, most efforts were made on it to iterate the abstraction layer by layer. Throughout the process of modeling, structured schedule was also adopted to reduce coupling among the function modules and increase cohesion and reusability

of those modules.

In the second step, the author accomplished the implementation of most packages in BOL and some modules in UI, including: Profile Server package, Profile Client package, Bluetooth Device package and Virtual Device package in BOL; and device management module, transport layer configuration module and PNP device detection module in UI.

In addition, to support the core system functions, the dissertation also designed and implemented supplementary module for it. Such module consisted of three major parts: installation program, manually uninstall program and virtual device management module.

第一章 绪论

1.1 工程背景

随着计算机网络和移动通信技术的迅猛发展，人们越来越感到发展微小范围内的无线数据与语音通信的迫切需要。于是，在1998年，爱立信、IBM、Intel、诺基亚和东芝等公司联合推出了一项最新的无线网络技术，即蓝牙（Bluetooth）技术。随后这五家公司组建了一个特殊组织（SIG）来负责此项技术的开发。1999年7月，蓝牙协议的1.0版发布，从而将其推向应用阶段。如今，SIG已经拥有9个成员，近100个高级成员和2100多个普通成员，其协议标准业已发展到1.1版本[1][2]。

蓝牙技术是一种无线数据与语音通信的开放性全球规范，应用对象主要是桌面设备和小型网络设备，如移动PC、掌上电脑、手机等。目的是方便这些设备之间以及这些设备与Internet之间的通信，免除在计算机、笔记本电脑、调制解调器、无绳电话或移动电话、头套式送/受话器、PDA、打印机、幻灯机、局域网等之间加装电线、电缆和连接器。而且，这种技术可以延伸到那些完全不同的新设备和新应用中去。例如，如果把蓝牙技术引入到移动电话和膝上型电脑中，就可以去掉移动电话与膝上型电脑之间的令人讨厌的连接电缆而通过无线使其建立通信。打印机、PDA、桌上型电脑、传真机、键盘、游戏操纵杆以及所有其它的数字设备都可以成为蓝牙系统的一部分。除此之外，蓝牙无线技术还为已存在的数字网络和外设提供通用接口以组建一个远离固定网络的个人特别连接设备群。

在蓝芽技术所描绘的广阔应用前景中，桌面系统和嵌入式系统是相辅相成的两个主流方向，本文的研究工作主要集中在前一方向。在桌面操作系统领域，微软公司的Windows系列产品以其友好的图形用户界面，强大的开发环境和极其丰富的应用程序支持占据了统治地位。一些欧美的软件公司如Widcomm、ESI、DGAnswer以及日本的Sony公司等都在开发基于Windows的蓝牙软件；另外一些大公司，如Alcatel、

Philips、National Semicondector、Fujitsu、CSR 等虽然并不自己从事开发工作，但都在积极寻找合作伙伴，以便将第三方软件 OEM 在自己的产品中。在这种情况下，开发基于 Windows 的蓝牙应用系统，对于加快蓝牙技术的推广，抢占国际市场有着重要战略意义。

国际认证与测试公司（IVT）于 2000 年底启动蓝牙桌面系统开发项目，笔者在课题期间作为主要研发人员参加其中。该项目在国内首次提出了蓝牙通信系统桌面集成解决方案并将其产品化。作为公司主打产品，Bluelet Software Suite 的评估版已经向一些主要的分销商发布，反馈结果表明该软件以其强大易用和稳定的性能在国际同类产品中占据一席之地。

1.2 论文主要工作

笔者在论文期间认真研究了蓝牙协议，深入分析了串口仿真应用及其之上的各项应用；结合软件工程的思想，利用面向对象的技术和相关工具，给出了在 Windows 环境下蓝牙通信系统集成解决方案（以下简称**集成系统**）。主要工作包括：

1）研究了软件工程中面向对象的分析与设计思想和利用统一建模语言 UML 进行软件建模的方法。

2）在深入研究蓝牙协议和 profile 规范的基础上，以命令行方式分别实现了基于仿真串口的三个应用：拨号网络，传真和局域网访问。

3）利用面向对象的思想和 UML 语言对集成系统进行了需求分析，并依据分析结果构建了集成系统软件模型。在模型设计中，我们将集成系统从顶层分解为 UI(User Interface)与 BOL(Business Object Layer)两个包，重点完成了 BOL 包逐层细化的设计过程。

4）完成了集成系统模型中 BOL 包大部分和 UI 包部分模块的实现工作，主要包括：BOL 中 Profile Server 包、Profile Client 包、Bluetooth Device 包和 Virtual Device 包以及 UI 中设备

管理、传输层配置和 PNP 硬件检测模块。

5）设计并实现了包括安装程序、手工卸载程序以及虚拟设备管理模块在内的集成系统支撑模块。

在上述工作中主要包含了笔者的两方面创新和贡献：

首先，将软件工程的思想和面向对象的技术贯穿了集成系统从分析、设计到编码实现的全过程。通过工程性和规范性开发流程实现软件模块间高内聚、低耦合、可扩展与重用的内部特性，从而确保软件产品的外部质量。

其次，在基于仿真串口的三个应用实现中，在分析三者异同的基础上提出了模块化和结构化的集成策略。从而降低代码冗余度，提高了系统的可维护性。

1.3 论文结构

第一章我们简要了介绍论文的工程背景和主要工作。

第二章按照自下而上的顺序系统介绍蓝牙技术及其应用，为后续工作奠定技术基础。主要分硬件工作原理，软件协议栈和蓝牙应用框架等三个部分进行阐述。

在第三章笔者从 profile 规范出发，分别给出了基于仿真串口的三个蓝牙应用：拨号网络，传真和局域网访问的应用模式与实现结构；并在总结三者异同的基础上提出了集成策略。

第四章根据软件工程的思想，利用面向对象的技术和 UML 语言对集成系统进行了需求分析与系统设计。我们将整个系统在业务对象层（BOL）分解为五个包：Profile Server、Profile Client、Virtual Device、Bluetooth Device 和 Device and Service Discovery，并将前四个包细化到类图层次。

第五章讨论了系统实现中的若干问题。主要包括模型实现、支撑模块的实现以及两个技术难点。

第六章对全文进行总结。

第二章 蓝牙技术及其应用框架

2.1 蓝牙通信系统

如图 2.1 所示，整个蓝牙协议体系结构可分为底层硬件模块、中层协议栈（软件模块）和高层应用三大部分[10]。链路管理层（LM）、基带层（BB）和射频层（RF）属于蓝牙的硬件模块。RF 层通过 2.4GHz 无需授权的 ISM 频段的微波，实现数据位流的过滤和传输，它主要定义了蓝牙收发器在此频带正常工作所满足的要求。BB 层负责跳频和蓝牙数据及信息帧的传输。LM 层负责连接的建立和拆除以及链路的安全和控制。它们为上层软件模块提供了不同的访问入口。

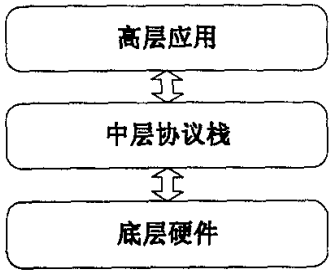

图 2.1 蓝牙通信系统

中层协议栈包括逻辑链路控制和适配协议（L2CAP：Logical Link Control And Adaptation Protocol）、服务发现协议（SDP：Service Discovery Protocol）、串口仿真协议（RFCOMM）和电话通信协议（TCS：Telephony Control Protocol）。L2CAP 完成数据拆装、服务质量控制和协议复用等功能，是其他上层协议实现的基础，因此也是蓝牙协议栈的核心成分。SDP 为上层应用程序提供一种机制来发现网络中可用的服务及其特性。RFCOMM 依据 ETSI 标准 TS07.10 在 L2CAP 上仿真 9 针 RS232 串口的功能。TCS 提供蓝牙设备间话音和数据的呼叫控制信令。

中层协议栈和底层硬件模块之间的消息和数据通过蓝牙主机控制器接口（HCI）的传递。也就是说，HCI 是蓝牙协议中软硬件之间的接口，它提供了一个调用下层 BB、LM、状态和控制寄存器等硬件的统一

命令接口。HCI 协议以上的协议软件实体运行在主机上，而 HCI 以下的功能由蓝牙设备来完成，二者之间通过一个对两端透明的传输层进行交互。

在蓝牙协议栈的最上部是高层应用（Applications），它对应于各种应用模型（profile），是 profile 的一部分。下面分别介绍蓝牙硬件模块个部件。蓝牙各层协议和应用模型将在下两节详细说明。

2.2 硬件模块工作原理

蓝牙硬件模块中包含有链路管理层（LM）、基带层（BB）和射频层（RF）[9]。

2.2.1 射频

蓝牙规定的天线功率以 0dBm（1mW）为基准，最大可达到 20dBm（100mW），其工作频率符合大多数国家（如美国、欧洲、日本等）的 ISM 频段标准，之所以选取此频段是为了能达到在全球均能运作的目标，即系统所需之频带必须是全球各地均能很容易取得，且此频带必须是未受法规限定及公开给无线电使用的，唯一符合此项要求的便是 2.4GHz--称为工业、科学、医疗（ISM）的频带。ISM 频带是对所有无线电系统都开放的频带，因此使用其中的任一频段都会遇到不可预测的干扰。例如某些家电、无绳电话、汽车房开门器、微波炉等，都可能是干扰源。为此，蓝牙特别设计了快速确认和跳频方案以确保链路稳定。蓝牙通过跳频方式将能量扩散到起始于 2.402GHz，终止于 2.480GHz 的 ISM 频段中，并将其划分为 79 个跳频信道，每个信道 1MHz。当前，蓝牙 SIG 正试图在全世界的范围内协调这 79 个信道，并已促使日本、西班牙等国政府调整了相应的限制政策。蓝牙的通信半径通常为 10cm--10m，但是如果增加发射功率，可以将半径扩展到 100m 外。

2.2.2 基带

如前所述，蓝牙在 2.4GHz 的 ISM 频段的 79 个信道里以跳频方式工作。当两个蓝牙设备成功建链后，一个 piconet 便形成了。两者之间

的通信通过无线电波在这 79 个信道中随机跳转而完成。蓝牙给每个 piconet 提供特定的跳转模式，因此它允许大量的 piconet 同时存在。

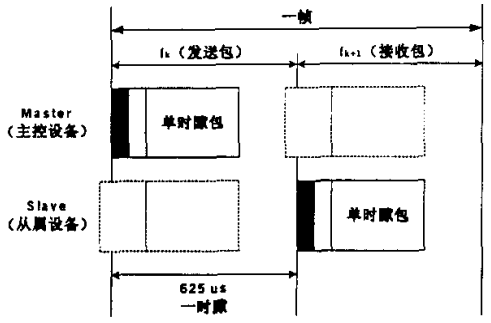

图 2.2　单时隙帧

蓝牙既支持电路交换也支持分组交换。蓝牙基带帧的时隙对同步分组（对应于电路交换）是保留的，每个分组在不同的跳频中发射，一个分组通常占用 1 个时隙，最多能扩展到 5 个时隙。蓝牙支持最大可达 3 个同步语音信道，同时也支持非同步数据信道，或者一个信道同时支持同步语音和非同步数据。

蓝牙采用时分双工（TDD：Time Division Duplex）方案来实现全双工传输，因此蓝牙的一个基带帧（Frame）包括两个包（Packet），首先是发送包，然后是接收包。每个包可由 1 个、3 个或 5 个时隙组成，每个时隙 625us。一个典型的单时隙帧（如图 2.2 所示）每秒跳 1600 次。多时隙帧由于节省了头信息开销而具有更高的数据速率。比如，单时隙帧的单向速率最大为 172kbps，而一个 5/1（5 表示一帧内的发送包的时隙数，1 表示接收包的时隙数）的多时隙帧则支持发率 721 kbps 和收率 57.6 kbps（对 Master 而言）。图 2.3 表明了一个 3/1 多时隙帧的示意图。

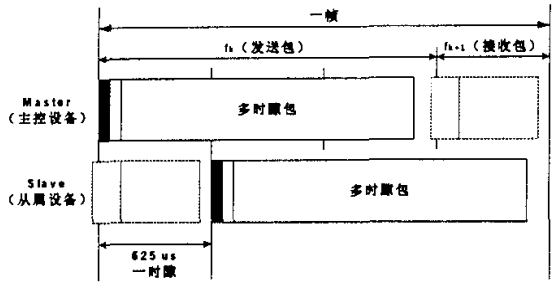

图 2.3　3/1 多时隙帧

2.2.3 链路管理与控制

蓝牙设备互连形成 piconet，每个 piconet 包括一个且只有一个主控设备（Master）和最多 7 个从属设备（Slave）。任何一个蓝牙设备既可以成为主控设备又可成为从属设备。角色的分配是在 piconet 形成时临时确定的。一般而言，发出连接指令的设备将成为主控设备，但是蓝牙系统的“主/从转换”功能可使角色改变。

为了形成 piconet，蓝牙设备需要知道两个参数，即它希望连接到的设备的跳转模式及其相应相位。每个蓝牙设备都有一个唯一的用于标识自身跳转模式的全球标识符（Global ID）。在形成 piconet 时，主控设备先和其他设备分享自己的 ID 号，再向那些设备提供自己的时钟偏移信息。这些信息由所谓的跳频包（FHS）发送。

通常未连接进 piconet 的设备处于旁观（Standby）模式。此时这些设备监听其他设备的搜询（Inquiry）消息或者构建 piconet 的请求（Page）。当某个设备发出查询（Inquiry）命令时，接收设备将用它们的 FHS 包发送自己的 ID 号和时钟偏移给询问者，以便使其形成一个完整的覆盖范围内的设备情况表。

为了形成 piconet，主控蓝牙设备会用所需设备的 ID 号寻呼（Page）这个设备（此 ID 号是在先前的 Inquiry 中得到的）。被呼设备将用自己的 ID 号回应，然后主控设备会再发一个 FHS 包（包括主控设备的 ID 号和时钟偏移）给被呼设备，随后被呼设备便加入了主控设备的 piconet 中。

一旦某个设备加入 piconet 中，它就被分配给一个 3 比特的主动成员地址（AMA：Active Member Address），其他成员可以用其访问该设备。一旦 piconet 内有 8 个活动从属设备，主控设备必须把一个从属设备强制成停等（Park）模式。在 Park 模式中，此设备仍然存在于 piconet 中，但是它释放了 AMA 地址而得到一个 8 比特的被动成员地址（PMA：Passive Member Address）。AMA 和 PMA 的结合允许超过 256 个设备同时存在于一个 piconet 中，但是只有 8 个具有 AMA 地址的设备（包括主控设备）才能进行通信。

停等（Park）的设备以一定间隔聆听外界发给它们的指令。这就要求主控设备有能力给所有的从属设备（不论是停等的还是活动的）广播信息。处于 Standby 状态的设备也监听其它设备发出的 Inquiry 或 Page 指令。每隔 1.25s 它们就做一次这样的扫描。

在查询（Inquiry）过程中，主控设备使用的是特别预留的全球统一的 Inquiry 事件 ID 标识号，并采用全球唯一的包含 32 个信道的信道序列发送此指令（32 个回复信道也是预留的）。进行 Inquiry 扫描的设备每隔 1.25s 就在这 32 个信道中的某个信道上停留 10ms，然后就跳转到序列中的下一个信道继续监听，直到该设备的 Inquiry 扫描功能被禁止（可能不止一个设备发出 Inquiry 指令，因此要连续监听）。在主询端，32 个 Inquiry 信道被分成 2 个频组，每组 16 个信道。主询设备先在第 1 频组上发布 16 条相同的 Inquiry 指令，随即每隔 1.25s 在反向回复信道上监听回音。如果被询设备扫描的信道正好和主询设备发布指令的信道重合，被询设备的监测相关器就会起较明显的反应，而后被询设备就会用 FHS 包发送自己的 ID 号和时钟偏移。在下一个 1.25s 内主询设备用第 2 组频率重新发布 Inquiry 指令，如此反复，直到主询设备的覆盖范围内的所有设备都发回 FHS 包。

寻呼（Page）过程也采用相似的信道序列。每个设备依据其 ID 号都有唯一的包含 32 个寻呼频率的信道序列和包含 32 个回复频率的信道序列。处于 Standby 状态的设备每隔 1.25s 在其特有的寻呼信道序列中的某个信道停留 10ms 以监听来自主呼方的寻呼 ID 信息，若此 ID 号不是自己的，该设备就跳转到序列中的下一个寻呼信道继续监听。在主呼

端，欲呼叫设备的32个寻呼信道也被分成2个频组，每组16个信道。主呼设备先根据它最近知道的被呼设备的时钟偏移作出被呼设备位置的估计，然后调整两个频组的频率，随即主呼设备先用第1组估计的频率持续地呼叫1.25s。如果位置估计是错误的（即主呼设备未收到回音），主呼设备将在下一个1.25s内使用第2频组。小的时钟偏移会使呼叫过程很快完成，而大的时钟偏移却会使该过程延长到最大2.5s（两个频组总共呼叫的时间）。一般而言，此过程的平均时延是0.64s。一旦一个设备通过Inquiry被发现并且通过Page加入到piconet中，piconet就形成了。

在活动（Active）状态中，每个蓝牙设备都被分配一个AMA地址，它指引数据传到不同的设备中（主控设备的地址总是默认为0）。为了在很低的功率状态下也能使蓝牙设备处于连接状态，蓝牙规定了三种节能状态，即停等（Park）状态、保持（Hold）状态和呼吸（Sniff）状态。在Sniff状态中，从属设备降低了从piconet“收听”消息的速率，一会儿醒一会儿睡，宛如呼吸一样；而在Hold状态中，设备停止传送数据，但一旦激活，数据传递就立即重新开始。在Park状态中，设备被赋予PMA地址，并以一定间隔监听主控设备是否(1)询问本设备想成为活动设备；(2)询问任何停等的设备想成为活动设备；(3)广播消息。如果我们把这几种工作模式按照节能效率以升序排一下队，那么依次是：呼吸模式、保持模式和停等模式。图2.4给出了蓝牙设备的状态转移情况。

在活动状态下，蓝牙设备能够支持两种链路类型，即面向连接的同步链路（SCO：Synchronous Connection Oriented）和面向无连接的异步链路（ACL：Asynchronous Connectionless）。每种链路支持16种不同的分组类型，其中4种是控制分组。

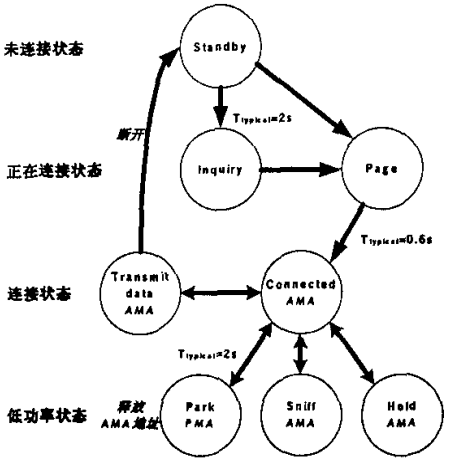

图 2.4　蓝牙设备状态转移图

SCO 数据包既可以传送话音，也可以传送数据，但在传送数据时，只用于重发被损坏的那部分数据。SCO 帧内的收发包结构必须是对称的，即必须同时包含 1 个、2 个或 3 个时隙。SCO 数据包在保留的时隙内发送，一旦 SCO 链路建立，主从设备就直接发送 SCO 分组，无需轮询（Poll）。为了建立 SCO 连接，必须先建立 ACL 链路以传送控制信息。

ACL 支持对称和非对称两种帧格式。ACL 的包（不论是收还是发）必须包含奇数个时隙，以使整个帧的时隙数为偶数（如 1/1、1/3、或 1/5 等）。主控设备负责控制 ACL 链路的带宽，并决定 piconet 中的每个从属设备可以占用多少带宽及连接的对称性。从属设备只有被选中时才能传送数据，即从属设备在发射数据前必须接受轮询。ACL 链路也支持接收主控设备发给 piconet 中所有从属设备的广播消息。

蓝牙采用三种纠错方案：1/3 前向纠错（FEC：Forward Error Correction）、2/3 前向纠错和自动请求重发（ARQ：Automatic Repeat Request）。前向纠错的目的是减少重发的可能性，但同时也增加了额外开销。然而在一个合理的无错误率环境中，多余的头标会减少输出，故分组定义的本身也保持灵活的方式，因此，在软件中可定义是否采用 FEC。一般而言，在信道的噪声干扰比较大时蓝牙系统会使用前向纠错方案以保证通信质量：对于 SCO 链路，使用 1/3 前向纠错；对于 ACL

链路，使用 2/3 前向纠错。

在无编号的自动请求重发方案中，一个时隙传送的数据必须在下一个时隙得到收到的确认。只有数据在收端通过了报头错误检测和循环冗余校验（CRC）后认为无错才向发端发回确认消息，否则返回一个错误消息。

目前蓝芽传送语音数据采用连续可变斜率增量调制（CVSD: Continuous Variable Slope Delta Modulation）编码。这种编码可以保证很高的信噪比，它擅长处理丢失的和被损坏的语音采样，即使比特错误率达到 4%，CVSD 编码的语音还是可听的。

2.2.4 安全

蓝牙系统的移动性和开放性使得安全问题极其重要。虽然蓝牙系统所采用的跳频技术就已经提供了一定的安全保障，但是蓝牙系统仍然需要链路层和应用层的安全管理。在链路层中，蓝牙系统提供了认证、加密和密钥管理等功能。每个用户都有一个个人标识码（PIN），它会被译成 128bit 的链路密钥（Link Key）来进行单双向认证。一旦认证完毕，链路就会以不同长度的密码（Encryption Key）来加密（此密码以 8bit 为单位增减，最大 128bit）链路层安全机制提供了大量的认证方案和一个灵活的加密方案（即允许协商密码长度）。当来自不同国家的设备互相通信时，这种机制是极其重要的，因为某些国家会指定最大密码长度。蓝牙系统会选取 piconet 网中各个设备的最小的最大允许密码长度。例如，美国允许 128bit 的密码长度，而西班牙仅允许 48bit，这样当两国的设备互通时，将选择 48bit 来加密。 蓝牙系统也支持高层协议栈的不同应用体内的特殊的安全机制。比如两台计算机在进行商业卡信息交流时，一台计算机就只能访问另一台计算机的该项业务，而无权访问其它业务。蓝牙安全机制依赖 PIN 码在设备间建立信任关系，一旦这种关系建立起来了，这些 PIN 码就可以存储在设备中以便将来更改。

2.3 蓝牙协议栈

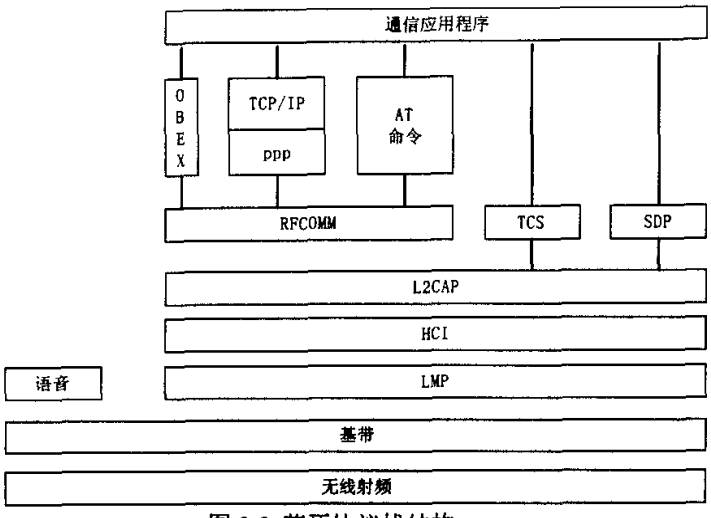

图 2.5 蓝牙协议栈结构

2.3.1 HCI 协议

HCI 协议分为主机端驱动 HCI Driver 和硬件模块上的固件 HCI Firmware，二者之间由一个传输层提供数据的透明传输[1]。目前定义了三种传输层：UART、USB 和 PCMCIA。

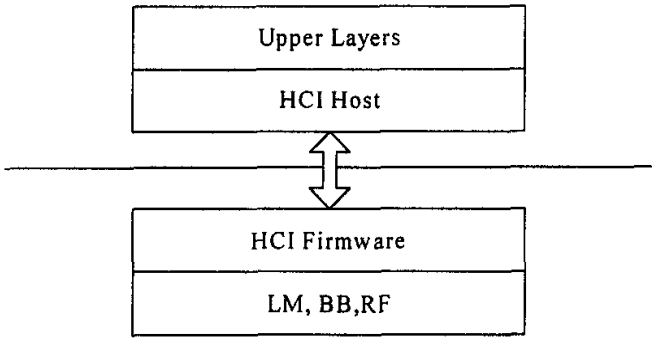

图 2.6 蓝牙系统低层体系结构

如图 2.6 所示，下层 HCI 协议通过传输层从主机端收到命令时，根据命令内容执行一定的动作，发出特定的 LM 层或者 BB 层命令；同样下层 HCI 协议实体从 LM 层或者 BB 层接收到一定的事件时，也应该转换

为正确的HCI事件通过传输层送到主机端。因此对于主机而言，蓝牙的硬件模块就相当于一个通信平台，所有的用户和系统命令通过主机端HCI发出，所有用户和系统事件也都从HCI接收。

2.3.2 L2CAP协议

逻辑链路控制和适应层协议（Logical Link Control and Adaptation Layer Protocol简称L2CAP），是一个为高层传输层和应用层协议屏蔽基带协议的适配协议[1]。它主要完成协议复用、数据拆装、QoS和组管理。

L2CAP为高层提供数据服务，允许高层和应用层协议收发大小为64kB字节的L2CAP数据包。L2CAP只支持基带面向无连接的异步传输（ACL），不支持面向连接的同步传输（SCO），SCO链路主要用预留的带宽进行实时语音传输。

在基带的异步无连接（ACL）物理链路上传输的包头结构如图2.7所示。

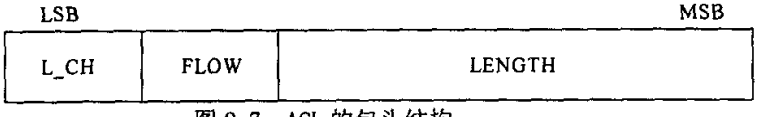

图2.7 ACL的包头结构

基带根据2比特的L_CH标志来区分L2CAP或LMP的包。LMP包的L_CH为11，而L2CAP包的L_CH为01（起始包）、10（后续包）。1比特的FLOW标志由链路控制器（LC）管理。该标志位缺省值为1，表示正常传输；如果该标志为0，则意味着不再有L2CAP包会在该ACL链路上传输。L2CAP主要向上层提供以下功能。

1. 协议复用（Protocol Multiplexing）。多个高层协议共享一个公共的物理连接，从逻辑上看每个协议都有自己的通道，但由于基带协议不能识别任何高层协议，所以L2CAP必须支持上层协议复用，它应能区别诸如SDP、RFCOMM、TCS等高层协议，并正确地收发相应的包。

2. 分段和重组（Segment and Reassembly）。与其它有线的物理

连接相比，蓝牙的基带包的大小有一定的限制。最大的基带包只能传输341字节的信息，而这限制了高层协议有效地利用带宽以传输更大的包。L2CAP允许高层和应用层协议收发大小为64kB字节的L2CAP数据包，所以，L2CAP必须在传往基带前将其包进行分段，以适应基带的要求。同样的，在接收方，L2CAP必须能将多个基带包重组为一个L2CAP包传往高层。

3. 服务质量（QoS, Quality of Service）。在L2CAP的建立连接过程中允许改变两台设备间的服务质量。每个L2CAP实体应确保服务质量的实现并管理所使用的资源。

4. 组管理（Group Manager）。很多协议支持组地址的概念，蓝牙的基带协议支持微微网，即一组设备使用同一时钟同步跳频。L2CAP的组提取功能可以有效地将协议的组映射为基带的微微网，以避免高层协议为了有效地管理组而必须与基带协议直接联系。

值得注意的是，L2CAP只是利用基带的机制来提供可靠的信道，其本身不提供任何重传和校验功能，它能正确地传送包也是建立在基带能有序地传送同一包的不同分组基础上的。

2.3.3 电缆替代协议(RFCOMM)

RFCOMM是基于ETSI 07.10规范的串行线仿真协议。“电缆替代”协议在蓝芽基带协议上仿真RS232控制和数据信号，为使用串行线传送机制的上层协议（如OBEX）提供服务[1]。Bluetooth特别兴趣小组提出RFCOMM的目的在于以下几点：提供对现有使用串行线接口的应用软件的支持；利用已有的GSM 07.10标准；支持Bluetooth设备之间点对点的通信。

RFCOMM完成了对RS232串口的仿真，这样就可以尽可能利用现有的各种高层应用程序，保证Bluetooth技术与现有技术的融合以及各种应用之间的互通性，充分利用兼容Bluetooth技术规范的软硬件体系。其中最常用的是基于串行线传送机制的高层协议，如：OBEX、PPP和AT命令集。其中PPP完成点对点的连接；OBEX（对象交换协议）是由红外数据协会（IrDA）制定的会话层协议，它采用简单的和自发的方式交换

目标数据，它是一种类似于HTTP（Hypertext Transfer Protocol）的协议，假设传输层是可靠的，采用客户机一服务器模式，独立于传输机制和传输应用程序接口；AT命令集是专门为调制解调器设计的接口，用来提供拨号上网和收发传真的功能。

根据实现方式不同，存在两种不同的BT设备。类型1的BT设备本身具有完整的Bluetooth（BT）硬件和软件，能独立完成Bluetooth的功能，实现通信设备之间的短距离无线连接。类型2的BT设备仅仅起到类似于Modem的功能，负责将普通设备所要传输的信息与Bluetooth格式的码流的相互转换。当采用类型2的BT设备时，数据传输的瓶颈在于实际的串口之间的有线连接，即受到标准串口最高速率921600bit/s的限制；而采用类型1的BT设备时，数据传输的速率则不受到串口速率的限制。

2.3.4 服务发现协议（SDP）

发现服务在蓝芽技术框架中起到至关重要的作用，它是所有用户应用的基础。使用SDP，可以查询到设备信息和服务类型，从而在蓝芽设备间建立相应的连接[1]。

2.3.5 电话控制协议(TCS)

TCS协议基于Q.931协议，采用了Q.931中呼叫的对等部分并针对蓝牙网络的实际特点有所拓展[1]。TCS协议根据功能可以划分为三个部分： CC（Call Control）、GM（Group management）和CL（Connectionless TCS）。其中，CC就是从Q.931化来的关于呼叫控制的部分，用于控制一个蓝牙语音或数据呼叫的建立、释放和管理；GM是根据基于TCS的应用的特点从而对一组蓝牙设备进行管理的部分，当前主要用于加快两个终端之间呼叫的建链过程；CL用于无连接的TCS数据的交互，这里的无连接是针对TCS这层而言的，是指没有呼叫相对应。

TCS设备之间呼叫的建立可以有两种方式：一种是点对点呼叫，另一种是点对多点呼叫。点对点呼叫适用于被呼叫方已知，信令承载信道

使用面向连接的 L2CAP 信道；点对多点适用于被呼叫方有待确定，信令承载信道只能采用面向无连接 L2CAP 信道。TCS 呼叫的语音承载信道可以是 L2CAP 数据信道也可以是面向连接的 SCO 信道。

2.4 蓝牙应用框架

为了清晰描述蓝牙的多种使用方式，有效实现互连互通，SIG 定义了一系列的应用（profile）[2]，见图。每个应用都从蓝牙技术规范（Specification）中选取了一组消息和过程，规定了协议栈各层必须实现的功能子集。可以形象的说，每个 profile 都是协议栈的一个纵向切片。

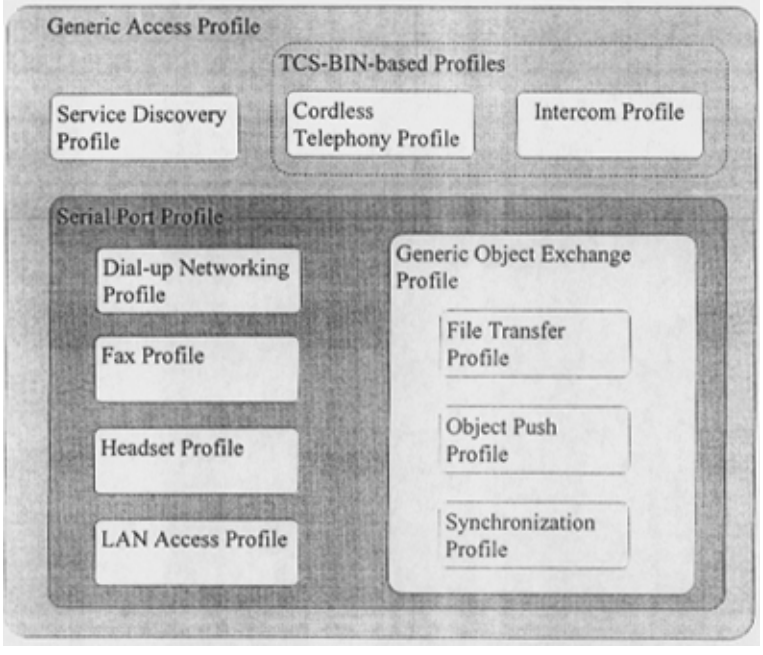

图 2.8　Bluetooth 应用框架

2.4.1 通用访问应用（GAP）和服务发现应用（SDP）

通用访问应用（GAP）和服务发现应用（SDP）通用于所有其它的应用。GAP 定义了 Bluetooth 设备的工作模式、安全模式以及设备发现

和建立连接的通用过程。Bluetooth 设备必须遵循 GAP 的定义，以保证与其它设备最基本的互连性。SDP 定义了 Bluetooth 网络环境下发现服务的基本流程，并提出服务发现应用的三种可能模式。SDP 还定义了关于服务发现的高层接口。

SDP 与下文中所述的应用都依赖于 GAP 来完成设备的发现和连接的建立，同时下文中所述的应用要依赖 SDP 来完成服务的发现。设备和服务的发现是访问任何服务的前提。

2.4.2 基于 SPP 的应用

串口仿真应用 SPP（Serial Port Profile）建立在电缆替代协议 RFCOMM 之上。SPP 定义了如何在 Bluetooth 设备上实现虚拟串行端口，在两台蓝牙设备间建立基于 RS-232 规范的仿真串行线路。

在 SPP 的基础上，针对不同的应用模式又分别定义了局域网接入（LAP）、拨号网络（DUN）、传真（Fax）、耳机（Headset）等应用。通用对象交换应用（GOEP）是在 SPP 提供的串行线路的基础上，定义了实现对象交换的一组协议和过程，主要包括连接的建立和断开、向服务器“推”（Push）对象和从服务器“拉”（Pull）对象四种操作。文件传输（FTP）、对象推送（OPP）和同步（Synchronization）应用都是基于 GOEP 的。

串口仿真应用的意义在于：串行接口规范 RS-232 在通信和计算机领域中具有广泛应用，现有基于串口的通信软件极其丰富，如 Windows 下的电话拨号程序、超级终端、拨号网络客户和服务器程序，以及传真软件等。实现蓝牙仿真串口后，这些现成的应用软件即可直接移植到蓝牙链路上(称为 legacy porting)，达成电缆替代的初衷。

从实现角度讲，具备丰富串行通信软硬件资源的桌面系统非常适于作为基于 SPP 的应用的宿主机。本文所实现的蓝牙桌面通信系统主要集成了基于 SPP 的局域网接入（LAP）、拨号网络（DUN）、传真（Fax）和文件传输（FTP）等应用。

2.4.3 基于 TCS 的应用

目前，基于 TCS 的应用主要包括无绳电话(CTP)和互连(Intercom)两种应用。CTP 使移动终端可以接入公众电话交换网（PSTN）或综合业务数字网（ISDN），而 Intercom 使移动终端之间可以通信。这样，一部同时支持CTP和Intercom应用的移动电话可兼作无绳电话和对讲机，从而具有三合一的功能。

从实现角度讲，基于 TCS 的应用主要以嵌入式系统作为应用环境。

第三章 基于仿真串口诸应用的研究与实现

3.1 应用之一：拨号网络

3.1.1 应用模式

拨号网络应用(Dial-up Networking Profile)是一种名为“因特网网桥”的使用模型，允许计算机使用蓝牙无线 Modem 或者带有蓝牙功能的移动电话无线地连接到因特网上[2]。

图 3.1 拨号网络应用模式

拨号网络应用定义了蓝牙设备两种典型的角色：网关 **Gateway (GW)**，提供访问公众网络(PSTN)的接入；数据终端 **Data Terminal (DT)**，使用 GW 提供的拨号服务。在 DT 通过蓝牙连接到 GW 以后，我们在 RFCOMM 上用 PPP 访问因特网的。在 Window 环境下，GW 通常配置为 PC+蓝牙硬件+Modem，DT 为 PC+蓝牙硬件。

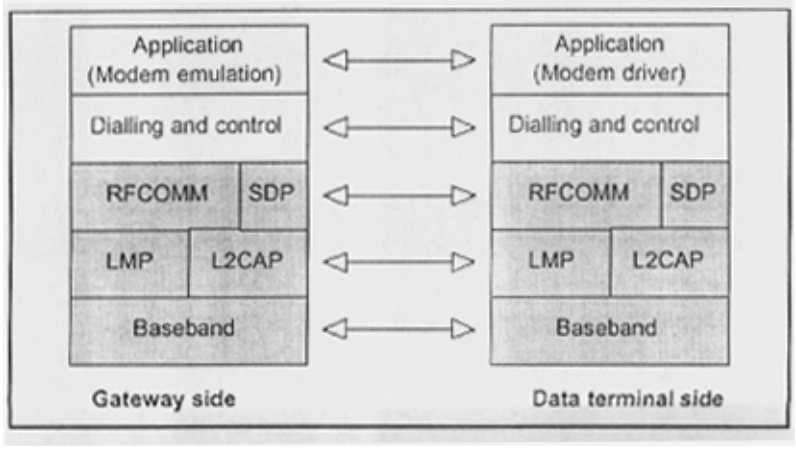

图 3.2 拨号网络应用协议模型

3.1.2 系统结构

DT 和 GW 对协议和程序的要求并不是完全一样的。我们根据它们的功能定义了不同的系统结构，如图 3.3 所示。

GW 应用程序负责向 SDP 数据库注册服务的一些信息，包括服务类型和相应的服务信道号，然后在注册的服务信道号上等待拨入的呼叫。

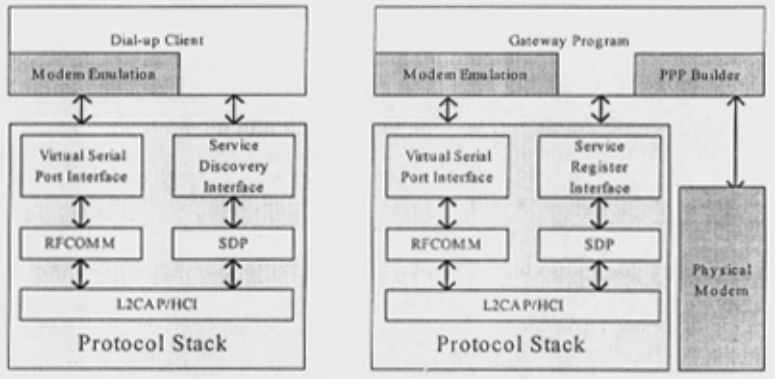

图 3.3 拨号网络应用的系统结构

DT 应用程序负责调用 SDP 输出的接口函数查找附近设备上存在的 GW 服务。如果找到所要求的服务，GW 的地址和服务信道号就会被返回。这样，DT 就可以用 GW 地址和服务信道号与远端的 GW 建立蓝牙连接，然后再通过 GW 与 ISP（Internet Service Provider）建立起 PPP 连接。

尽管 DT 和 GW 在操作模式方面存在许多区别，它们都使用基于蓝牙协议栈的共用模块。这个公用模块输出两类接口：虚拟串口接口和服务发现/注册接口。

- **数据终端（DT）**

在 DT 开始拨入 GW 之前，DT 必须先用网络邻居浏览程序查找周围设备上存在的拨号网络应用的服务。然后再选择一个服务，将此服务的属性通过调用由虚拟串口接口的输出接口加入本地设备上的某个虚拟串口（此串口号由蓝牙系统分配）的属性表中并开启该虚拟串口，接着，

再启动拨号客户程序打开该虚拟串口并连接 GW。拨号客户程序将使用 AT 命令控制 GW 端 Modem 向远端 ISP 的拨号服务器拨号；拨号服务器应答后，拨号客户将启动 PPP 以在 DT 和远端的 ISP 之间创建一条 PPP 连接。以下是 DT 的系统结构，如图 3.4 所示。

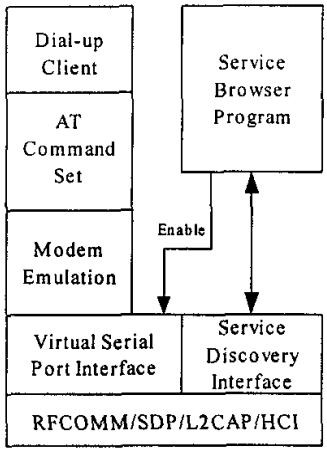

图 3.4 DT 的系统结构

- **拨号程序**

拨号程序是利用 Window 操作系统提供的拨号命令 RasDial()实现的。使用中用户首先设置所需的拨号参数，如连接名称、电话号码、用户名和密码、所用串口设备等；之后使用这些参数进行拨号。对于拨号程序来讲，它并不区分所用串口的是系统的真实的 UART 设备还是蓝牙虚拟串口，也不知道所用的 Modem 是连接在本机还是远端（GW）。具体实现参见附录 A4.2。

蓝牙规范定义了 DUN 专用的 AT 命令，拨号程序通过 RFCOMM 使用这些 AT 命令来控制远端的 MODEM。在 Win98 和 Win2k 中，拨号程序使用 Modem 的 inf 配置文件指定 AT 命令系列。在安装 MODEM 时，Win98 将 Modem 的 inf 配置文件中的各种属性条目加入到系统注册表中，这样当使用该 Modem 时，Windows 系统将根据需要调用这些 AT 命令。

- **服务浏览程序**

服务浏览程序是基于蓝牙协议栈的 SDP（服务发现协议）部分，负责查找周围的蓝牙设备以及浏览指定设备上的服务。服务浏览程序提供了一个良好的人机界面，让用户方便提交 SDP 的查找请求，并显示查找结果。

➢ 网关（GW）

GW 的系统结构如图 3.5 所示，包含两个功能模块：服务注册程序和转发程序。

在 GW 接受服务查找之前，必须将服务通过服务注册接口注册到本地 SDP 数据库里。将该服务的属性通过调用虚拟串口接口加入本地设备上的某个虚拟串口（此串口号由蓝牙系统分配）的属性表中并使能该虚拟串口。接着，再启动转发程序打开该虚拟串口等待远端 DT 的连接，注意，此时，GW 并不初始化 Modem，而是由 DT 发送 AT 命令到 GW，GW 再将 AT 命令转发给 Modem，命令 Modem 与远端的 ISP 建立连接。如果 GW 想停止提供服务，可以注销已注册的服务。

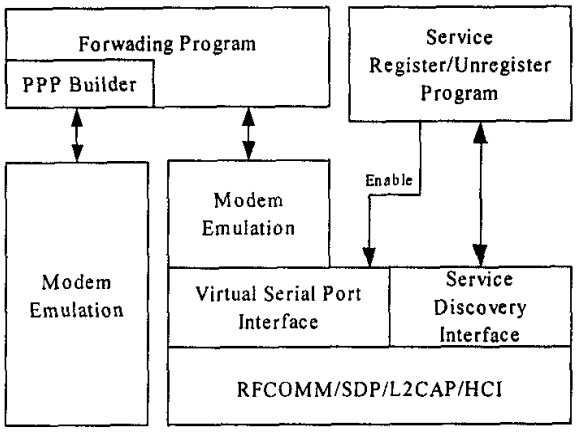

图 3.5 GW 的系统结构

转发程序主要在虚拟串口和插有物理 Modem 的真实串口间转发两类数据：1、来自 DT 的 AT 命令和 Modem 的响应；2、PPP 数据包。GW 的数据流如图 3.6 所示。

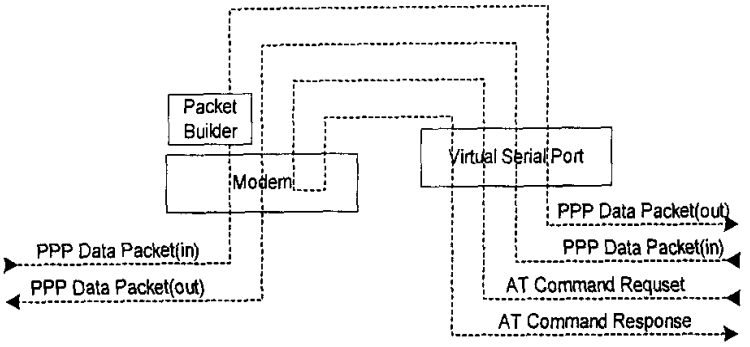

图 3.6 GW 的数据流

在转发程序中，我们创建了两个线程分别处理两个数据流：来自虚拟串口的数据流以及来自 Modem 的数据流。在用于处理来自虚拟串口数据流的线程中，将来自虚拟串口的 AT 命令、Modem 的控制信令以及 PPP 数据直接发往 Modem，而在用于处理来自物理 Modem 数据流的线程中，将来自 Modem 的响应及 Modem 的状态变化报告通过 RFCOMM 报告给拨号程序，对于从 Modem 接收到的数据则由组包模块处理，然后再发给拨号程序。

在转发程序中，组包模块是很重要的。因为串口是面向字符流的的设备，而且 PPP 对不完整的 PPP 包有超时处理机制，所以必须将从 Modem 来的数据进行组包，等到组成一个完整的 PPP 数据包再发送给拨号程序，这样拨号程序端的 PPP 实体就可以处理一个完整的 PPP 数据包，也就可以将 PPP 包里的用户数据发送给拨号程序端的 PPP 实体。如果不这样做的法，拨号程序端的 PPP 实体将会收到不完整的 PPP 包，而在规定的超时时间里，如果 PPP 包的后续部分没有到来的话（因为经过蓝牙链路的传输，会增加一定延时，有可能超过 PPP 规定的超时时间），PPP 实体将会丢掉已收到的不完整的 PPP 包。这样，网关程序的数据传输效率就会降低。实际上，组包模块并不能减小蓝牙链路的时延，而是将一个 PPP 包内前后部分间的时延转化为 PPP 包间的时延，而根据协议，后者是大于前者的，从而减少超时包，提高传输效率。

系统组包程序最初由同事开发，组包算法复杂，代码长而效率低。

在实现中，笔者对其进行了改进，细节可参见第五章。

3.2 应用之二：传真

3.2.1 应用模式

传真应用(Fax Profile)规定了数据设备通过蓝牙链路收发传真所需的协议和规程[2]。

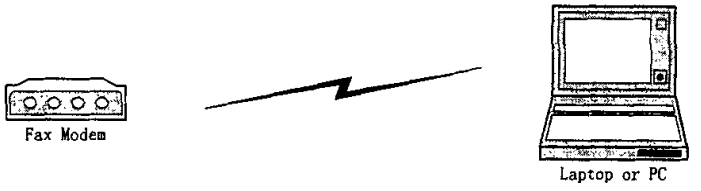

图 3.7 传真应用模式

类似于拨号网络，传真应用也定义了蓝牙设备两种典型的角色：网关 **Gateway (GW)**，提供通过公众网络(PSTN)收发传真的服务；数据终端 **Data Terminal (DT)**，使用 Gateway 提供的传真服务。在 Window 环境下，GW 通常配置为 PC+蓝牙硬件+Fax Modem，DT 为 PC+蓝牙硬件。

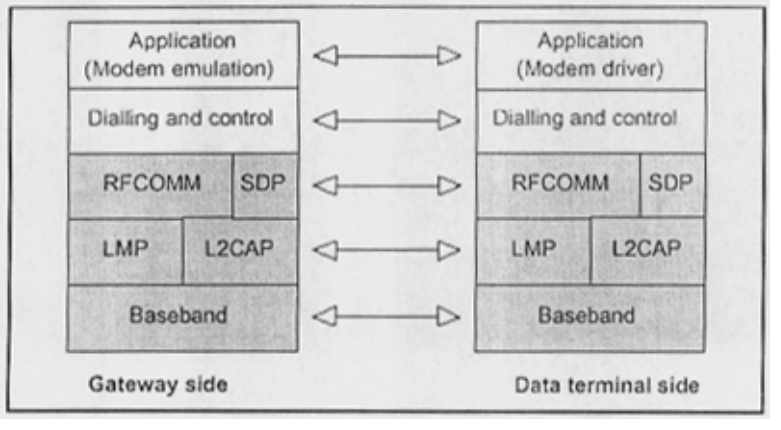

图 3.8 传真应用协议模型

3.2.2 系统结构

DT 和 GW 对协议和程序的要求并不是完全一样的。我们根据它们的功能定义了不同的系统结构，如图 3.9 所示。

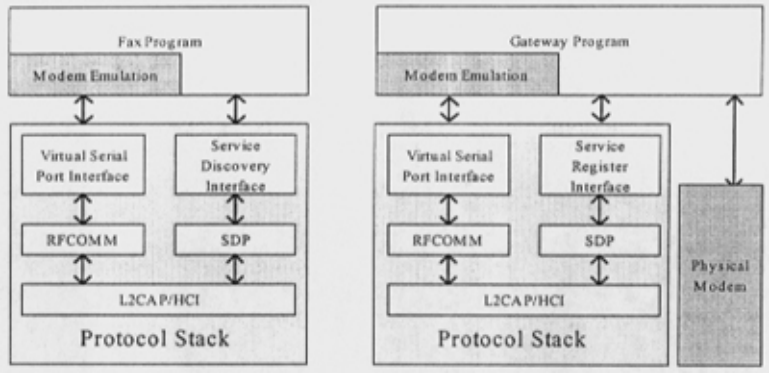

图 3.9 传真应用的系统结构

GW 应用程序负责向 SDP 数据库注册服务的一些信息，包括服务类型和相应的服务信道号，然后在注册的服务信道号上等待拨入的呼叫。DT 应用程序负责调用 SDP 输出的接口函数查找附近设备上存在的 GW 服务。之后 DT 就可以用 GW 地址和服务信道号与远端的 GW 建立蓝牙数据链路，然后 DT 就可以通过 GW 端 Fax Modem 收发传真。

- **数据终端（DT）**

在 DT 开始拨入 GW 之前，DT 必须先用网络邻居浏览程序查找周围设备上存在的拨号网络应用的服务。如果找到所要求的服务，GW 的地址和服务信道号就会被返回。然后再选择一个服务，将此服务的属性通过调用由虚拟串口接口的输出接口加入本地设备上的某个虚拟串口（此串口号由蓝牙系统分配）的属性表中并开启该虚拟串口，接着再启动传真程序，传真程序将打开该虚拟串口并连接 GW 端 Fax Modem，使用 AT 命令控制 GW 端 Fax Modem 拨叫远端传真机或等待远端传真机拨入。以下是 DT 的系统结构，如图 3.10 所示。

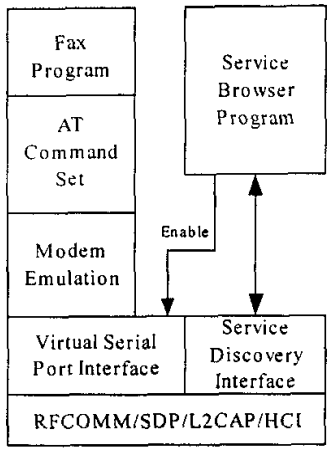

图 3.10 DT 的系统结构

- **传真程序**

传真程序可以使用 Window 操作系统提供的传真功能，也可以使用第三方传真软件。使用中用户只需设置对方电话号码、选择要传输的文件和所用串口设备，之后即可进行拨号。对于传真程序来讲，它并不区分所用串口的是系统的真实的 UART 设备还是蓝牙虚拟串口，也不知道所用的 Modem 是连接在本机还是远端（GW）。

蓝牙规范定义了 Fax 专用的 AT 命令，拨号程序通过 RFCOMM 使用这些 AT 命令来控制远端的 Fax MODEM。在 Win98 和 Win2k 中，拨号程序使用 Modem 的 inf 配置文件指定 AT 命令系列。在安装 MODEM 时，Win98 将 Modem 的 inf 配置文件中的各种属性条目加入到系统注册表中，这样当使用该 Modem 时，Windows 系统将根据需要调用这些 AT 命令。

- **服务浏览程序**

服务浏览程序是基于蓝牙的 SDP（服务发现协议）部分，负责查找周围的蓝牙设备以及浏览指定设备上的服务。服务浏览程序提供了一个良好的人机界面，让用户方便提交 SDP 的查找请求，并显示查找结果。

➢ 网关（GW）

GW 的系统结构如图 2.10 所示，包含两个功能模块：服务注册程序和转发程序。

在 GW 接受服务查找之前，必须将服务通过服务注册接口注册到本地 SDP 数据库里。将该服务的属性通过调用虚拟串口接口加入本地设备上的某个虚拟串口（此串口号由蓝牙系统分配）的属性表中并开启该虚拟串口。接着，再启动转发程序打开该虚拟串口等待远端 DT 的连接，注意，此时，GW 并不初始化 Modem，而是由 DT 发送 AT 命令到 GW，GW 再将 AT 命令转发给 Modem，命令 MODEM 与远端的 ISP 建立连接。如果 GW 想停止提供服务，可以注销已注册的服务。以下是 GW 的系统结构，如图 3.11 所示。

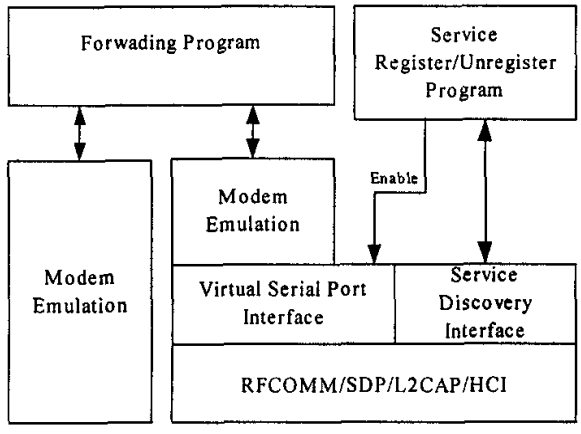

图 3.11 GW 的系统结构

转发程序主要在虚拟串口和插有 Fax Modem 的真实串口间转发三类数据：1、来自 DT 的 AT 命令和来自 Modem 的响应；2、传真信令帧；3、传真图像编码数据。实践中为简化程序，加快进度我们对 2、3 类数据与第一类一样直接在两种串口间进行转发而没有经过类似于 DUN GW 的组包程序，性能并不可靠，只能达到可以演示的程度。试验表明问题主要出在传真信令帧的转发延时过长，这一方面仍需要进一步工作。

3.3 应用之三：局域网访问

3.3.1 应用模式

局域网访问应用(LAN Access Profile)规定了蓝芽设备如何在RFCOMM上用PPP访问LAN的服务，以及显示了如何用同样的PPP机制由两台蓝芽设备组成IP网络[2]。

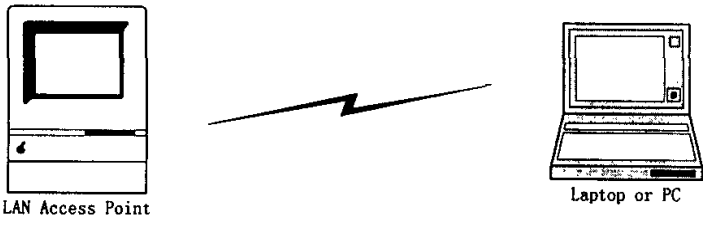

图3.12 局域网访问应用模式

局域网访问应用给蓝芽设备定义了两种角色：局域网访问点（LAN Access Point--**LAP**）和数据终端（**DT**）。需要明确一点：在不引起混淆的情况下，本文中有时把局域网访问应用也缩写成LAP（LAN Access Profile）。

LAP是提供访问局域网(如以太网、令牌环网等)服务的蓝芽设备。LAP提供PPP服务器的服务。PPP连接是建立在RFCOMM之上的。RFCOMM被用来传输PPP包，也可以用来对PPP数据流的流控。在Window环境下，LAP通常是一台LAP接有蓝牙设备的服务器或PC机。

DT是使用LAP服务的设备，典型的DT设备有笔记本电脑、台式机、PDA等。DT作为PPP的客户端访问LAP的服务。在LAP和DT之间建立一条基于RFCOMM的PPP链路。一旦DT连上LAP，DT就可以访问LAP甚至LAP所在的LAN上的服务。

局域网访问应用定义了三种使用方式。

- 一台LAP一次只能接受一台DT的请求。这种情况下，DT就好象通过拨号网络拨上局域网一样，可以访问局域网提供的所有服务。

- 一台 LAP 一次可以接受多台 DT 的请求。这种情况下，这些 DT 不仅可以访问局域网的服务，而且它们还可以通过 LAP 进行相互间的通信。

- 两台 PC 之间建立起一条虚拟串行链路。这种情况类似两台 PC 机之间的直接电缆连接。这时候，可以一台 PC 做为 LAP，另一台做为 DT。

有些 LAP 可能与一个内部局域网连接或使用 PSTN 来访问 Internet。这种连接到其他局域网或 Internet 的机制是与 LAP 的配置与实现相关的，DT 不会意识到这些行为，只是可能连接的时间会变长或通信延迟变大。图 2.12 显示了 LAN Access Profile 中用到的协议和实体。

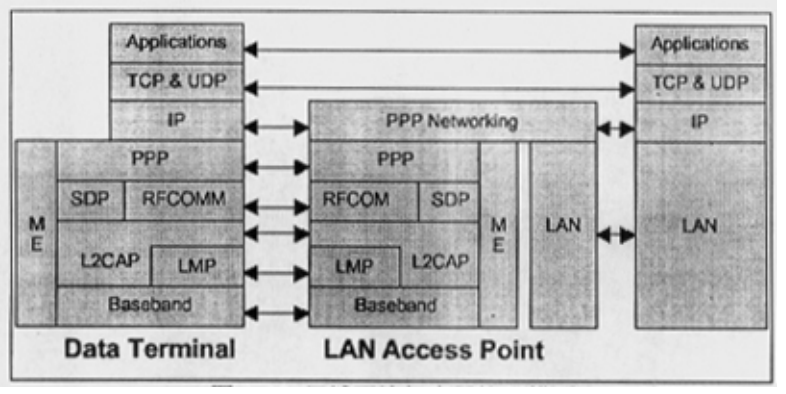

图 3.13 局域网访问应用协议模型

一次完整的连接过程中，DT 和 LAP 交互过程如下：

首先查找在射频范围内的提供 PPP/RFCOMM/L2CAP 服务的 LAP。例如，DT 用户可以用服务浏览程序来查找和选择一个合适的 LAP。

如果在 DT 和 LAP 之间不存在已建立的基带物理链路，DT 就会请求和选定的 LAP 建立基带物理连接。在建立了物理连接之后，设备会进行相互认证。

DT 建立建立 PPP/RFCOMM/L2CAP 连接。

做为可选功能，LAP 可以使用一些 PPP 验证机制（例如 CHAP）。例如，LAP 可能会要求 DT 提供用户名和密码以进行身份验证。如果使用了这些机制而 DT 没有通过身份验证，PPP 连接就会失败。

使用 PPP 的机制，可以在 LAP 和 DT 之间协商 IP 地址。

现在就可以在 PPP 上承载 IP 协议了。

任何时候 DT 或 LAP 都可以中断 PPP 连接。

3.3.2 系统结构

上一节中我们介绍了局域网访问应用的三种使用方式，其中第二种一对多的方式由于现有硬件和协议栈的局限性，还不能实现；而第三种虚拟电缆直连方式非常简单，仅使用串口仿真应用 SPP 即可实现。所以下面仅介绍第一种模仿拨号网络的实现方式，实际生产中我们也这样做的。

LAP 和 DT 对协议和程序的要求并不是完全一样的。我们根据它们的功能定义了不同的系统结构，如图 3.14 所示。

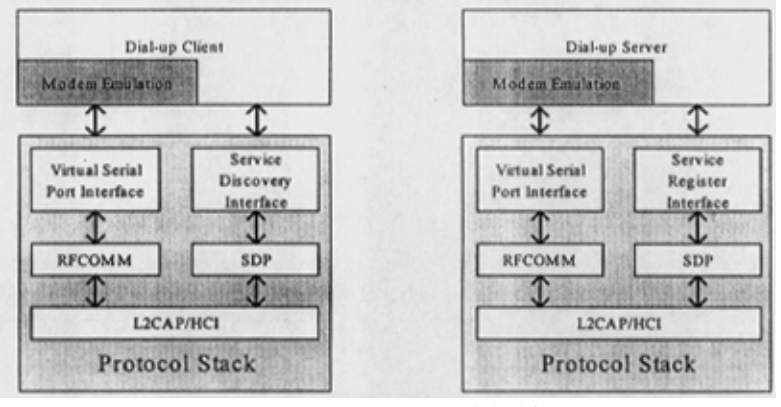

图 3.14 局域网访问应用的系统结构

LAP 应用程序负责向本地 SDP 数据库注册服务的一些信息，包括服务类型和相应的服务信道号，之后启动拨号服务器程序在注册的服务信道号上等待拨入的呼叫。

服务浏览程序负责调用 SDP 输出的接口函数查找附近蓝牙设备上

存在的 LAP 服务。如果找到所要求的服务，LAP 的设备地址和服务信道号就会被返回。这样，DT 就可以用 LAP 设备地址和服务信道号与远端的 LAP 建立连接。

➢ 数据终端（DT）

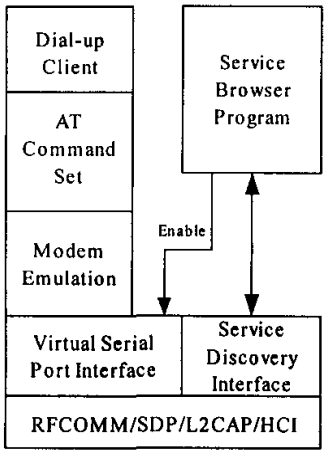

图 3.15 DT 的系统结构

与拨号网络应用的结构相同，局域网访问应用的数据终端也分为拨号客户程序和服务浏览程序两部分。不同之处主要在于拨号客户的参数，由于拨号服务器也由我们实现（见下节），则合法客户的用户名、密码和接入电话号码也可由我们指定。实现中，我们把 LAP 端 Windows 用户的用户名和密码作为拨号客户程序的拨号参数，而电话号码则固化在 RasDial()的参数中，从而兼顾了安全性和易用性。具体实现参见附录 A4.2。

➢ 局域网访问点（LAP）

LAP 的系统结构如图 2.15 所示，包含两个功能模块：服务注册程序和拨号服务器。

在 LAP 接受服务查找之前，必须将服务通过服务注册接口注册到本地 SDP 数据库里。将该服务的属性通过调用虚拟串口接口加入本地设备上的某个虚拟串口（此串口号由蓝牙系统分配）的属性表中并开启该

虚拟串口。接着，再启动拨号网络服务器程序，拨号网络服务器将打开该虚拟串口，初始化 Modem（在这里是指 Modem 仿真实体）并等待远端的呼叫拨入。如果 LAP 想停止提供服务，可以注销已注册的服务。以下是 LAP 的系统结构，如图 3.16 所示。

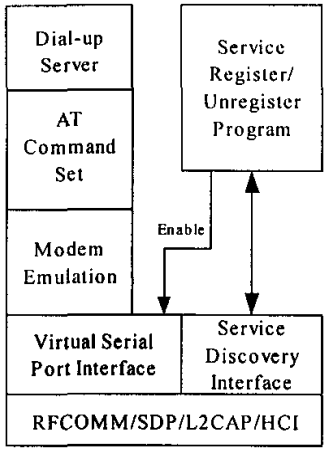

图 3.16　LAP 的系统结构

其中拨号服务器是 Windows 内嵌功能，通过调用系统函数实现，实现代码参见附录 A4.1。

3.4 小结

由前三节的讨论可以看出，三个基于虚拟串口的 profile 在实现中有着许多的相似之处，这促使我们在集成系统的实现中寻找一种模块化策略来提高系统的可重用性和可扩展性。与下一章将要详细阐述的集成系统宏观的分析与软件建模不同，本节主要从三个 profile 之间关系及其对 Windows 环境依赖性的角度讨论实现策略[3]。

首先分析三个 profile 之间关系。概括来讲，三个 profile 的都可以实现为 client-server 模式，而且都依赖于操作系统或第三方提供的一些设备和服务。其中 DUN profile 分别与其他二者发生密切关系，具体总结如下：

- DUN client 与 LAP client 在找到对应服务后都启动 Windows 提供的拨号网络客户程序。而且都依赖于我们在 Windows 下创建的虚拟串口和虚拟 Modem（供拨号客户程序使用）。

- DUN server 与 Fax server 都实现了转发程序，在蓝牙虚拟串口和插有物理 Modem 的物理串口之间转发数据；当然，它们都需要一个物理 Modem。

- PPP 组包模块、拨号服务器程序和传真程序分别是 DUN server、LAP server 和 Fax client 的专有模块。

图 3.17 给出了三个 profile 的集成策略，图中浅灰色部分表示三者的共有模块，深灰色部分表示专有模块；（123）表示用到的 profile，定义 DUN 为 1，Fax 为 2，LAP 为 3 。

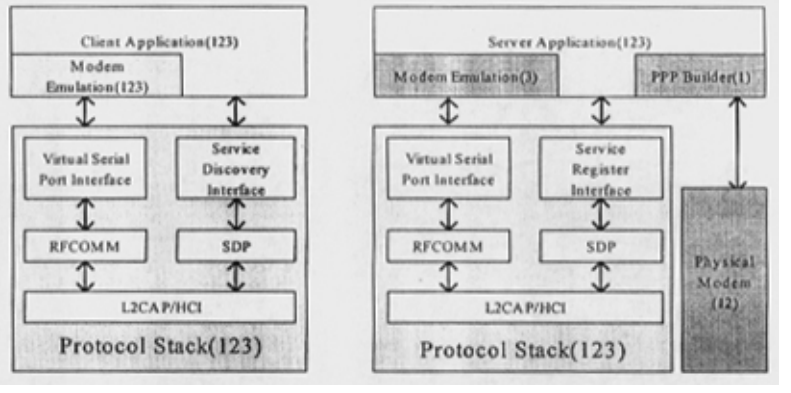

图 3.17 三个 profile 的集成策略

这里需要强调虚拟串口和虚拟 Modem 的作用。它们是系统中的软件实体，分别在注册表中存储自己的串行通信设置（Modem 还包括 AT 命令），通过驱动程序（.vxd 或.sys）与硬件打交道，从而屏蔽了硬件细节向上层应用提供统一的操作接口（API）[12] [13]。串口和 Modem 是 Windows 远端访问服务 RAS 的关键设备，要在 Windows 上实现 RAS 服务基于蓝牙的 legacy porting 就必须实现自己的串口和 Modem。在 profile 的实现中，我们通过开发自己的串口驱动程序，将上层 API 与下层的蓝牙协议栈相连接。事实上，协议栈的 RFCOMM 层之上有一个串

口驱动程序的适配层，这个适配层就是串口仿真应用 SPP。

第四章 集成系统分析与建模

4.1 通信软件的开发过程

凡是用来实现两个或多个实体（计算机，电信终端，交换设备等）之间相互通信的软件都可称之为通信软件[6]。通信软件主要有两大类：电信软件和计算机网络软件。前者包括：电话交换软件、移动通信软件、智能网软件等；后者主要有：各种网络协议和网络应用软件。与开发其他软件类似，开发通信软件需经历一下阶段：

- 需求分析
- 概要设计
- 详细设计
- 编码
- 单元测试
- 集成测试
- 系统测试
- 运行维护

在集成系统开发过程中，我们着重关注前三个步骤。

4.1.1 需求分析

需求分析的主要任务是把用户对软件产品的需求和软件产品的运行环境搞清楚，经过对要求和环境的分析，采用某种形式化的语言对需求进行描述，最后形成需求规格说明书。用户需求分为功能性需求和非功能性需求。对于通信软件而言， 功能性需求主要包括：软件系统应具有的功能、采用协议或信令系统的情况、硬件环境的接口关系、于其他系统交互信息的情况等。非功能性要求主要包括：软件的可移植性、可靠性、实时性、可用性、安全保密性和可重用性等。

4.1.2 概要设计

需求分析阶段解决了系统应该“做什么”的问题，而概要设计和详细设计阶段就是要解决系统“如何做”的问题。概要设计的任务及时根据需求规格说明书，采用合适的形式化语言给出系统的结构设计，划分功能模块，给出模块件接口定义，定义主要的数据结构，设计主要算法等。

在概要设计阶段，可以采用对象建模技术（OMT）把分析结果抽象成不同对象，并描述对象之间的关系。

4.1.3 详细设计

概要设计完成后，系统功能已划分清楚，系统结构也已确定。详细设计的任务就是细化个功能模块的功能，详细设计系统行为，并用形式化语言来描述设计结果。详细设计分功能模块进行，测试计划也应在本阶段完成。

4.2 面向对象的分析与设计

4.2.1 面向对象的方法概述

针对日趋复杂的软件需求的挑战，软件业界发展出了面向对象(OO)的软件开发模式。目前作为针对“软件危机”的最佳对策，OO 技术已经引起人们的普遍关注。

面向对象方法都支持三种基本的活动：识别对象和类，描述对象和类之间的关系，以及通过描述每个类的功能定义对象的行为。为了发现对象和类，开发人员要在系统需求和系统分析的文档中查找名词和名词短语，包括可感知的事物（汽车、压力、传感器）；角色（母亲、教师、政治家）；事件（着陆、中断、请求）；互相作用（借贷、开会、交叉）；人员；场所；组织；设备；地点。

当重要的对象被发现后，通过一组互相关联的模型详细表示类之

间的关系和对象的行为，这些模型从四个不同的侧面表示了软件的体系结构：静态逻辑、动态逻辑、静态物理和动态物理。

静态逻辑模型描述实例化（类成员关系）、关联、聚集（整体/部分）、和一般化（继承）等关系。这被称为对象模型。一般化关系表示属性和方法的继承关系。定义对象模型的图形符号体系通常是从用于数据建模的实体关系图导出的。对设计十分重要的约束，如基数（一对一、一对多、多对多），也在对象模型中表示。

动态逻辑模型描述对象之间的互相作用。互相作用通过一组协同的对象，对象之间消息的有序的序列，参与对象的可见性定义，来定义系统运行时的行为。

静态物理模型通过模块描述代码的布局。动态物理模型描述软件的进程和线程体系结构。

4.2.2 当前主流的面向对象方法

➢ Booch 方法

Booch 方法的过程包括以下步骤[7]：

- 在给定的抽象层次上识别类和对象
- 识别这些对象和类的语义
- 识别这些类和对象之间的关系
- 实现类和对象

这四种活动不仅仅是一个简单的步骤序列，而是对系统的逻辑和物理视图不断细化的迭代和渐增的开发过程。

类和对象的识别包括找出问题空间中关键的抽象和产生动态行为的重要机制。开发人员可以通过研究问题域的术语发现关键的抽象。语义的识别主要是建立前一阶段识别出的类和对象的含义。开发人员确定类的行为（即方法）和类及对象之间的互相作用（即行为的规范描述）。该阶段利用状态转移图描述对象的状态的模型，利用时态图（系统中的

时态约束）和对象图（对象之间的互相作用）描述行为模型。

在关系识别阶段描述静态和动态关系模型。这些关系包括使用、实例化、继承、关联和聚集等。类和对象之间的可见性也在此时确定。

在类和对象的实现阶段要考虑如何用选定的编程语言实现，如何将类和对象组织成模块。

在面向对象的设计方法中，Booch 强调基于类和对象的系统逻辑视图与基于模块和进程的系统物理视图之间的区别。他还区别了系统的静态和动态模型。然而，他的方法偏向于系统的静态描述，对动态描述支持较少。

Booch 方法的力量在于其丰富的符号体系，包括：

- 类图（类结构－静态视图）
- 对象图（对象结构－静态视图）
- 状态转移图（类结构－动态视图）
- 时态图（对象结构－动态视图）
- 模块图（模块体系结构）
- 进程图（进程体系结构）

用于类和对象建模的符号体系使用注释和不同的图符（如不同的箭头）表达详细的信息。Booch 建议在设计的初期可以用符号体系的一个子集，随后不断添加细节。对每一个符号体系还有一个文本的形式，由每一个主要结构的描述模板组成。符号体系由大量的图符定义，但是，其语法和语义并没有严格地定义。

➢ Rumbaugh 的 OMT 方法

Rumbaugh 的 OMT 方法从三个视角描述系统，相应地提供了三种模型，对象模型，动态模型和功能模型[7]。

对象模型描述对象的静态结构和它们之间的关系。主要的概念包括：类，属性，操作，继承，关联（即关系），聚集。

动态模型描述系统那些随时间变化的方面，其主要概念有：状态，子状态和超状态，事件，行为，活动。

功能模型描述系统内部数据值的转换，其主要概念有：加工，数据存储，数据流，控制流，角色（源/潭）。

该方法将开发过程分为四个阶段：

分析：基于问题和用户需求的描述，建立现实世界的模型。分析阶段的产物有：

问题描述

对象模型＝对象图＋数据词典

动态模型＝状态图＋全局事件流图

功能模型＝数据流图＋约束

系统设计：结合问题域的知识和目标系统的体系结构（求解域），将目标系统分解为子系统。

对象设计：基于分析模型和求解域中的体系结构等添加的实现细节，完成系统设计。主要产物包括：细化的对象模型，细化的动态模型，细化的功能模型。

实现：将设计转换为特定的编程语言或硬件，同时保持可追踪性、灵活性和可扩展性。

➢ Coad/Yourdon 方法

Coad/Yourdon 方法严格区分了面向对象分析 OOA 和面向对象设计 OOD。在面向对象分析阶段，该方法利用五个层次和活动定义和记录系统行为，输入和输出[7]。这五个层次的活动包括：

发现类及对象：描述如何发现类及对象。从应用领域开始识别类及对象，形成整个应用的基础，然后，据此分析系统的责任。

识别结构：该阶段分为两个步骤。第一，识别一般－特殊结构，该结构捕获了识别出的类的层次结构；第二，识别整体－部分结构，该

结构用来表示一个对象如何成为另一个对象的一部分，以及多个对象如何组装成更大的对象。

定义主题：主题由一组类及对象组成，用于将类及对象模型划分为更大的单位，便于理解。

定义属性：其中包括定义类的实例（对象）之间的实例连接。

定义服务：其中包括定义对象之间的消息连接。

经过五个层次的活动后的结果是一个分成五个层次的问题域模型，包括主题、类及对象、结构、属性和服务五个层次，由类及对象图表示。五个层次活动的顺序并不重要。

面向对象设计阶段需要进一步区分以下四个部分：

问题域部分（PDC）：面向对象分析的结果直接放入该部分。

人机交互部分（HIC）：这部分的活动包括对用户分类，描述人机交互的脚本，设计命令层次结构，设计详细的交互，生成用户界面的原型，定义 HIC 类。

任务管理部分（TMC）：这部分的活动包括识别任务（进程）、任务所提供的服务、任务的优先级、进程是事件驱动还是时钟驱动、以及任务与其它进程和外界如何通信。

数据管理部分（DMC）：这一部分依赖于存储技术，是文件系统，还是关系数据库管理系统，还是面向对象数据库管理系统。

➢ Jacobson 方法

Jacobson 方法与上述三种方法有所不同，它涉及到整个软件生命周期，包括需求分析、设计、实现和测试等四个阶段[7]。需求分析和设计密切相关。需求分析阶段的活动包括定义潜在的角色（角色指使用系统的人和与系统互相作用的软、硬件环境），识别问题域中的对象和关系，基于需求规范说明和角色的需要发现 use case，详细描述 use case。设计阶段包括两个主要活动，从需求分析模型中发现设计对象，以及针对实现环境调整设计模型。第一个活动包括从 use case 的描述

发现设计对象，并描述对象的属性、行为和关联。在这里还要把 use case 的行为分派给对象。

在需求分析阶段的识别领域对象和关系的活动中，开发人员识别类、属性和关系。关系包括继承、熟悉（关联）、组成（聚集）和通信关联。定义 use case 的活动和识别设计对象的活动，两个活动共同完成行为的描述。Jacobson 方法还将对象区分为语义对象（领域对象）、界面对象（如用户界面对象）和控制对象（处理界面对象和领域对象之间的控制）。

在该方法中的一个关键概念就是 use case。use case 是指行为相关的事务（transaction）序列，该序列将由用户在与系统对话中执行。因此，每一个 use case 就是一个使用系统的方式，当用户给定一个输入，就执行一个 use case 的实例并引发执行属于该 use case 的一个事务。基于这种系统视图，Jacobson 将 use case 模型与其它五种系统模型关联：

领域对象模型：use case 模型根据领域来表示。

分析模型：use case 模型通过分析来构造。

设计模型：use case 模型通过设计来具体化。

实现模型：该模型依据具体化的设计来实现 use case 模型。

测试模型：用来测试具体化的 use case 模型。

➢ 统一建模语言 UML

UML（Unified Modeling Language），称为统一建模语言。它结合了 Booch，OMT，和 Jacobson 方法的优点，统一了符号体系，并从其它的方法和工程实践中吸收了许多经过实际检验的概念和技术，非常适于分析和设计阶段的系统建模。UML 已于 1997 年 9 月提交给对象管理集团 OMG，成为面向对象方法的标准[6]。

UML 由视图(view)、模型元素(model element)、通用机制(general mechanism) 等几部分构成。

- 视图

视图是表达了系统某一方面特性的UML建模组件的子集，它从不同角度出发，为系统建立多个模型，且都反映同一个系统。UML中的视图包括：用例视图、逻辑视图、组件视图、并发视图、配置视图等五种。

- 图

每个视图由一组图组成，图用图形符号表示，代表系统中的模型元素。UML中的图包括：用例图、类图、对象图、状态图、顺序图、协作图、活动图、组件图、配置图共九种。

- 模型元素

可以在图中使用的概念统称为模型元素，模型元素在图中以其相应的图形符号表示。UML规定了用例、类、对象、包、状态、节点、组件等模型元素的图例；模型元素的关系也是模型元素，如关联、泛化、依赖、聚合等，UML也规定了这些关系的图例。这些元素中的大部分将在后续的建模过程中出现，不再一一列出。

- 通用机制和扩展机制

UML利用通用机制为图附加一些信息；利用扩展机制向用户提供定制UML的能力。

4.2.3 使用UML进行对象建模

用UML语言构造系统模型时，应不是只建立一个模型，而是在系统开发的不同阶段都要建立不同的模型。UML支持静态和动态建模方式。所有的系统均有静态结构和动态行为，UML提供图来描述系统的结构和行为，其中结构可以用静态模型元素来描述，行为描述了结构内部元素如何交互。

在系统开发的需求分析阶段，可使用用例图和类图建立系统的静态模型，使用状态图、顺序图、协作图和活动图描述系统的动态行为，即建立动态模型。对于实时系统的动态建模还包含组件图和配置图[16]。在蓝牙集成系统的分析和设计阶段，我们主要应用静态建模技术。

➢ 静态建模

1. 用例图

用例图用于显示若干角色以及这些角色与系统提供的用例之间的连接关系，用例是系统 提供的功能描述。在进行定义系统、发现角色和用例、描述用例、定义用例之间的关系、验证最终模型等工作时，需要建立用例模型。

用例模型可供各种不同的人员使用。客户（最终用户）使用它，开发者使用它，因为他帮助开发者理解系统应该做些什么工作，为将来的开发（如建立其他模型、设计和实现结构）奠定基础；因为它详细说明了系统应有的功能（集），且描述了系统的使用方法，这样当客户选择执行某个操作之前，就能知道模型工作起来是否与其愿望相符合；系统集成和测试人员使用它，因为用它可以验证被测对象的实际运行与图中说明功能（集）是否一致；另外，市场、销售、技术支持和文档管理人员也同样关心用例模型。

2. 类图和对象图

类和对象是面向对象系统组织结构的核心。类表示被建模的应用领域中的离散概念，如物理实体、商业事务、应用事物等。而对象就是可以控制和操作的实体，是用来构造实际运行系统的个体。类是对象的抽象描述，它包括属性和行为两个方面：属性描述类的基本特征（如汽车的颜色、型号等），行为描述类具有的功能，即其可以进行的操作（如汽车能启动、行驶、制动等）。对象是类的实例化，所有的操作都是由对象完成的。建立面向对象模型的基础就是类、对象及它们之间的关系。

➢ 动态建模

1. 状态图

状态图根据对象的生存周期建立模型，来描述对象随时间变化的动态行为。它显示了类的所有对象可能具有的状态，以及引起状态变化的事件。状态图仅适用于具有下列特点的类：具有若干个确定的状态，类的行为在这些状态下会受到影响，且被不同的状态改变。

2．顺序图和协作图

对象间的相互作用体现了对象的行为，状态图在一个是可只集中描述一个对象，要确定整个系统的行为必须同世界和多个状态图来考察。顺序图和协作图正适于描述以组对象的整体行为，它们构成了对象间交互关系的模型。

交互实质上指一个消息集合的传递，顺序图和协作图在表示消息的传递方式上有所不同：顺序图突出消息的时间顺序，主要反映对象间发送消息的先后次序，说明对象间的交互过程，以及系统执行过程中在某一具体位置会有什么事件发生；协作图则突出了交换消息的对象之间的关系（称为上下文相关）。

3．活动图

活动图是一种特殊形式的状态机，用于对计算流程和工作流程的建模，即它是强调计算过程重顺序和并发步骤的状态机。活动图通常在设计前期的分析用例、理解多个用例的工作流、处理多线程应用时使用。

4.3 蓝牙集成系统需求分析

需求分析的主要任务是弄清楚软件产品要做什么，并采用形式化的语言对需求进行描述。在蓝牙集成系统的需求分析阶段，作者首先讨论了系统的功能要求，之后使用 UML 定义的用例图和活动图对系统需求进行描述。

4.3.1 系统功能描述

本集成系统对应于蓝牙通信系统的高层应用部分（图 2.1），设计目标是在现有蓝牙硬件设备和蓝牙协议栈基础之上，将基于 SPP 的诸项应用集成在一起，向用户提供统一的管理和操作方式。其主要功能应包括：

- 各应用（profile）的实现。此项为集成系统的基本需求，实现方式已在第三章中给出；

- 统一的profile管理。主要指client端向server进行一次连接后，连接参数纪录在本地的保存，修改与删除；
- 蓝牙设备管理。主要包括硬件设备的属性、工作模式，角色选择和安全模式等；
- 虚拟设备管理。如第三章所述，在windows下为实现基于串口的通信软件的legacy porting，需要实现虚拟串口和虚拟Modem，并对这些虚拟设备进行简单管理，如添加，删除和配置等；
- 清晰友好的用户界面；

4.3.2 用例图与活动图

由集成系统的功能描述可得系统顶层用例图如下，

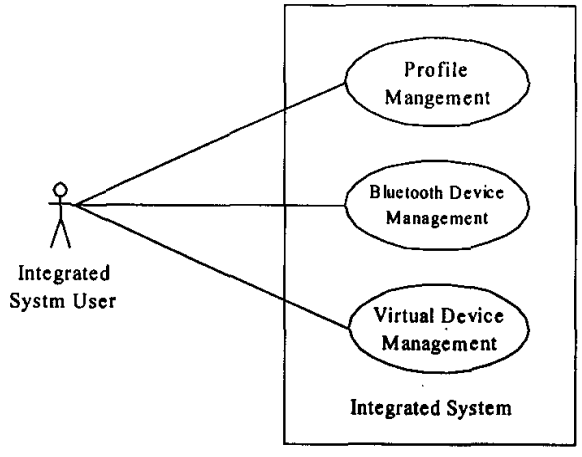

图 4.1 集成系统顶层用例图

整个集成系统需求可以初步分解为三个相对独立的用例：1. 蓝牙应用管理(Profile Management)，负责实现和管理诸蓝牙应用；2. 蓝牙硬件设备管理(Bluetooth Device Management)，负责硬件设备的工作模式，角色选择和安全模式等；3. 虚拟设备管理(Virtual Device Management)，负责虚拟串口和虚拟Modem的添加，删除和配置，并向

蓝牙应用模块提供可用的虚拟设备；其中蓝牙应用管理是集成系统的核心功能。图 4.2 至图 4.4 给出了三个子用例的用例图。

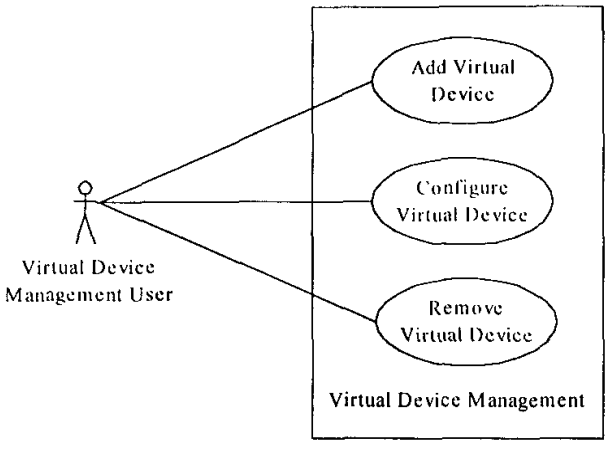

图 4.2 虚拟设备管理用例图

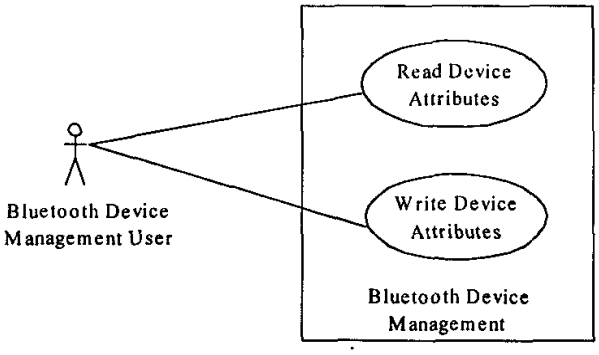

图 4.3 蓝牙硬件设备管理用例图

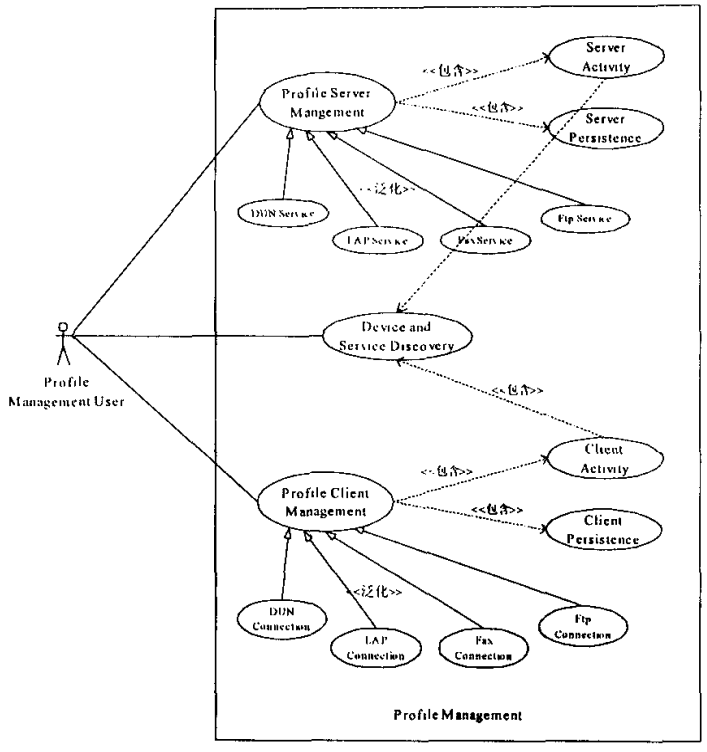

图 4.4 应用管理用例图

由于在已经第三章中给出了基于虚拟串口 profile 的具体实现，这里我们不再关心各 profile 的实现细节，而把注意力放在如何对诸多 profile 的用例进行分析和布局，从而构建出功能完善，结构合理高效的集成系统模型。如图 4.4 所示，根据实现上的相似之处，我们把应用管理细化为三个主要用例：服务管理（Profile Server Magagement）、客户管理(Profile Client Magagement)和设备与服务发现(Device and Service Discover）。

服务管理用例由四类具体 Profile Service 用例泛化而来，并依赖于另外两个相关用例：Service Activity 和 Service Persistance。前者提供抽象的 Profile Service 操作，后者提供 Profile Service 持久化功能。

客户管理用例由四类具体 Profile Connection 用例泛化而来，并依赖于另外两个相关用例：Connection Activity 和 Connection Persistance。前者提供抽象的 Profile Connection 操作，后者提供 Profile Connection 持久化功能。

Profile Service 和 Profile Connection 的活动图可分别参见图 4.5 和图 4.6。

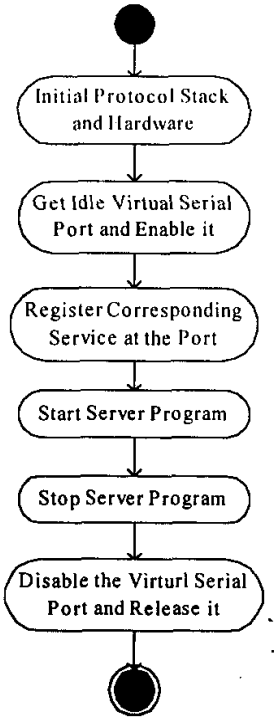

图 4.5 Profile Service 活动图

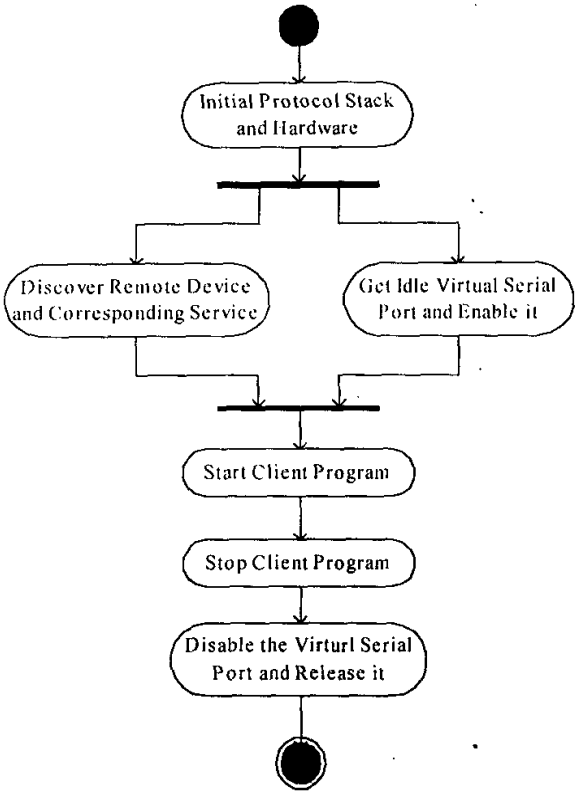

图 4.6 Profile Conncetion 活动图

设备与服务发现用例主要解决两个问题：1、设备发现，2、服务的注册（Profile Service）与发现（Profile Connection），并以方便的形式把周围设备信息提供给服务管理用例和客户管理用例。

4.4 蓝牙集成系统设计与建模

在面向对象的设计(OOD)阶段,我们注意的焦点从问题空间转移到了解空间,主要工作是把分析阶段得到的需求转变为符合成本和质量要求的、抽象的系统实现方案。事实上在面向对象的分析(OOA)和面向对象的设计(OOD)之间已经很难画出一条清晰的界限，从 OOA 到 OOD 是一个逐渐扩充模型的过程[4]。模型细化和分解的基本原则是，利用结构

化的设计方法尽量减少模块之间的耦合度，增强各模块的内聚性和可重用度[5]。本节我们将从需求用例模型出发，依据上述原则完成系统分包，并对各功能包进一步细化得到抽象类。系统运行数据的持久化也是我们在设计中比较关心的问题，本节结尾部分讨论了两种持久化方式。

4.4.1 用户接口(UI) 和业务对象层 (BOL)

作为一个 面向最终用户的软件产品，除强大的功能外，集成系统要求友好而清晰的用户界面。而蓝牙通信系统的功能的复杂性决定其用户界面必然庞大而复杂，我们在设计中不得不将系统的各功能实体与图形用户接口彻底分离，从而得到集成系统顶层包图如下：

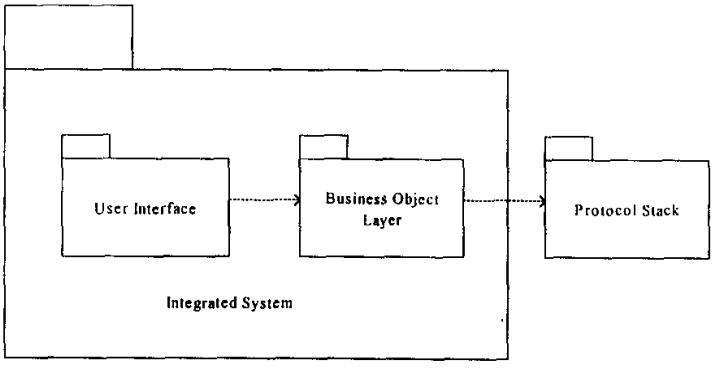

图 4.7 集成系统顶层包图

User Interface(UI)包负责向用户提供图形操作界面，集成系统的各项业务功能由 Business Object Layer(BOL)包实现。作为上层用户界面和下层协议栈之间的实体层，BOL 包主要向 UI 包提供两类功能：1、利用协议栈提供的接口，实现各项高层应用；2、管理虚拟设备。尽管 UI 包庞大而复杂，但 BOL 包因为封装了系统的所有业务实体而成为整个集成系统的核心，因而也是本文工作的重点。依据 4.3.2 中给出的用例图，我们从功能角度对 BOL 包进行进一步细化得到图 4.8。

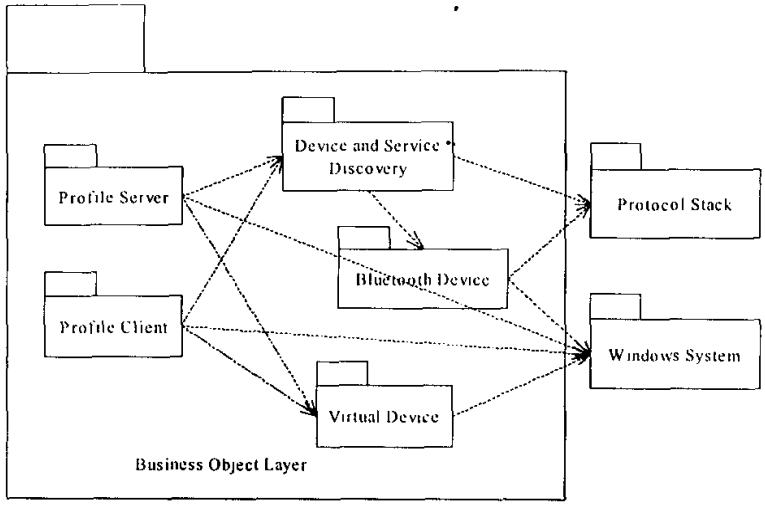

图 4.8 业务对象层包图

BOL 包被进一步细分为五个包：Profile Server 包、Profile Client 包、Device and Service Discovery 包、Bluetooth Device 包和 Virtual Device 包。其中前四个包实现 BOL 的第一类功能，Virtual Device 包实现第二类功能。为表达清晰起见，我们把集成系统之外的 Protocol Stack 和 Window 一些系统功能作为两个包加在图 4.8 中，具体介绍如下如下：

Profile Server 包和 Profile Client 包分别提供拨号网络、传真、局域网接入和文件传输等应用的 Server 端和 Client 端实现，并管理和维护这些 Server 和 Client。它们分别依赖 Device and Service Discovery 包 Virtual Device 包。

Device and Service Discovery 包依赖于系统外的 Protocol Stack 包，负责发现附近的蓝牙设备、浏览指定设备上的服务并维护一个周围设备和服务的列表；Server 端服务注册的功能也由本包提供。

Bluetooth Device 包管理蓝牙硬件设备的属性、工作模式，角色选择和安全模式等。以上信息分别通过 Protocol Stack 包和 Window System 包进行管理。

Virtual Device 包维护虚拟串口和虚拟 Modem；包括两类工作：1、维护虚拟设备的状态信息，向 Profile Server 包和 Profile Client 包提供空闲设备；2、添加，删除和配置虚拟设备。本包只依赖于系统外的 Windows System 包。

作者在集成系统的研发过程中，独立完成了 Profile Server 包、Profile Client 包、Bluetooth Device 包和 Virtual Device 包设计和大部分实现，以及 Device and Service Discovery 包的部分设计工作。在下一节中我们将主要给出前四个包的设计，实现中的问题将在第五章进行讨论。

4.4.2 BOL 数据模型

➢ Profile Server 包

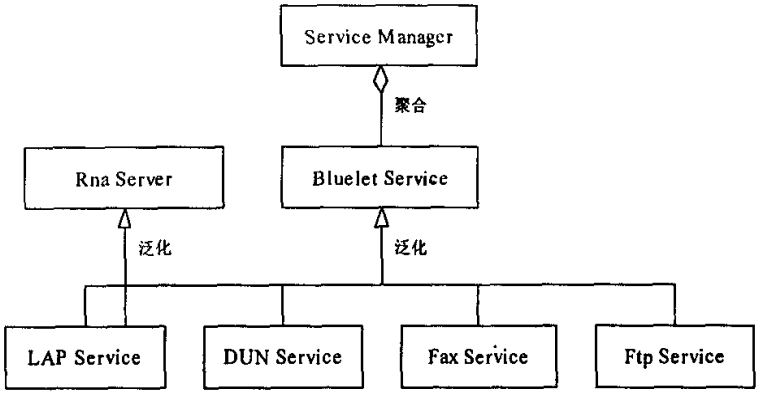

图 4.9 Profile Server 包抽象类图

Profile Server 包抽象类的结构关系如图 4.9 所示，整个包被细分为三个层次。最底层是四个 service 类，分别实现了四个 profile server。它们的具体实现原理和结构已在第三章阐述，这里不再赘述，指出一点：局域网访问点 LAP 的拨号服务器功能由 Rna Server 类实现，LAP Service 类通过继承 Rna Server 类获得其拨号服务器功能。

Bluelet Service 类是抽取底层四个 service 的公共特征而得到的

中间层，从底层具体 Service 类到抽象的 Bluelet Service 类是一个泛化的关系；在实现上，可以先设计 Bluelet Service 类，具体 Service 类通过继承 Bluelet Service 类获得通用功能，之后加入每个 profile 特有功能即可。最顶层的 Service Manager 类可以看作一个 Bluelet Service 对象的容器，是由 Bluelet Service 类聚合而成，主要提供 Bluelet Service 对象的动态管理功能。现分别详述如下：

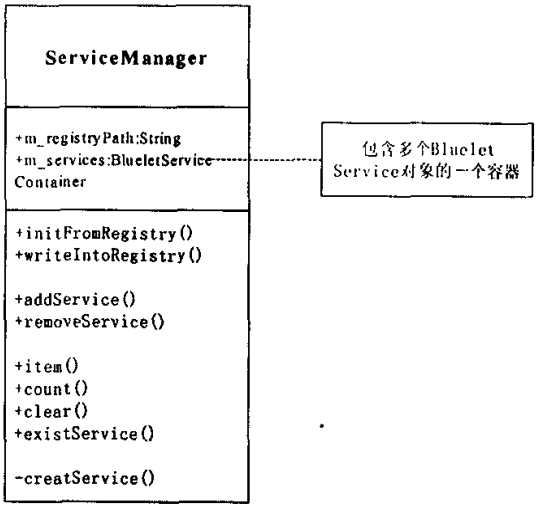

图 4.10 ServiceManager 类图

ServiceManager 类：

ServiceManger 类的核心是成员变量 m_services。它利用 C++标准模板库中的<vector>实现了一个 BlueletService 对象的容器，对数量不确定的 BlueletService 对象进行管理。管理功能包括：添加和删除 service，按照序号和名称获取 service，清空 service 以及检查某 service 是否存在。

成员变量 m_registryPath 保存了 service 信息在注册表中的存储路径，供 initFromRegistry()和 writeIntoRegistry()使用，通过读写注册表完成对 service 信息的持久化处理。具体过程是：在系统初始化阶段创建 ServiceManager 对象，由 ServiceManager 对象调用

initFromRegistry()方法从注册表中读出各个 service 持久化信息存入 ServiceManager 对象；系统退出前由 ServiceManager 对象调用 writeIntoRegistry()方法将运行中产生的 service 信息写入注册表。

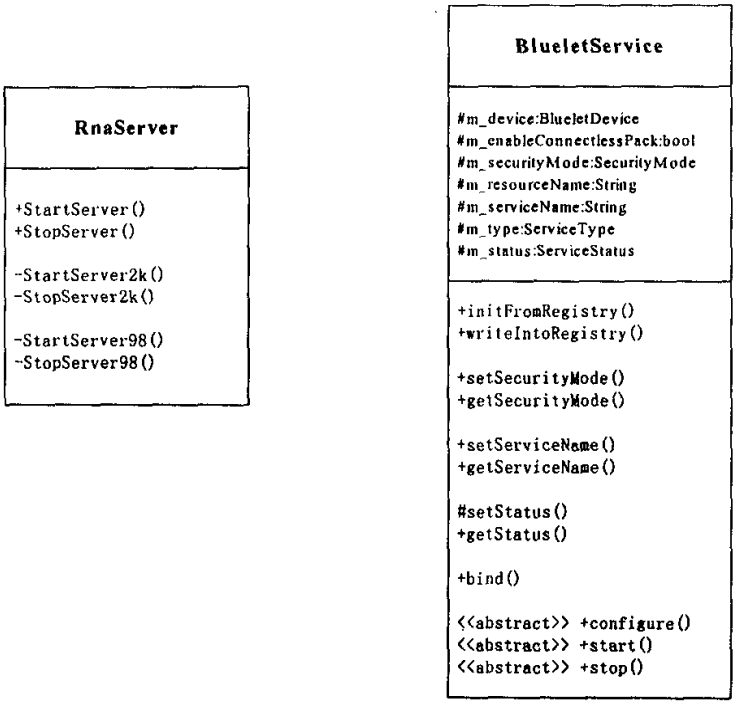

图 4.11 BlueletService 类和 RnaServer 类

BlueletService 类：

BlueletService 类抽象了所有具体 service 的共同特征，这些特征可以分为属性和方法两个范畴。BlueletService 类的属性包括服务名称、服务状态、绑定的虚拟设备、应用级安全属性等；BlueletService 类的方法主要包括上述属性的读写接口和 service 动作。现面简单介绍 service 动作：

与 ServiceManager 类对应，BlueletService 类也有 initFromRegistry()和 writeIntoRegistry()方法，其目的也是通过读写注册表完成对 service 信息的持久化，但二者在层次上有所不同。事

实上，ServiceManager 只是在 initFromRegistry()中创建 BlueletService 对象，之后的注册表操作由 BlueletService 对象调用 initFromRegistry()完成；同理，ServiceManager 类的 writeIntoRegistry()方法也通过 BlueletService 对象的 writeIntoRegistry()方法实现。

start()、stop()和 configure()也是各个 service 共有的动作，但它们的实现各不相同。为兼顾 BlueletService 作为基类的抽象性和实现的多样性，在设计中我们把这三个 service 操作在 BlueletService 类中定义为虚函数，它们的实现在每个具体 service 类中完成。

RnaServer 类：

为实现软件功能清晰化和模块可重用性，我们专门设计了 RnaServer 类来完成局域网访问点的拨号服务器功能，供 LAPService 继承。RnaServer 类对外只提供两种方法：startServer()和 stopServer()，在内部隐藏了 Win98 和 Win2k 下不同的实现。

➢ Profile Client 包

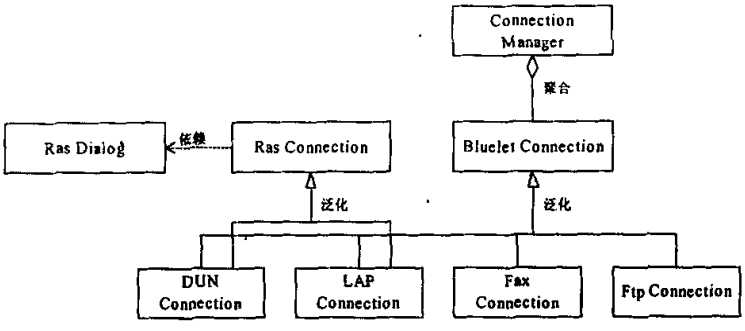

图 4.12 Profile Client 包抽象类图

Profiel Client 包的抽象类结构关系如图 4.12 所示。与 Profile Server 包类似，四个具体 profile client 类（在图中命名为 xxx Connection）泛化为 BlueletConnection 类，多个 BlueletConnection 对象进一步聚合成 ConnectionManager 类，不再赘述。

值得一提的是由于 DUNConnection 类和 LAPConnection 类都实现了拨号网络客户，在设计中我们将拨号网络客户的功能单独放在 RasConnection 类中实现，DUNConnection 类和 LAPConnection 类通过继承 RasConnection 类获得拨号网络客户功能。为实时报告拨号状态，我们用对话框类 RasDialog 向用户提供可视化界面。RasConnection 类和 BlueletConnection 类的细节详见图 4.13、4.14。

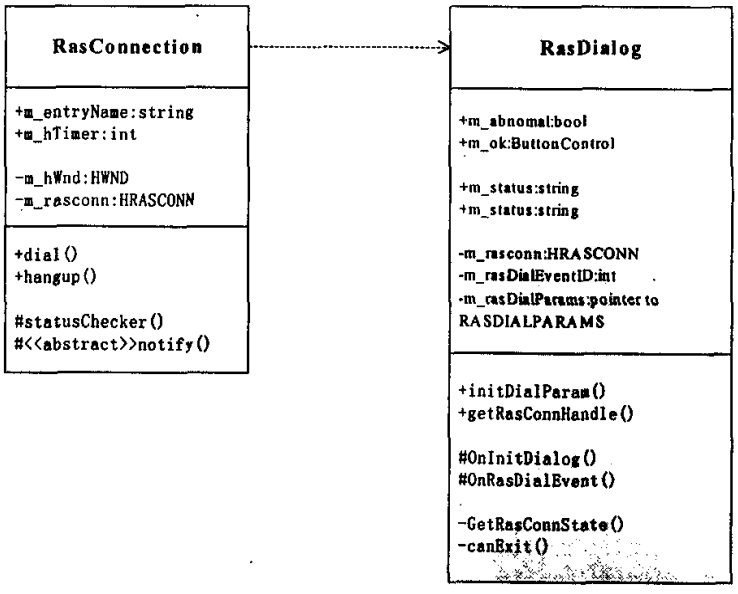

图 4.13 RasConnection 类图

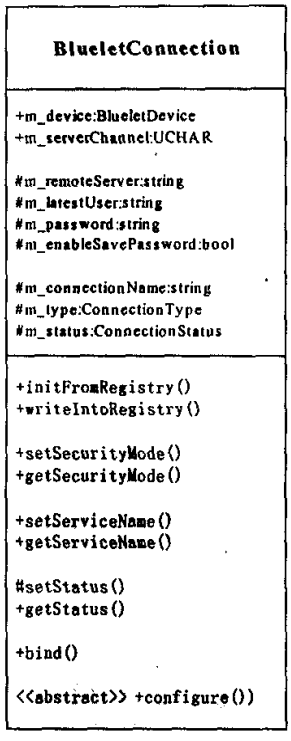

图 4.14 BlueletConnection 类图

➢ Virtual Device 包

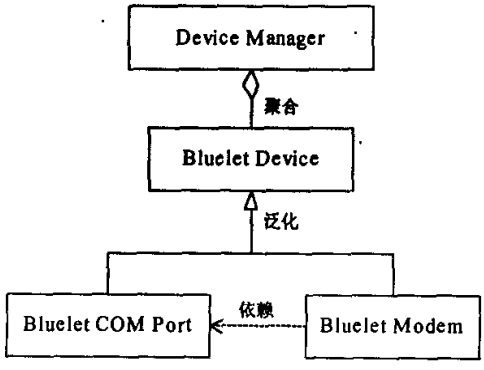

图 4.14 VirtualDevice 抽象类图

在 Virtual Device 包的设计中我们依旧利用依赖、泛化和聚合的关系来进行构造，类之间的关系如图 4.14 所示，其核心是 BlueletDevice 类。

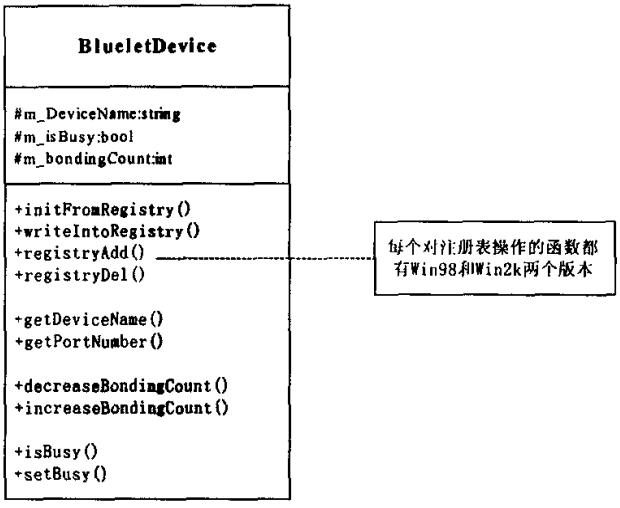

图 4.15 BlueletDevice 类图

Virtual Device 包的主要功能有两项：1、维护虚拟设备的状态信息，向 Profile Server 包和 Profile Client 包提供空闲设备；2、添加，删除和配置虚拟设备。具体方式是： DeviceManager 对象经过与 ServiceManager 类似的初始化过程后开始对 BlueletDevice 对象的管理和维护。当建立新的 Service 或 Connection 时，系统根据不同需要向 DeviceManager 申请空闲 COM Port 或 Modem 进行绑定，绑定之后的设备的绑定计数(m_bondingCount)加一；绑定释放后绑定计数(m_bondingCount)减一；Service 或 Connection 启动时，绑定的设备状态(m_isBusy)置为忙；Service 或 Connection 停止时，绑定的设备状态(m_isBusy)置为闲。用户可以根据 device 的绑定计数和状态信息通过 DeviceManager 对 device 对象进行添加，删除和配置。

由于 BlueletCOMPort 在读取和修改属性，添加和删除的方式等方面不尽相同，而且同种设备的同种操作在 Win98 和 Win2k 下也完全不同，我们在 BlueletDevice 类中把 initFromRegistry()，

writeIntoRegistry(), registryAdd(), registryDelete()四个方法设计成虚函数。Virtual Device包实现的主要工作量集中在BlueletCOMPort和BlueletModem类的上述四个函数的实现，具体讨论在第五章给出。

➢ Bluetooth Device包

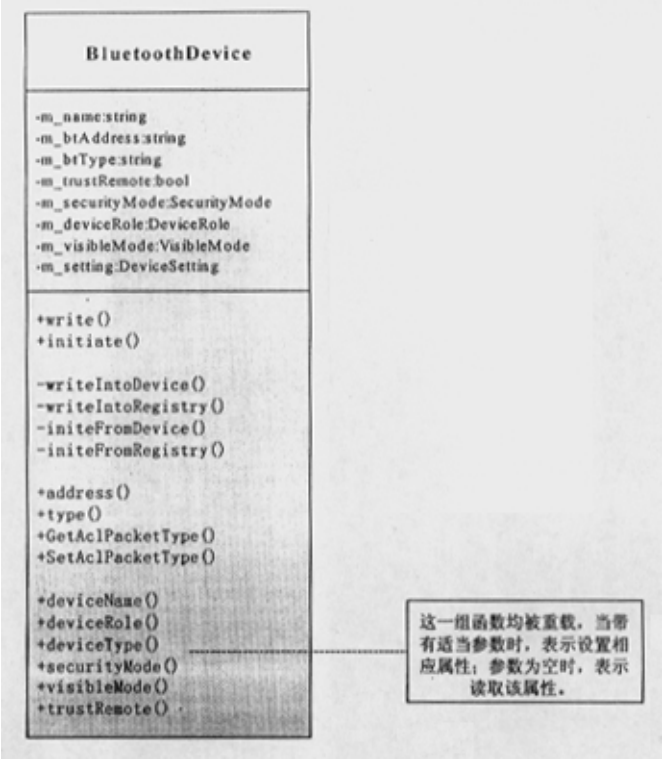

图 4.16 BluetoothDevice类图

Bluetooth Device包结构比较简单，只有一个类：BluetoothDevice。BluetoothDevice类提供对蓝牙硬件设备属性的访问，包括设备名、硬件地址、设备类型、设备角色、安全模式、以及对呼叫和配对请求的反映模式等一些杂项。包含两类操作：一类是直接调用HCI层的接口命令读写蓝牙硬件，如设备名、硬件地址和设备类型等；其他属性属于系统的持久化信息，由集成系统和下层协议栈约定一些固

定位置存放（注册表或文件），当协议栈与其它蓝牙设备通信时将遵循这些持久化属性。

以上 BOL 数据模型的细节参见附录 2。

4.4.3 持久化处理

应用程序运行期间产生的数据、对象和结构等在程序退出后需要保存，供程序下次运行时使用，称为数据的持久化。Bertrand Meyer 在他的 Object-Oriented Software Construction 中讨论了两类持久化策略[4]。一种简单而常用的策略是通过编程语言和应用环境自身实现持久化，实现方式是将数据存放在文件中，通过读写文件完成对象和结构数据的保存和初始化。这种方式适用于简单的数据结构，访问方式也比较简单。另一种持久化策略是利用数据库技术，这种策略提供独立于应用程序的，支持多种访问方式和权限的复杂数据的管理方式，但显然实现也相对复杂。当存在现成数据库工具时，策略二不失为一种简便有效的方法。

在 Windows 环境下集成系统的设计中，两种策略都有用武之地。

➢ 1）注册表结构设计

注册表是 Windows 保存系统信息和用户信息的一种数据库，它具有自己的目录结构和数据类型，Windows 还提供了一组专门的 API 来访问注册表。以上特征非常适合存放 BOL 中各个对象的持久化信息，包括各 service 和 connection 对象的名称、绑定设备、安全属性，connection 对象的远端 service 设备地址、serverChannel，以及 BlueletDevice 和 BluetoothDevice 的参数等。下面给出注册表结构的概要设计，详细结构见附录 3。

顶层目录：[HKEY_LOCAL_MACHINE\SOFTWARE\IVT Corporation\Bluelet Software Suite]

下层目录：

\Bluelet Service\

```
        \LAN Access Using PPP\
            \LAP Service1\
        \Dialu-up Networking\
            \DUN Service1\
            ......
    \Bluelet Connection\
        \LAN Access Using PPP\
            \LAP Connection1\
        \Dialu-up Networking\
            \DUN Connection1\
            ......
    \Bluelet Device\
    \Bluetooth Device\
```

➢ 2）bttl.ini 文件

除了在注册表中保存的结构化数据信息外，集成系统还有一类相对零散而简单的数据需要进行持久化，主要是传输层设置。协议栈启动时需要根据不同 HCI 传输层装载不同的设备驱动程序，如果是 USB 或 PCMCIA 设备，则只需通知协议栈设备类型和驱动程序路径；若是 UART 设备还需指明串口号、波特率、校验方式等串口参数。

上述传输层设置信息需要在一次运行退出之后进行保存，以便用户下次启动时默认使用此设置。我们采用读写文件的方式实现传输层设置信息持久化，所有相关信息放在名为 bttl.ini 的初始化文件中。作为具有特定格式和访问方式的 Windows 文件类型，初始化文件十分便于存放此类数据，bttl.ini 的结构如下：

```
[COM]
```

```
    DLL =D:\Program
Files\IVTCorporation\MiniBluelet\Pan\Driver\COM\ivt_com.dll
    BAUD_RATE = 57600
    STOP_BITS = 1
    FLOW_CONTROL = None
    BYTE_SIZE = 8
    PARITY = None

    [USB]
    DLL =D:\Program
Files\IVTCorporation\MiniBluelet\Pan\Driver\USB\ivt_usb.dll

    [PCMCIA]
    DLL =D:\Program
Files\IVTCorporation\MiniBluelet\Pan\Driver\USB\ivt_pcmcia.dl
l

    [DEFAULT]
    TL =COM1
```

第五章 系统实现中的若干问题

蓝牙集成系统作为一个完整的软件产品的开发过程中，我们的工作除了系统核心模块的分析、设计和编码实现以外，还包括了安装程序、卸载程序、设备管理模块以及测试方案等系统支撑模块的设计与实现。本章首先介绍第四章所建模型的实现，之后分别讨论几类支撑模块，最后对集成系统实现中的技术难点进行有针对性的探讨。

5.1 核心模块的实现

5.1.1 BTFrame.exe 与 BTBOL.dll

在 4.4.1 节中，我们将整个集成系统在最顶层设计为两个大包：User Interface(UI) 和 Business Object Layer(BOL)。前者负责向用户提供图形操作界面；后者提供集成系统的各项业务功能。在 Visual C++开发环境下，实现我们将 UI 包实现为基于 MFC 的应用程序(.exe)，而将 BOL 包实现为基于 MFC 的动态连接库(.dll)。在 BTBOL.dll 中，集成系统的各项业务功能作为接口函数输出给 BTFrame.exe，这样实现的好处是：UI 和 BOL 分为功能明确的两个工程，二者中任何一方改变后，只要接口不变，则不会影响另一方，便于合作开发与维护。

虽然 BTBOL.dll 实现了集成系统的各项主要业务功能，但有一些和界面关系密切又不便归入 BOL 的简单功能我们放在了 BTFrame.exe 中实现，这里简单介绍的两个功能类。

TransportLayerWizard 类向用户提供直观的图形界面用以修改传输层设置。该类主要对 bttl.ini 文件的操作进行了可视化封装；定义了传输层结构；在初始化时自动获取传输层的当前设置并将用户所设新值写入文件。该类的实现并不复杂，但由于结构合理、接口清晰，在开发过程中多次被同伴引用。

HardwareManager 类实现即插即用(PNP)设备的自动检测。4.4.3 节我们曾提到了协议栈的三种传输层：UART、USB 和 PCMCIA，其中 USB 和 PCMCIA 就属于 PNP 设备。PNP 设备的特点是其插拔状态由操作系统

实时监控并反映在系统注册表中，据此我们设计了 HardwareManager 类来检测当前活动的 PNP 设备，其工作流程是：在初始化阶段从 USB 和 PCMCIA 的安装信息文件(.inf)中读取当前支持的 PNP 设备列表；之后从注册表中读取当前活动的 PNP 设备并与列表比较；最后根据比较结果做出适当输出。

5.1.2 对象的动态管理--C++容器与迭代子

在 4.4.2 节的 ServiceManager、ConnectionManager 和 DeviceManager 类的设计中，我们反复用到了 C++中“容器”的概念，那么容器到底是什么，又是怎样使用的呢？

简而言之，C++容器是由 C++标准模板库(Standard Template Library—STL)提供的用来容纳对象的对象[12]。对象可以放置在容器中，从容器中移除，或由容器进行组织和访问。STL 提供两种容器：顺序容器(Sequence Container)以顺序方式存取数据；关联容器(Associative Container)按照其包含对象之间的关系进行存取操作，换言之关联容器以归类的方式存取数据。

迭代子(iterator)是一种面向对象的广义的指针，用来遍历容器或者其他序列中的所有成员。在遍历过程中，迭代子是作为容器或序列中对象的位置定位器使用，当迭代子到达指定的对象，就可以通过解析迭代子来访问这个对象。与 C++指针指向数据在内存中的存储位置不同，迭代子是指向对象在容器或序列中的相对位置。

在附录 A4.3 中，我们以 ServiceManager 类的实现为例，介绍容器与迭代子的用法。由于 ServiceManager 所管理的 BlueletService 对象之间只是简单的聚合关系，并不发生复杂联系，所以我们选用了顺序容器 vector 来实现 ServiceManager 类。

5.2 支撑模块

作为完整的解决方案，我们的集成系统开发工作除了系统核心模块的分析、设计和编码实现以外，还体现在系统支撑模块的设计与实现

上。支撑模块主要包括：安装程序，卸载程序，设备管理模块。

5.2.1 安装与卸载程序

InstallShield系列安装工具是由InstallShield软件公司开发的基于Windows的安装盘开发环境。InstallShield系列是当前主流的安装盘制作工具，其中InstallShield for Visuall C++版因作为OEM嵌入在微软的MSDN工具包中而广泛流传。我们在开发中使用的是InstallShield Professional 6.3版，该版本除提供标准对话框和文件拷贝等基本功能外，还提供了注册表编辑、多语言安装、快捷方式创建以及强大的脚本语言等功能。这些丰富的功能极大的便利了我们的开发工作。

集成系统的安装工程主要包括以下几个方面：

首先是文件的归类与拷贝。InstallShield提供三层文件管理模式：最底层是按照不同目标路径而划分的File Group，如<TARGETDIR>表示用户在安装对话框中指定的路径，<WINDIR>表示Windows系统目录（在Win98或ME下是\Windows而在Win2k下是\WinNT），<PROGRAMFILESDIR>表示\Program Files；中间层是Component，一个Component由一个或多个File Group组成；最上层是Setup Type，通常有三种：Typical、Compact、Custom，每一种Setup Type又可以包含一个或多个Component。由于集成系统就代码长度而言并不是一个很大型的软件（所有总共有4MB左右），结构也并不复杂，因而我们只创建了一个Component来包含所有目标代码，并根据不同安装路径将它们分配到四个File Group中。

其次是安装脚本的编写。InstallShield提供了便捷的脚本编辑和调试环境、丰富的API函数和数据类型、以及简单的程序框架结构，借助它们用户可以方便的完成创建对话框、文件拷贝、读写注册表和初始化文件等功能。InstallShield脚本甚至还提供了调用系统API和第三方动态链接库的功能，这一点对我们尤其重要，在安装阶段添加虚拟串口和虚拟Modem的工作就是在我们自己写的设备管理模块setup.dll中完成的。安装脚本除了调用setup.dll添加虚拟设备外，还贯穿了从

弹出Welcom对话框之前到关闭Finish对话框之后再到系统卸载的整个过程，覆盖了检测版本信息、获取用户输入、根据输入拷贝或删除文件、设置注册表和bttl.ini文件等多种操作，不再一一列举。

最后是快捷方式和多语言支持。对这两项功能InstallShield都提供了简单的设置方式，前者只需在控制板中设置相应的创建路径和执行路径即可；后者需要定义一个语言无关的String Table，给出每个字符串在不同语言下的值，然后在脚本中将具体语言换成抽象的字符串。

另外值得一提的是手工卸载程序。事实上InstallShield提供从安装到卸载的全套功能，其机制是安装程序setup.exe以连续执行两次为一个周期，第一执行安装功能并记录所有操作，第二次执行卸载功能，根据上次记录进行删除与恢复，从而避免在无意的情况下连续执行两次setup.exe时造成的目标代码版本混乱。但好的初衷并不意味着好的结果，在实践中常常发生的问题是某一次卸载出现异常后（往往因为用户操作不善导致），系统再也无法重新正常安装或卸载，而是反复执行卸载模式。经过分析笔者发现这种无法自行恢复的情况是由于最初一次卸载不彻底造成的，那次卸载删除了记录文件却没有删除第一次安装的标志，因而每次都进入卸载模式却什么也没做。在这种情况下，我们必须手工完成上次未完的卸载工作。由于同类问题在小组内部反复出现，作为暂时手段笔者编写了手工卸载程序来自动完成这一功能，实践表明程序是有效的。

5.2.2 设备管理模块

设备管理模块是集成系统重要的支撑模块，主要供安装程序用来添加和删除虚拟串口和虚拟Modem。它与BOL中的Virtual Device包既有区别也有联系。先说区别：从应用目的讲，Virtual Device包是为了与UI包相配合，在应用程序内部向用户提供全套的设备添加删除、属性读取和写入等管理和维护功能；设备管理模块仅仅是设计用来被InstallShield脚本或Virtual Device包调用，是独立于应用程序的。从功能角度讲，设备管理模块仅实现了设备添加删除的功能，是

Virtual Device 包的一个子集。二者的相同点主要是设备添加删除功能的实现方式。

在实现中我们将设备管理模块实现为动态链接库 setup.dll。所谓 Windows 设备，实际上代表存储在注册表中的一系列目录和健值，我们在 Windows 设备管理器中所作的操作最终都反映为注册表的变动。Windows 设备操作是系统相关的，在 Win98 \ME 和 Win2k 下有不同的实现，在 setup.dll 中我们隐藏了这种区别，对外输出单一接口。下面分别介绍：

Win2k 提供了标准的设备管理和注册表访问 API，通过这些系统 API 用户可以方便地访问设备和设置属性而不必关心具体的注册表路径和键值。setup.dll 中 Win2k 版本的操作比较简单，不再详述。

Win98 中问题相对复杂，解决这些问题我们经历了一个过程。由于 Win98 并不提供类似于 Win2k 的系统 API，因而我们首先采取了直接写注册表的方法来进行安装，把串口和 Modem 安装信息文件（.inf）中的条目逐条写入注册表。这种方法在获取串口号的方式上存在隐患。我们的算法是从注册表的 HKEY_LOCAL_MACHINE\Enum\Root\Ports 路径下穷举系统当前所有串口，然后从小到大选择一个可用串口号。这种方法在台式机上屡试不爽，但是在笔记本上却屡屡出错。仔细研究之后我们发现笔记本上通常都装有内置 Modem，为此 Modem 厂商都开发了自己的虚拟串口驱动程序，安装完成后内置 Modem 的虚拟串口并不出现在上述注册表路径中，其结果是我们获取的虚拟串口号是系统中已经存在的，我们的串口安装完就把人家的冲掉了。

为解决这一问题，我们参照 Win2k 中由系统分配串口号的办法，在 VC1.5 中找到了为 Win95 提供的设备管理 API，在 VC1.5 环境下编了一个可执行程序来添加串口。用这种方式单独添加一个串口是成功了，再也不会出现串口号冲突的问题，但却引入了新问题：当在 VC6.0 或 InstallShield 脚本中连续调用该程序时，除第一个外后续串口仍会添加失败。我们分析可能是新老版本的兼容性不好造成的，需要另想办法。

查阅有关资料后，我们在注册表中找到了另外的串口存储路径：HKEY_LOCAL_MACHINE\Hardware\DeviceMap\SerialComm，该路径下的串

口列表由系统启动自检是得到，因而是真实而完备的。为保险起见还对算法进行小改动：穷举完该路径下串口号后，由原先的从 COM3 开始从小到大选择改为从 COM15 开始，因为大多数虚拟串口都集中在 COM3 到 COM10 范围内。经过这两项改进后，Win98 下添加虚拟串口的问题得到了较好的解决。

5.3 几点讨论

5.3.1 PPP 组包算法的改进

正如我们在 3.1.2 节中所讨论的，原有组包算法复杂，代码长而效率低。下面给出原有算法和现有算法流程图并进行简单介绍，实现代码见附录 A4.4 和 A4.5。

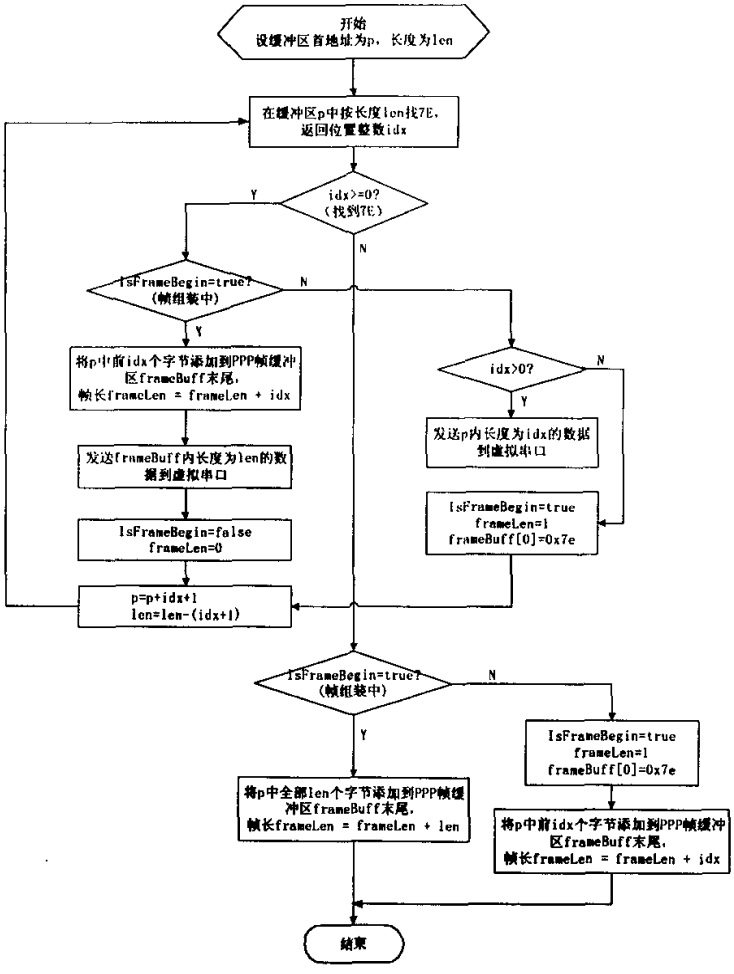

图 5.1 改进前的 PPP 组包算法

改进前的组包算法是面向字节串的，其基本思路是主动在一段缓冲区中搜索 PPP 的包头 7E，找到后将 7E 之前的数据（含 7E）根据当前组包状态作不同操作：若在**组包**中状态，则将数据添加到一段以 7E 开头的帧缓冲区的尾部并将其发送到虚拟串口，组包状态转到**发新包**状态；若在**发新包**状态，则将数据直接发送到虚拟串口，组包状态转到**组包**中状态。当找不到 7E，若在**组包**中状态，则将数据添加到一段以 7E 开头的帧缓冲区的尾部，状态不变；若在**发新包**状态，则重置帧缓冲区

并将数据添加到其尾部，组包状态转到**组包**中状态。详见图 5.1。

由于 7E 之间的字节串在一段缓冲区中出现的情况是多种多样的，因而上述算法中事实上分别处理了 四种情况：找到或找不到 7E*组包中或发新包。人为地将问题复杂化。

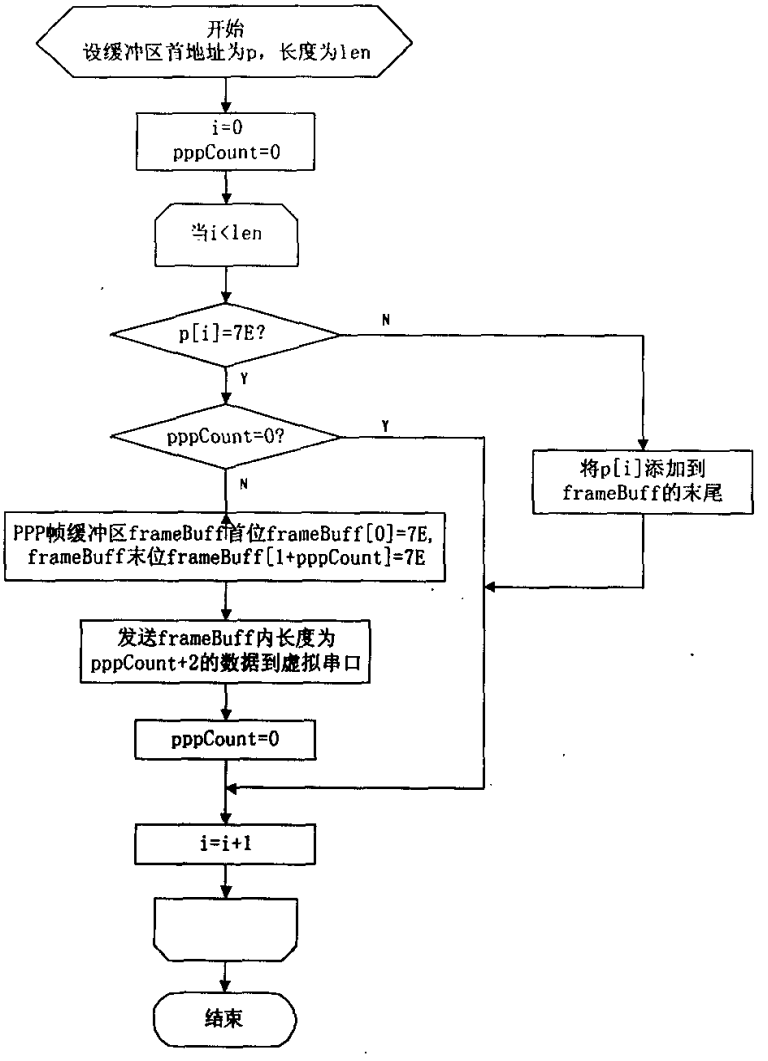

图 5.2 改进后的 PPP 组包算法

改进后的组包算法摒弃了面向字节串的思想，改为面向字节。基本思路是对缓冲区中的数据逐字节判断，若不是 7E 则添加到以 7E 开头的帧缓冲区的尾部，若 是 7E 则进行发送。从而简化了算法，提供代码效率，细节见图 5.2。

5.3.2 DUN 和 Fax Profile 中 GW 端的蓝牙数据访问方式

首先让我们回顾 3.1 节和 3.2 节中关于 DUN 和 Fax 的 GW 端的实现，它们都有一个转发程序在蓝牙虚拟串口和插有物理 Modem 的物理串口之间转发数据。与三个 profile 的 client 端和 LAP server 移植第三方上层应用的实现机制不同，DUN 和 Fax 的 GW 是由我们自己实现的。结合在 3.4 节中介绍的系统串口和 RFCOMM 之上的虚拟串口适配层之间的关系，可以看出在 GW 程序中访问来自蓝牙协议栈的数据时有两种方案备选：一种是在串口驱动程序之上，通过访问虚拟串口的方式读写蓝牙数据，这种方式与访问真正串口相同；另一种是串口驱动程序之下 SPP 之上，使用 SPP 接口读写蓝牙数据。两种方式如图 5.3 所示。

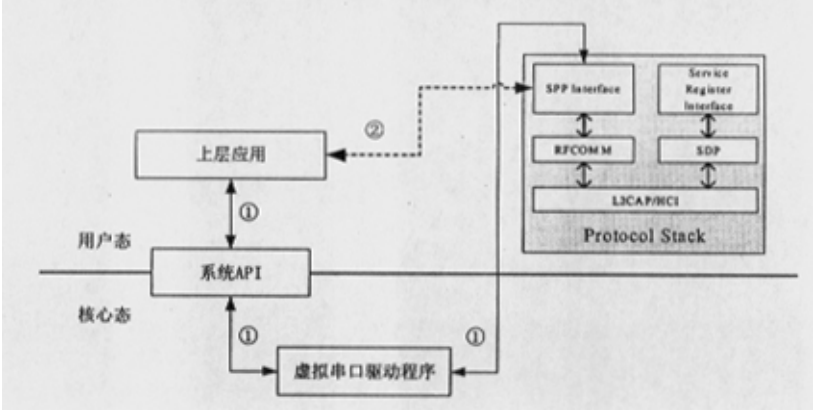

图 5.3 两种蓝牙数据访问方式

两种访问方式各有优缺点：方式一的优点是使用系统 API 访问虚拟串口，对上层应用而言虚拟串口与真实串口并没有不同。但由于数据流经过了串口驱动程序转接，即便转接过程是透明的（并不添加包头包尾），也仍然增加了开销和延时。方式二由于直接使用 SPP API，减少

了数据流转接路径，效率更高。

考虑到集成系统所应用的桌面系统通常都具有更快的CPU和更大的内存（与嵌入式系统相比），因而延时差距并不显著；另一方面由于三个profile client和LAP server都只能使用方式一，我们专门在BOL包中设计了虚拟设备管理模块DeviceManager来支持这种方式。为统一操作，增加系统可维护性，我们在DUN和Fax的server实现中采用了方式一访问蓝牙数据。

第六章 结束语

6.1 课题总结

桌面系统上的解决方案是蓝牙技术的两个主流应用方向之一，而使用无线链路替代串行电缆又是蓝牙技术提出的一个最初动因，因而基于Windows平台、综合了诸串口应用的集成系统的研究具有其技术先进性和巨大的现实意义。国际认证与测试公司（IVT）于2000年底启动蓝牙桌面系统开发项目，在国内首次提出了蓝牙通信系统桌面集成解决方案并将其产品化。。

笔者作为主要研发人员参加项目期间，认真研究了蓝牙协议，深入分析了串口仿真应用及其之上的各项应用；结合软件工程的思想，利用面向对象的技术和相关工具，主要完成了以下工作：

首先以命令行方式单独实现了基于仿真串口的三个应用：拨号网络，传真和局域网访问。之后在此基础上，本文根据软件工程的要求，运用面向对象的分析和设计方法进行了集成系统的开发工作；工作分为两个阶段：

第一阶段，利用面向对象的思想和UML语言对集成系统进行了需求分析，并依据分析结果构建了集成系统软件模型。在模型设计中，我们将集成系统从顶层分解为UI(User Interface)与BOL(Business Object Layer)两个包，重点完成了BOL包逐层细化的设计过程。贯穿与分析与设计全过程，我们采用了结构化的设计方案，从而减少了UI与BOL之间，以及BOL包内各功能模块之间的耦合度，增强了各模块的内聚性和可重用度。

第二阶段，完成了集成系统模型中BOL包大部分和UI包部分模块的实现工作，主要包括：BOL中Profile Server包、Profile Client包、Bluetooth Device包和Virtual Device包以及UI中设备管理、传输层配置和PNP硬件检测模块。此外为配合核心程序运行，笔者设计并实现了包括安装程序、手工卸载程序以及虚拟设备管理模块在内的集

成系统支撑模块。

目前笔者参与开发的桌面集成系统 Bluelet Software Suite 的评估版已经向一些主要的分销商发布，反馈结果表明该软件以其强大易用和稳定的性能在世界同类产品中占有一席之地。同时也暴露了系统的一些不足，从外部特性看主要是系统集成的应用还不够丰富；设备管理功能对初级用户而言相对复杂；以及与其他同类产品互通性不好。从内部实现看主要包括传真应用 GW 端转发程序尚不够完善；设备管理机制有待进一步简化；安装程序异常保护机制不健全等。这些缺陷与不足有待于后来者提出更合理的设计和实现方案去解决。

6.2 展望—蓝牙面临的挑战与机遇

作为一项新兴的短距无线通信技术，近年来蓝牙技术获得快速的发展的同时也暴露出了许多不足，在 Monte Carlo 举行的“蓝牙 SIG2000 年会”上提出的阻碍蓝牙技术发展与应用的几个问题仍未得到有效解决[10]，主要包括：

（1）蓝牙产品的价格是很关键的问题，因为只有较低的价格，才能站稳市场。（2）ISM 波段的 RF 问题，在一段时间内，还将会是蓝牙行业的棘手问题，因为目前有两个主要国家（ 西班牙和法国），还限制使用 ISM 波段，而且所使用的频谱定义还多样化。（3）蓝牙行业还必须做些工作， 以消除消费者的忧虑。日常生活中到处是高频传送波，会对人们的健康产生危害。（4）互通问题（互操作性）还不完善，不同制造商的蓝牙组件之间互通还有问题。

另一方面，蓝牙还面临着其他同类技术的挑战。 IEEE802.11 是当前比较热门的短距无线技术，它只规定了开放式系统互联参考模型（OSI/RM）的物理层和 MAC 层，其 MAC 层利用载波监听多重访问/冲突避免（CSMA/CA）协议，而在物理层，IEEE802.11 定义了三种不同的物理介质：红外线、跳频扩频方式（FH）以及直扩方式（DS）。IEEE802.11 支持 1--11Mb/s 较高的数据速率，但是它的成本比较高而且功耗比较大。

另一种无线局域网技术 HomeRF 是专门为家庭用户设计的。HomeRF 利用跳频扩频方式，通过家庭中的一台主机在移动数据和语音设备之间实现通信，既可以通过时分复用支持语音通信，又能通过载波监听多重访问/冲突避免协议提供数据通信服务。

总的来讲，802.11 比较适于办公室中的企业无线网络，HomeRF 可应用于家庭中的移动数据和语音设备与主机之间的通信，而蓝牙技术则身兼二者之长，可以应用于任何可以用无线方式取代电缆的场合。目前这些技术还处于并存和竞争的阶段，现实应用中有可能引起干扰等问题。从长远看，随着产品市场的不断发展，它们将走向融合，而其中最具竞争力的主导技术仍然是蓝牙。

参考文献

[1] "Specification of the Bluetooth System--Core" v1.1 Feb. 22. 2001

[2] "Specification of the Bluetooth System--Profiles" v1.1 Feb. 22. 2001

[3] Liu Guanglei, Qiu Zhengding, "Implementation of Bluetooth Serial Port Based Profiles in Embedded Systems", SPIE APOC 2001, Nov. 2001

[4] Bertrand Meyer ,"Object-Oriented Software Construction" second edition, Prentice Hall,1999

[5] 郑人杰，殷人昆，陶永雷，"实用软件工程"第二版，清华大学出版社，1998

[6] 宋茂强等，"通信软件设计基础"，北京邮电大学出版社，2001.10

[7] Linda M. Northrop, "面向对象的软件开发", www.21cmm.com

[8] AU-System, "Bluetooth White Paper 1.1", www.ausystem.com

[9] James Kardach, "Bluetooth Architecture Overview", *Intel Technology Journal*, Q2, 2000

[10] 陈忠艺，"蓝牙技术的研究及其在Win98上的实现" 北京邮电大学硕士论文，2001

[11] Douglas E. Comer, David L. Stevens, "Internetworking With TCP/IP (Second Edition, VOL 3)", Prentice Hall, 1996

[12] Microsoft Windows Network Architecture, MicroSoft MSDN 2001

[13] 陈坚，孙志月，"MODEM通信编程技术"，西安电子科技大学出版社，1998

[14] M. Billinghurst, T. Stamer, "Wearable Devices – New Ways to Manage Information", Computer, Vol.32, No.1, IEEE, Jan, 1999

[15] Markus Albrecht, Matthias Frank, Peter Martini, "IP Services Over Bluetooth: Leading the Way to a New Mobility", 2000

[16] Wendy Boggs, Michael Boggs, "UML with Rational Rose", 邱仲潘等译，电子工业出版社, 2000

[17] Rational Software Corporation, "Unified Modeling Language for Real-Time Systems Design", Rational Software Corporation, June 2000

致　谢

三年的学习与生活中，我得到了周围许多师长、同学与朋友的帮助，他们的热情指导与帮助令我终生难忘。值此论文完成之际，我衷心的感谢所有曾经帮助过我的人们。

我的论文得以顺利的完成，首先要深深地感谢我尊敬的导师裘正定教授，他在研究生期间给了我许多的宏观指导，并为我创造了便利的实践条件，没有裘老师的全力支持就没有本文的成果！

在课题研究与开发过程中，中科院软件所的高强研究员和项目组负责人徐菲女士给我许多理论与实践上的指导，让我受益匪浅。我还要感谢陈忠艺硕士，曾永平硕士，吴鹏硕士，夏万强硕士、罗洪波硕士以及林鸿、何刚两位博士，他们在开发工作中与我并肩作战、给我无私帮助，直接促成了研发工作的顺利完成。

感谢所有一起工作的其他同学和同事：叶东翔、肖炜、叶可、杨涛、卜洁和李华胜等，与他们一起学习和生活给我留下了非常愉快的回忆。另外还要特别感谢光波所的童治和魏淮两位博士在论文的撰写和整理阶段给我的大力支持。

最后，我要深深地感激我的父亲母亲，感谢他们多年来对我学业和生活默默无闻的理解和支持。养育之恩，无以为报，谨以此文献给我敬爱的父母！