华中科技大学

硕士学位论文

一键通用户信息管理系统的设计和实现

姓名：王青山

申请学位级别：硕士

专业：软件工程

指导教师：陆永忠

20041106

摘　要

一键通用户信息管理系统是一个服务器应用程序，通过扩展 ASP.NET 的 HTTP 模块来实现，手机能通过该系统进行群组和列表的管理。在研究一键通相关核心技术的基础上，包括体系结构、信令协议以及服务器和终端的实现方式，设计和实现了该系统。系统结构分为 5 大模块，消息入口模块、鉴权模块、请求处理模块、日志模块和数据访问模块。其中日志模块和数据访问模块设计成系统中的横向模块，为其它三个模块提供日志记录、访问数据库和生成可扩展标记语言的文件的服务；消息入口模块、鉴权模块和请求处理模块是系统的逻辑部分，分三个步骤依次处理用户发来的请求，实现控制的分离。

消息入口模块完成对用户的请求的检查，并解析该请求和鉴权头消息；鉴权模块负责对用户进行鉴权，主要采用摘要访问控制验证策略，计算用户密码的哈希散列值，用于网上传输；请求处理模块实现对用户的请求的处理，调用数据访问层，根据返回结果，发送不同的处理码和描述信息给用户；日志模块分为安全日志、调试日志和跟踪日志；数据访问模块利用存储过程访问数据库，返回错误码或者可扩展标记语言文件给调用者。由于鉴权模块和请求处理模块的设计相对比较复杂，在详细设计中，分别采用了策略模式和工厂方法模式，提高了系统的扩展能力和灵活性。

关键词：一键通，可扩展标记语言，设计模式

Abstract

PTT(Push-To-Talk) user's information management system is a server application program, which is realized through expanding HttpModule of ASP.NET. The cell-phone can carry on one group of groups and management that tabulates through this system. This system is designed and realized based on studying PTT relevant key technology, including architecture, signaling, and implementations of server and terminal. The systematic structure is divided into 5 big modules, IMServer module, authorization module, action module, log module, data access module. Among these modules, log module and data access module offer the service that write down log, visit the database and produce XML(Extensible Markup Language) file for other three pieces of the system. IMServer module, authorization module are logic part of the system. The request is dealt with through these three steps one by one. Thus, the system realizes the separation of control.

IMServer module finishes inspection of request from users, and analyzes information of request and authorization. Authorization module checks the user by digest access schema which calculates the hash value of user's password. The value will be transmitted on the net safely. Action module realizes the treatment to users' request by visit data access module. According to the result of returning, action module will send different processing yard and describe information to users. The log module is divided into safe record, debugging record and trace record. Data access module utilizes storing process to visit database which return error code or XML file. Because the authorization module and action module are more complicated than others, they are designed by strategy pattern and factory method pattern which can improve systematic expansion ability and flexibility.

Key words: Push To Talk, Extensible Markup Language, Design Pattern

1 绪论

1.1 课题背景

随着 3G 网络建设的临近，移动运营商对于新业务的需求也日益迫切。现有的 GSM/GPRS（Group Special Mobile/General Packet Radio Service）和 CDMA（Code Division Multiple Access）网络的运营经验已证明：2G/2.5G 的移动网络已经能够满足用户对于窄带业务的需求。而 3G 网络要想吸引用户，就必须提供能够充分利用 3G 网络带宽的优势，有足够吸引力的宽带业务。在这种情况下，PTT 业务自然引起了业界的广泛关注。

PTT 业务又称一键通，是一种实现了 Walkie-Talkie 功能的移动话音业务。北美运营商 Nextel 的经验证明：PTT 业务可以有效地提高 ARPU，降低用户转网率，同时对传统话音业务的业务量几乎没有影响。这种特点使得 PTT 成为令运营商欢迎的业务，也因此成为近年来业界共同关注的技术热点。

手机实现 PTT 的概念可追溯于 1995 年，美国摩托罗拉公司在 iDEN（integrated Dispatch Enhanced Network）集群系统上推出 PTT 对讲解决方案。一年后，美国的 Nextel 通信公司在其 iDEN 集群网络上开始提供名为 Direct Connect 的相关业务，该业务的最大特点是用户可以仅按一键即可快速而方便地即时通话。

Nextel 公司也因高 ARPU 值和其低用户流失率一直引起同行注目，而业界普遍认为这在很大程度上要归功于其基于 PTT 业务的成功应用，这在企业市场中尤为如此。Nextel 的 90%以上的新用户都是冲着 PTT 而来的。

但是，由于 Nextel 经营的是一个集群共网，其网络的特殊性决定了此项业务移植到蜂窝网上的难度。虽然美国各大移动运营商纷纷觊觎其 PTT 业务，然而在蜂窝通信网络上推出 PTT 业务，无论对于设备提供商还是运营商都是一个完全的新课题。一些有远见的设备提供商自二十世纪末以来一直在探索 PTT 于 2.5G 和 3G 蜂窝移动网络的可能性，并且已经取得了突破性的成果。

Nextel 公司的 PTT 服务在部分市场获得很大的成功，但到目前为止，接受这一服务的主要是“蓝领”和“小型企业”的消费层。造成这一现象的主要原因是由于

这一服务只在提供该服务的运营商的网络上才能实现，这样就失去了那些主流网络，如 CDMA 和 GSM 上的用户。而且，即使在 Nextel 自己的网络上，这一服务的提供也受到地域的限制，这就给需要出差的人造成了很大不便，因此这一服务存在巨大的空间可以提高完善。除了原有的集群通话业务的消费层，我们相信，PTT 也能吸引那些职业经理人、青少年和家庭消费者。

在移动市场逐步饱和与竞争日趋激烈的情况下，美国各大移动运营商也开始关注 PTT 技术并积极投身试验，如 Sprint PCS 公司目前正在进行 CDMA 1X 网络上 PTT 业务的试验，Verizonb Wireless 公司计划在 2003 年第三季度启动 PTT 试验，美国 AT&T Wireless 也已经宣布将在 2003 年第 4 季度进行 GPRS 网络上该业务的试验，但为了竞争，各公司都避而不谈其详细部署计划，一切似乎都在秘密进行之中。

对于 PTT，中国的移动运营商也很关注，积极推动开发。华为、海信等企业已开发出商用版，处在小规模试用阶段。

1.2 课题来源

本课题来源于在华为技术有限公司无线核心网的 PTT 项目。该项目主要开发后台服务器，为移动供应商提供技术服务；前台手机由手机商协调开发。本人主要负责 PTT 用户信息管理系统的开发，用于提供用户群组和列表的管理服务。

1.3 研究内容

本论文主要研究了 PTT 的设计方案，在此基础上，设计和实现了 PTT 用户信息管理系统。用户管理子系统主要涉及到用户数据的处理，例如，群组、好友等的表示方式和传输方式。这一块主要要用到 XML 的知识来定义数据格式；另外一个是服务器平台的选择，可以实现自己的服务器，也可以才用现有的服务器平台进行二次开发。我们选择了 ASP.NET，需要对扩展 HTTP 模块进行深入的研究；在系统设计过程中，需要考虑系统灵活性，研究了工厂模式和策略模式，并应用到系统架构中。通过实际开发，学习软件工程开发流程，掌握实际项目的开发，也是一个重要的研究方面。

2 PTT 技术基础

PTT 中文翻译为“一键通”，本章首先介绍了 PTT 的技术定位和相关标准和规范；然后介绍和分析了作为支撑技术的 XML 和 ASP.NET。

2.1 PTT/POC 的技术与标准

从技术上说，PTT 业务是一种应用于一对一通信和群组通信的半双工的移动通信业务[1]。PTT 业务的用户体验类似对讲机,呼叫方无需拨号，只要按下终端上的 PTT 键就进入了呼叫状态，可以立即发起对预定义个人或群组的呼叫;PTT 键弹起后就进入了接收状态，来话无需振铃即可自动播放。PTT 业务同一时刻只允许一个人处于呼叫状态，系统通过判断各 PTT 用户按下 PTT 键的先后顺序以及预定义的优先级来决定呼叫权的分配。

2.1.1 PTT/POC 的技术定位

PTT 业务的概念起源于集群通信系统，在模拟集群通信系统和数字集群通信系统上都能够提供 PTT 业务[2,3]。

本文所研究的 PTT 主要是基于蜂窝移动通信网络的 PTT 业务，称为 PTT Over Cellular(POC)。

POC 基于 2.5G 网络（GSM/GPRS，CDMA 1X）或 3G 网络（WCDMA，CDMA 2000），它充分利用了 GPRS 或 CDMA 1X 移动分组网络的特性，通过半双工 VoIP 技术来实现 PTT。同时，POC 还结合了即时消息，Presence 等业务属性，成为一种综合了话音和数据的个性化业务。

未来 PTT/POC 还将与基于存储转发的非实时的 PTM（Push-To-Multimedia）相结合，最终成为 PTC（Push-To-Connect）。其中，PTM（也称 P2M）是指通过按键发送非实时语音消息，MMS 等多媒体消息的业务。在技术上 PTT 与 PTM 的主要区别是 PTT 使用基于 UDP/IP 的实时传送技术，而 PTM 使用基于 TCP/IP 的可靠的存储转发技术。

2.1.2 PTT/POC 的标准与规范

PTT 自从提出以来一直是某些厂商的私有技术，也是某些运营商的私有品牌[5]。这种封闭的技术体制导致了业务互通性很差，除非不同运营商都采用同一厂商的 PTT 设备，否则它们之间的业务互通极难实现。近年来提出的 POC 使 PTT 可以在 GSM/GPRS 和 CDMA 1X 网络中实现[6]，当 POC 有望成为广泛部署和使用的业务时，POC 技术的标准化便成为迫在眉睫的任务。开放的 POC 标准有助于减少市场分割，保证各厂商 POC 系统间的互联互通。

在 POC 的工业标准和国际标准颁布之前，各厂商对 POC 自由解释，提出了很多私有的解决方案，这些方案大部分都是基于分组交换域的，即面向现有的 2.5G 网络，通过在 GPRS 或 CDMA 1X 核心网上直接增加 POC 服务器来实现 POC 业务，因此可以称为“基于 PS 域的方案”。

工业标准方面 2003 年 8 月，Ericsson, Motorola、Siemens 和 Nokia 四大厂商联合推出了 POC V1.0 系列规范，包括 POC 的“Architecture”、“User Requirements”、“Signaling Flows”、“User Plane; Transport Protocols”、“User Plane、 (E)GPRS/UMTS Specification”、“List Management and Do Not Disturb”。共 6 个技术规范。这些规范基于 3GPP（Third Generation Partnership Project）和 3GPP2 定义的 IP 多媒体子系统（IMS），强调系统的开放性和标准性。目前，这四家公司正在积极活动，与 OMA、3GPP 等国际组织合作，谋求使该规范成为国际标准，并使 POC 业务纳入多媒体子系统的业务中[7]。

国际标准方面，移动开放联盟（OMA）于 2003 年 4 月正式成立了 POC 工作组。POC 工作组在工业标准 POC V1.0 的基础上进行修改和补充，已经发布了 POC 体系结构规范的草案。不过，OMA 目前还没有正式发布任何技术标准。

POC 国际规范分为很多方面：OMA 制订的主要是 POC 需求、业务体系结构、互操作技术规范和测试规范；3GPP 和 3GPP2 制订的 IMS 技术体制实际上是 POC 国际规范的网络基础；而具体信令则大量使用了 IETF（Internet Engineering Task Force）定义的协议，如 SIP（会话发起协议）、SIMPLE、RTP 等[8]。

无论工业标准 POC V1.0 还是 OMA 制订的 POC 国际标准都是基于 3GPP 和

3GPP2 的 IMS 域的，仅有的不同是 POC V1.0 基于标准的 IMS，而 POC 国际规范则只要求“基于具有 IMS 能力的 SIP/IP 核心网”。然而，这些方案大体上都可以称为“基于 IMS 域的方案”[9]。

POC 国际规范对工业标准 POC V1.0 的修改主要将集中在以下方面：

（1）增加了 POC 网络内部的接口，如 POC 服务器之间的接口；

（2）纠正了少数不符合 SIP 协议的信令流程；

（3）建议由 IETF 定义呼叫权控制（Floor Control）的协议，并由 OMA 修改；

（4）使呼叫权控制的协议头和协议体的压缩符合 CDMA 2000 中对突发数据的定义；

（5）定义一个安全方案；

（6）由 POC 服务器用 SIP/SIMPLE 来实现在线状态的“免打扰”功能；

（7）解决 SIP/UDP 导致的会话连接不稳定的问题；

（8）了引自 3GPP 的标准以及相关的描述不清的部分；

（9）同时支持 IPV4 和 IPV6；

（10）纠正呼叫权控制的一些错误；

（11）计费结算部分增加 AAA 机制。

2.1.3 PTT/POC 的客户端和服务端

主要分为服务器端和客户端两个部分：

（1）服务器端技术

无论在基于分组核心网的方案中还是在基于 IMS（IP Multimedia Subsystem）的方案中，PTT/POC 服务器都起着关键的作用。从内部实现技术的角度看，POC 服务器有两种实现方式，综合业务平台方式和软交换方式[10-12]。

在综合业务平台方式中，PTT/POC 服务器同时负责呼叫控制和媒体处理，内部各模块耦合紧密。这种方式可以使用纯软件实现，也可以基于特定媒体处理硬件板卡的 API 开发。综合业务平台方式一般适用于较小容量的系统，当系统容量持续扩大时，由于呼叫控制和媒体处理耦合紧密，升级和扩容不够灵活，因此成本较高[13,14]。

软交换方式采用了呼叫控制与媒体处理相分离的软交换技术[15]，PTT/POC 服务

器主要分为呼叫控制模块与媒体处理器模块，二者之间采用标准的 H.248 协议。呼叫控制模块用软件实现，而媒体处理模块可用软件或硬件实现，硬件实现将能极大的提高媒体处理的性能。这种分层的体系结构降低了系统的耦合度，增强了稳定性。由于采用开放的协议，因此开发者可以专注于呼叫控制及其它业务层模块的开发，并通过外购媒体处理模块的方式缩短开发周期，提高产品质量。另一方面，当产品商用后，运营商可以根据当前系统的处理瓶颈对媒体处理器或呼叫控制分别进行升级和扩容，极大地提高了系统的可扩展性，降低了系统的成本[16,17]。因此，软交换方式特别适用于频繁升级或容量大的 PTT/POC 系统。

（2）终端技术

终端支持 POC 业务有两种方式，第一种方式是硬件方式，即终端厂商在其手机芯片中设计实现 POC 功能。这种类型 POC 终端的特点是速度快，性能好，但由于 POC 规范尚未完全确定，所以可能会存在兼容性问题，而且用户必须更换手机，这在一定程度上形成了终端的瓶颈[18-20]。

另一种方式是软件方式，即终端上安装了主流的移动终端操作系统或执行环境，用户可下载和安装专门的 POC 软件。当前主要的移动终端操作系统或执行环境包括：Symbian、MS Smart Phone、Pocket PC、Palm、Brew 等，支持的手机终端包括 Nokia7650、Nokia3650、Nokia6600、Nokia N-Gage、索尼爱立信 P800、P908、Siemens SX-1、Motorola A920、Mitsubishi Concept、BenQ P30 等大部分智能手机终端[23]。软件方式不需更换手机，安装方便，便于修改升级，但也存在速度慢，性能不好的缺点。

2.2 XML 技术

XML 称为可扩展标记语言，伴随着互联网的出现而发展起来。到目前为止，XML 在各种领域得到了广泛的应用，例如在数据交换和数据整合领域，媒体无关的数据发布的领域等[24]。

2.2.1 XML 的特点

XML 是 Extensible Markup Language(可扩展标记语言)的缩写，是 W3C 组织于

1998 年 2 月发布的标准。W3C 采用了简化 SGML（Standard Generalized Markup Language）的策略，在 SGML 基础上去掉语法定义部分，适当简化 DTD（文档类型定义）部分，并增加了部分互联网的特殊部分。因此，XML 也是一种标记语言，基本上是 SGML 的一个子集。因为 XML 也有 DTD，所以 XML 也可以作为派生其它标记语言的元语言。

XML 文档最显著的特点是信息的描述与信息的处理是分开的，这也正体现了标记语言的精髓[25-27]。XML 文档的主要任务就是定义内容本身，它保持用户接口与结构化信息的独立。在 HTML 中，标注可以用来告诉浏览器将信息显示成粗体或斜体；而在 XML 中，对信息的处理和显示是用来描述数据本身，如书名，作者等。在 XML 中，对信息的处理和显示是通过样式实现的。将信息和对信息的处理隔离开来，使得多方来源的信息可以完全整合，让所有信息在中间层次转换成 XML（或者已经是 XML 格式的），然后在线交换并对信息进行各种处理，最后根据用户的需求和能力以特定格式将信息提交给用户。

XML 文档具有自我描述能力。在 XML 文档中，标记本身就是对被标记信息最好的说明。比如包含在标记<作者></作者>之间的，肯定就是某著作和文档的撰写人的名字[30,31]。再加上通过 XML 文档可以获得的 DTD 中定义的可以使用的标记以及标记之间的结构关系等各种规则，这样，即使在只有 XML 文档的情况下，计算机也能理解文档中的信息。

XML 具有很强的开放性和可扩展性。在 XML 中，可以定义无限量的标记。它提供了一个标记结构化信息的架构，用户根据自己的需求可以随时定义新的标记，这就大大拓展了可用标记的范围[33,34]。

XML 文档具有可验证的特性。利用 XML 文档对应的 DTD，可以验证 XML 文档中数据在结构上的正确性和标记使用的规范性，因为在制订 DTD 时一切都已定义好了。这样，我们在数据形成阶段，而不是数据使用阶段就能对数据的有效性和正确性进行部分检验，提高了数据的可靠性和可用性。

2.2.2 XML 的应用领域

作为互联网的新技术，XML 的应用非常广泛，可以说 XML 已经渗透到了互联

网的各个角落。大致来讲，XML 的应用可以分为以下几类[35-37]：

（1）设计标记语言

作为元标记语言，XML 为用户提供了针对自己的特定需求定义本行业本领域的标记语言的最好工具。目前这一应用的成功例子比比皆是，例如化学领域的标记语言，数学领域的标记语言，移动通信领域的标记语言等。

（2）数据交换和数据整合

数据交换无疑是 XML 最令人激动的应用。数据交换的核心问题是信息描述的标准化，主要解决信息的可理解性问题，包括人和机器对信息的理解。而且，更重要的是机器对信息的识别，并能根据数据进行自动处理。由于 XML 描述数据的非凡能力和对数据描述格式的一致性，使得它逐渐替代传统的 EDI（Electronic Data Interchange）成为电子商务得核心基础技术。电子商务就是利用电子手段尤其是互联网进行商务活动。从技术上说，电子商务是通过互联网传输和交换商务数据，并能根据商务数据进行人工或自动处理。XML 的可扩展性和自相容性等特点，使它成为数据交换的有力工具。

在客户需要与不同的数据源进行交互时，数据可能来自不同的数据库，有各自不同的复杂格式。但客户与这些数据库间只通过一种标准语言进行交互，那就是 XML。由于 XML 的自定义性及可扩展性，它可表达各种类型的数据。客户收到数据后可以进行处理，也可以在不同的数据库间进行传递。总之，在这类应用中，XML 解决了数据的统一接口问题。但是，与其它的数据传递标准不同的是，XML 并没有定义数据文件中数据出现的具体规范，而是在数据中附加标记来表达数据的逻辑结构和含义，这使 XML 成为一种程序能自动理解的规范。

（3）媒体无关的数据发布

媒体无关的发布实际上是一个比数据交换要难得多的问题。可以说，一般意义上的发布需求是数据交换需求的超集。通过 XML 发布可以将同一数据以不同的面貌展现给不同的用户。我们所要做的只是尽善尽美地将数据本身描述好，然后通过 XSL 地转换将它发布到各种媒体上，不管是显示器，打印机，无线设备还是盲人阅读设备。就好像同一个剧本，我们却可以用电视剧，电影，话剧，动画片等不同形

式表现出来。这一应用将会为网络用户界面个性化，风格化的发展铺平道路。随着互联网的飞速发展，互联网已成为继报刊、广播、电视之后的第四媒体。网络出版自从出现以来，用于信息发布的主要是 HTML 技术，但是这种方式在跨媒体出版时遇到了极大的困难，因为人们需要为不同媒体制作不同版本。XML 具有内容与显示分离的特点，人们可以一次性制作内容，配以不同的样式单，实现一次制作多次出版。

（4）智能代理和本地计算

如果数据是结构化的 XML 数据，基于 XML 文档的自描述特性，智能代理就能够非常容易地利用已有的知识库理解数据，然后做出相应的反应。XML 的出现，为智能代理的发展提供了新的推动力，智能代理能够对所取得的信息进行编辑，增减以适应用户的需要。

利用 XML 文档能够将大量运算负荷分布在客户端，即客户可根据自己的需求选择和制作不同的应用程序以处理数据。如果按传统的 C/S 或 B/S 工作方式，用户向服务器发出不同的请求，服务器分别予以响应，这不仅加重服务器本身的负荷，而且网络管理者还须事先调查各种不同的用户需求以做出相应不同的程序。但用户的需求繁杂而多变，将所有业务逻辑集中在服务器端是不合适的，因为服务器端的编程人员可能来不及满足众多的应用需求，也来不及跟上需求的变化，这样的处理方式使双方都很被动。应用 XML 则将处理数据的主动权交给了用户，服务器所作的只是尽可能完善，准确的将数据封装进 XML 文件中，正是各取所需，各司其职。XML 的自解释性使客户端在收到数据的同时也理解数据的逻辑结构与含义，从而使广泛、通用的分布式计算成为可能，也使真正的个性化服务变得可行。

（5）精确搜索

现在 Internet 上得信息使浩如烟海。虽然有各种搜索引擎，但他们得搜索效果实在难以令人满意，结果中包含大量得无用信息。比如我们想搜索莎士比亚得作品，返回得结果几乎使与这有关得所有信息。究其原因，不仅仅在于搜索引擎本身，也在于信息得描述方式。因为搜索过程中，搜索引擎要分辩网络上那些莎士比亚字段

对应得使作品名，哪些是关于莎士比亚得介绍、传记等。

如果信息是 XML 格式得，那么情况将大为改观。XML 文档中得标记很好得说明了各段信息的意义，标记之间得层次关系很好地表达了信息地结构。利用 XML 文档地这种自描述特性，能够有效提高信息检索地效率和准确度。

（6）文件保值

XML 良好地保值性和自描述性使它成为保存历史档案地最佳选择。随着技术地发展和时间地流逝，XML 在这方面地意义将越来越大。在新的计算机系统中，很多格式的文件早已不支持了，就像今天我们要读取用 WordStar 生成的文档所面临的问题那样。

2.2.3 XML 带来的影响

XML 作为一种技术或者思想，对整个 IT 行业带来的影响将是革命性的，它将深刻改变人们描述，获得和使用信息的全过程。概括来说可以包括一下几个方面[38]：

（1）使通用的信息描述成为可能

如果我们仔细考察一下今天的 Web，会发现有很多通用的技术，比如 TCP/IP 协议就是通用的链接协议，任何支持 TCP/IP 的设备之间可以实现互通。XML 具有开放性和描述复杂数据的能力，使得各种信息能够通过 XML 这种格式来进行统一的描述，建立有任何复杂层次的数据模型。现有的信息也可以转换成 XML，这一切将促使 XML 成为一个通用的信息描述平台。

（2）信息发布的真正独立

如同 Java 的一次编译，到处运行一样，XML 的出现和广泛采用将实现信息的一次描述，到处使用。由于 XML 真正实现了信息描述和信息处理的分离，在信息形成的过程中，无需考虑用户对数据可能进行的处理和数据的最终表现形态，这样，信息的发布者将获得完全意义上的独立，他们能更好的关注信息本身，使信息的描述尽善尽美。

（3）独立于供应商和平台的信息交换与整合

XML 作为通用的信息描述手段，采用的文件格式是最简单的文本格式，使用标

准的商业软件甚至文本工具都能处理，因而 XML 将大大简化信息交换的过程。交换双方只要就使用什么标记来标注信息到成共识（即通过同一个 DTD）就可以方便的进行数据交换，无需考虑对方的后台系统是如何实现的，唯一要要做的是使用同一个标记集合。

（4）更符合用户意图的信息搜索

以图书搜索为例。图书信息用 XML 标注以后，用户在搜索时就能指定作者为莎士比亚的书籍，而非有关莎士比亚的书籍。相较之下，使用目前的方法搜索，可能找出两类书籍，并且混在一起。没有 XML，搜寻应用程序需要解读每一个数据库的 Schema（描述其组建方式），这几乎不可能，因为每一个数据库表示资料的方式都不相同。使用 XML，书籍信息可以轻松的按照作者，书名，ISBN（国际标准书号）号码或者其它准则归类。XML 文档中的标记很好的说明了各段信息的意义，并表达了文档的结构。基于 XML 文档的这种自描述性，计算机能够理解 XML 文档本身。于是在信息检索过程中，计算机能够更加贴切的理解用户的检索意图，从而有效提高信息获取的效率。

（5）信息的本地化处理

当 XML 格式的信息传递到客户端以后，客户端应用程序可以就地运算，编辑和运作，可以使用各种方式运用资料，而非只是显示。由于用户能够理解数据并且有数据处理的主动性，原先集中于服务器端的大量运算负荷可以分布到客户端，大大减轻服务器的压力。

2.3 扩展 ASP.NET

ASP.NET 为我们提供了一个应用服务平台，可以基于它快速开发出服务端应用程序，节省开发时间和成本。下面介绍了一种对这个平台的一个利用，研究了扩展 HTTP 模块原理和实现，从而设计了系统架构。

2.3.1 HTTP 运行时研究

万维网(WWW, World Wide Web)使用超文本传输协议(HTTP, Hypertext Transfer Protocol)作为通信的基本协议。HTTP 是一种应用层协议，它主要负责在客户机(浏览

器)和 Web 服务器之间建立连接和传送客户机请求的信息。实际上，现在的 Web 服务器也要使用 HTTP 协议接收来自客户机的请求并向客户机发送响应。

ASP.NET 将客户机请求发送到用户定义的 HTTP 处理程序对象，称这类对象为“HTTP Handles”。运用 ASP.NET，可以通过实现称为 IHttpHandler 的.NET 接口创建这些用户定义的 HTTP 处理程序。在创建了用户定义的处理程序之后，可以在处理程序上绑定一个指定的 URL 请求，以便处理特定的请求。例如，可以绑定一个请求文件的 URL，指定文件的扩展名，让用户定义的处理程序进行处理。然后，如果某个指定的 URL 请求与处理程序不匹配，那么 ASP.NET 的默认处理程序将处理该请求。

在浏览器中输入一个 URL 地址时，浏览器将创建一个 HTTP 请求并将其发送到 URL 指定的地址。在创建 HTTP 请求时，使用了各种方法。这些方法指出了请求的目的所在。这些方法包括下列几种：

Get：当请求某个特殊的页面时使用该方法。当用户在浏览器的 Address(地址)框中输入一个链接或者单击某个超链接时，就使用 HTTP 的 Get 方法创建 HTTP 请求。通常在请求不更改数据库状态时使用 Get 方法。

Head：当用户希望只检索有关文档的信息而不是文档本身时使用该方法。

Post：当用户请求与数据库交互的某项资源时使用该方法。

包含被请求页面的 Web 服务器基于用于发送请求的方法执行必要的处理并返回客户机请求的页面。除了上述这些方法以外，还可以用更底层的控制来处理 Web 服务器上的请求。此时，可以借助应用程序编程接口(API)。

目前已经开发出很多 API，使开发人员能够在更底层控制对 Web 服务器上的请求的处理。例如，为 IIS Web Server 开发的因特网服务 API(ISAPI，Internet services API)使得开发人员能够创建高性能得应用程序。同时，也可以在更底层控制 IIS 处理请求的方式。

有了 ISAPI，可以创建自己的动态链接库，用它们来指定请求发送到 Web 服务器时必须执行的任务。ISAPI 提供的动态链接库有两种类型即“过滤器”和“扩展”。过滤器用于编写代码以便在处理请求的过程中接收来自 Web 服务器的通知。

因此，过滤器用于更改 Web 服务器的标准行为。使用过滤器执行的任务包括压缩和加密要发送的数据以及对用户进行验证等等。而 ISAPI 扩展接受用户请求，执行诸如接收来自数据库的数据和生成 HTML 页面这样的任务并且向客户机发送响应。

在 ASP.NET Web 应用程序中，使用 HTTP 运行时获得对客户机请求的底层控制权。HTTP 运行时是建立在.NET 框架的通用语言运行时(CLR, Common Language Runtime)基础上的，它提供了处理请求的环境。因此，在 IIS 中用 CLR 替代 ISAPI。HTTP 运行时执行各种功能，包括接收来自客户机的请求，解析 URL 中指定的地址以及为了进一步处理请求而向相应的应用程序发送请求。HTTP 运行时有能力同时接收多个请求。在不同的地址空间运行各应用程序，从而提高了可靠性并避免了跨平台的混乱性。因此，单个 Web 应用程序的失败并不影响 HTTP 运行时的运转。

就像 ISAPI 扩展和 ISAPI 过滤器一样，HTTP 运行时使开发人员能够对 Web 请求的处理实施更底层的控制。然而，与 ISAPI 不同的是，ISAPI 需要开发人员必须懂得 C++语言，HTTP 运行时是一种更简明的模型，它使开发人员可以用任何.NET 程序设计语言进行编程。因此，相对 ISAPI 体系结构来讲，.NET 框架的 CLR 更简单。

HTTP 运行时的体系结构类似与流水线的体系结构。它由多个 HTTP 模块和处理程序组成。简而言之，HTTP 模块和 HTTP 处理程序都是由实现 ASP.NET 预定义接口的开发人员创建的类。当客户机发出执行 Web 应用程序的请求时，该请求就会通过流水线时式的各个 HTTP 模块。HTTP 模块能够使 Web 应用程序执行特殊的任务，比如加密数据，执行提供应用程序访问权限的自定义验证以及管理客户会话和应用程序的状态等等。在通过一系列的 HTTP 模块之后，将把请求送往 HTTP 处理程序。HTTP 处理程序可以代替 ISAPI 扩展接收请求，提取所需数据并向发送请求的客户机发送响应数据。ASP.NET提供了更高级的程序设计模块，比如Web服务和Web Forms，它们都作为 HTTP 处理程序执行。HTTP 运行时的流水线体系结构使得通过添加新的 HTTP 模块和处理程序就能很容易地实现新地功能。

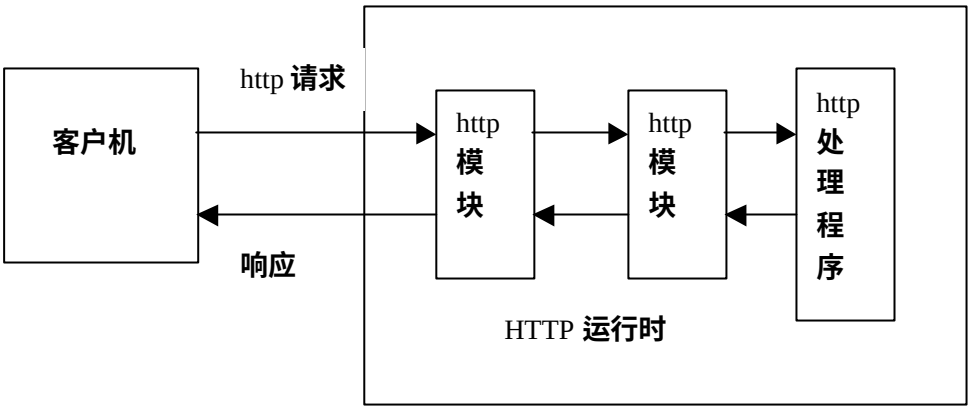

图 2-1　ASP.NET **提供的** HTTP **运行时的流水线体系结构**

ASP.NET **提供了实现** HTTP **模块和** HTTP **处理程序创建工作的各种接口。例如，提供的** IHttpModule **接口，使用该接口可创建执行有关安全和压缩任务的模块。在** HTTP **模块中经常实现各种** State(**状态**)**管理函数，这样可以很容易地从** HTTP **运行时流水线中删除或添加各种模块。**

除了 IHttpModule **接口以为，**ASP.NET **还提供了** IHttpHandler **接口，开发人员实现该接口以创建更底层地** HTTP **处理程序，从而接收请求和执行各种任务。**

有时有必要在每次显示 ASP.NET **页时执行某个动作，例如对访问页的用户进行身份验证。可以在** ASP.NET **代码中实现对用户的身份验证，但是如果忘记在页中实现这项功能，在应用程序中就会有安全漏洞。此外，如果正在访问的文件不是** ASP **文件，就不能实现对该文件的安全保护。在** ASP.NET **以前的各种版本的** ASP **都不能解决这个问题。如果需要对每个用户请求都执行一个动作，就必须编写** Internet Server Application Programming Interface(ISAPI)**筛选器。虽然** ISAPI **筛选器的功能很强大，但是不容易编写，而且只能用** C++**编写。在** ASP.NET **中，依靠** HttpModule **的概念解决了这个问题。由于** HttpModule **是** ASP.NET **的一部分，所以可以用**.Net **语言中的任意一种语言编写。任何能编写** ASP.NET **代码的程序员都能在自己选择的编程语言中实现** HttpModule。

2.3.2　扩展 HttpModule 原理

HttpModule **是实现和注册一个类的方法，这个类可以处理在请求的处理过程中**

由 HttpApplication 类产生的事件。当产生一个事件时，HttpModule 可以对请求进行进一步的处理。HttpModule 提供和 ISAPI 筛选器类似的功能，但是更容易设计和实现。由于 HttpModule 是全局对象，因此即使不能完全影响所有的文件，也可以影响应用程序的大多数.aspx 文件。下面列出了可以利用 HttpModule 完成的任务：

（1）批准文件和对用户进行身份验证；

（2）维护给定用户使用的对象集；

（3）在返回 HTML 数据的每一页的下端添加消息；

（4）加密某类页中的数据；

（5）集中应用程序的错误日志。

HttpModule 有两个基本的目的，它为 Application 对象产生的事件提供事件处理程序，以及产生可在 Global.asax 文件中或隐藏在 Global.asax 文件后面的代码文件中处理事件。

（1）编写 HttpModule

虽然由于所要完成的任务不同，有时可能要涉及复杂的问题，但是编写 HttpModule 还是相当简单的。要编写 HttpModule，必须完成以下任务：

①编写一个实现 IHttpMoudle 接口的类。IHttpModule 接口有两个方法：一个 Init 方法和一个 Dispose 方法；

②在编写的类中实现一个方法，处理希望捕捉的事件；

③在类的 Init 方法中添加一项，捕捉希望处理的事件；

④在 web.config 文件或 machine.config 文件的<HttpModule>节中添加一项，注册编写的 HttpModule。为注册一个应用程序的模块，应该在 web.config 文件中添加项。为注册所有应用程序的模块，应该在 machine.config 文件中添加项。

自己编写的模块可以完成简单的任务，也可以完成复杂的任务，但是一定要记住很重要的一点，每个对.aspx 文件的请求都会引发正在处理的事件。如果事件处理方法要花费很长时间才能完成，就会显著影响服务器的性能。

（2）模块的生命周期

HttpModule 都是在第一次启动应用程序时构建的。应用程序保留着对模块的引

用。关闭应用程序时释放该引用。因此 HttpModule 的生命周期差不多和应用程序的生命周期一样了解这一点很重要，这是由于需要确保 HttpModule 不在类范围内存储东西，除非绝对有必要这样做，因为存储的内容直到应用程序结束时才会超出范围。大多数资源应该在事件处理程序开始时分配，而在事件处理程序结束时释放。

（3）处理事件

在处理事件时涉及两项任务。首先，需要实现一个方法来处理事件，然后需要注册这个方法和事件相联系。

实现处理事件的方法就和实现其他任何方法一样，只有一点不同，就是必须匹配事件的签名。源对象通常用做 System.Web.HttpApplication 类型的一个对象。虽然可以把事件处理方法命名成自己喜欢的名称，但是惯例是命名成 OnEventName,这里 EventName 是正在处理的事件的名称。

注册方法和事件一般在 Init 方法中进行。为此，在 Init 方法中添加以下一行代码：

```
context.EventName += new EventHandler(OnEventName);
```

用这种方法可以用任意多个模块捕捉一个事件。如果偶然地捕捉了同一个事件两次，就会对每个事件调用两次方法。

（4）HttpModule 配置

在编写完 HttpModule 后，必须注册，这样当 Web 应用程序启动时可以载入这个 HttpModule。为此，在 web.config 文件或 machine.config 文件的<HttpModule>节中添加一项。在 web.config 文件中添加一项可以使模块在特定的 Web 应用程序启动时载入。在 machine.config 文件中添加一项可以使模块在任何 Web 应用程序启动时载入。<HttpModule>节中项的格式如下所示：

```
<configuration>
    <system.web>
        <httpModules>
            <add name="HelloWorldModule"
                    type="HelloWorldModule, HelloWorldModule" />
        </httpModules>
```

</system.web>

</configuration>

<add>项添加一个新的 HttpModule，<remove>项删除一个继承的 HttpModule，<clear>项删除所有的 HttpModule。在可以使用模块之前需要一个<add>项。如果在 machine.config 文件中添加一个模块，就可以使用 Web 应用程序的 web.config 文件中的<remove>项，在一个特定 Web 应用程序的 web.config 文件中删除这个模块。这样除了在 web.config 文件中有<remove>项的 Web 应用程序以外，可以为每个 web 应用程序载入一个 HttpModule。

（5）HttpApplication 事件

HttpApplication 对象有几个可以由 HttpModule 处理的事件。由于任意个 HttpModule 都可以处理这些事件，因此如果需要自己编写的 HttpModule 是处理给定事件的唯一模块，就需要特别小心。保证其他方法不能处理事件的唯一方法是了解 Web 应用程序注册了其他哪些 HttpModule，以及其他 HttpModule 处理哪些事件。例如，有两个 HttpModule 都处理 AuthenticateRequest 事件，这两个 HttpModule 提供不同类型的身份验证。由于 Web 客户端可能只为一种身份验证提供数据，因此其中一个 HttpModule 总是不能验证请求，请求总是被其中一个 HttpModule 拒绝。

通过在事件处理程序中调用 HttpApplication 的成员函数 CompleteRequest 可以终止一个请求。如果没有调用 CompleteRequest，应用程序就照常继续处理请求。如果编写一个事件处理程序处理 AuthenticateRequest 事件，在身份验证失败时可能需要调用 CompleteRequest。在决定应该处理哪个事件时，需要考虑在每个事件发生之前和之后发生了什么事情以及将要发生什么事情。如果处理了错误的事件，就不能得到预期的输出。如果在请求的处理过程中有未处理的错误就激发 Error 事件。在应用程序最后关闭期间，在调用 HttpApplication 类的 Dispose 方法时激发 Disposed 事件。

2.4 小结

目前，PTT/POC 技术还存在一些尚待解决的技术问题。首先，与基于集群通信系统的 PTT 相比，POC 的呼叫建立时间较长。数字集群通信系统是专为 PTT 类型业

务设计的专用系统，iDEN 系统中手机每秒钟与基站联系一次，而蜂窝移动通信系统的手机则每 6~10 秒与基站联系一次，因此基于集群通信系统的 PTT 的呼叫建立时间为小于 1 秒，而目前 POC 的初次呼叫建立时间则为 3~10 秒（通常为 7~10 秒），后续呼叫如果不超时，则建立时间小于 1 秒。这个初次呼叫建立时间对 PTT 来说显然太长了。

POC 的另一个技术难题是缩短通话延时。集群通信系统（如 iDEN）使用协议栈中较低的层次传送话音，而蜂窝移动通信系统则都是采用协议栈中较高的层次传送话音（即 VoIP 方式），这种差异导致了 POC 的通话延时也明显的大于集群通信 PTT。同样的原因，当 POC 实现跨运营商的互通时，其性能和质量将大幅度下降，这是因为 PTT 是一种对实时性要求很高的业务，互连互通将增加系统的复杂性，从而使时延加剧。

目前业界正致力于解决上述技术问题，并取得了一定的成果。相信在不久的将来以上问题将在很大程度上得到缓解。

另外，对 XML 和 ASP.NET 的也作了针对性的讨论和分析，以应用到系统的实现上。

3 PTT 用户信息管理系统的设计

PTT 用户信息管理系统主要实现了 IM（Interface Management）的所有功能，是一个独立的应用程序实体。主要的研究工作集中在这上面；并对这一块进行了设计和分析。

3.1 PTT/POC 的总体技术实现方案

基于 IMS（IP Multimedia Subsystem）域的方案的主要标准包括由 Ericsson、Motorola、Siemens 和 Nokia 联合制订的工业标准 POC V1.0 和 OMA POC 工作组正在制订的 POC 国际规范。

基于 IMS 域的 POC 方案的特点是互联互通性很好，体系结构庞大而严谨，技术难度高，需要增加 IMS 域实体，对现有网络改造大（各大运营商现有的或拟建的 3G 网络大都是 R99 的，至多是 R4 的，都不包含 R5 的 IMS 域），部署难度大，成本高，是最终大规模商用网的最佳选择。目前该方案还不很成熟，但由于得到了各大厂商的强烈支持，因此发展很迅速。各大运营商对此十分重视，对相关规范保持密切的跟踪[40]。

OMA 提出的 POC 体系结构如图 3-1 所示。其中，黑框代表 POC 功能实体，单实线箭头代表信令接口，双实线箭头代表信令和媒体接口。

POC 所使用的接入网（access network）既包括无 线接入网又包含提供 IP 连接和 IP 移动性的节点。POC 使用基于 IMS 能力的 SIP/IP 核心网来提供寻址，选路，漫游等功能。图 3-1 中各实体的功能如下：

POC 客户端：位于移动终端上，用以接入和使用 POC 业务。

POC 服务器：实现了 POC 业务应用层的网络功能，包括以下方面：

（1）提供 POC 会话（session）处理；

（2）提供媒体的分发（media distribution）；

（3）提供包括成员鉴权在内的呼叫权控制（floor control）；

（4）提供 SIP 会话处理，如 SIP 会话的发起和结束等；

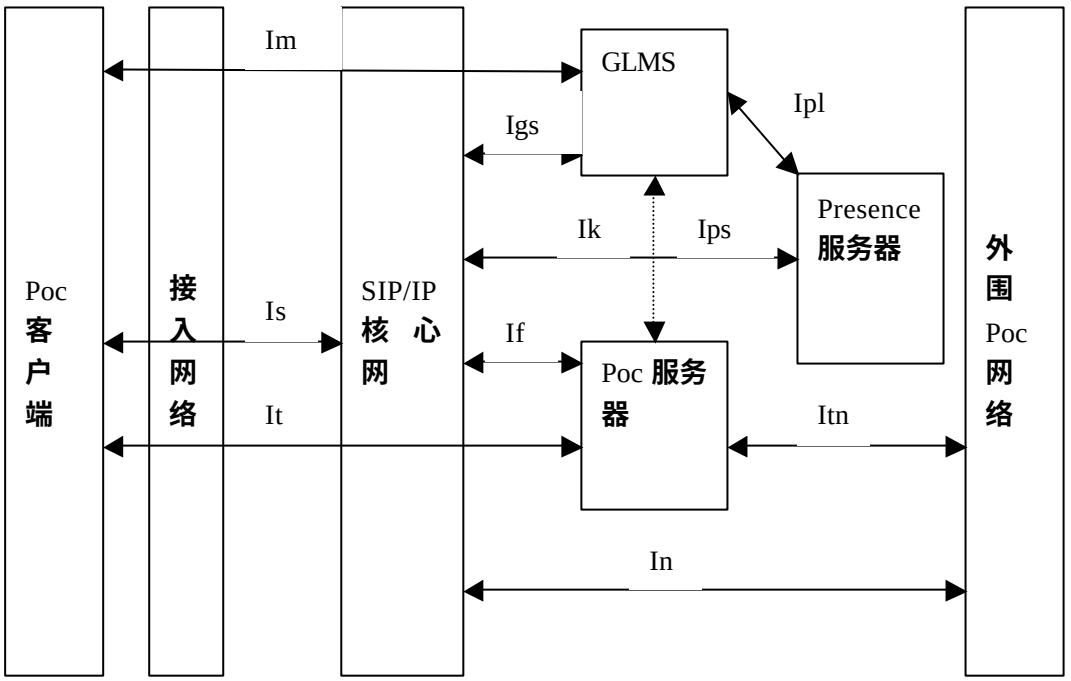

图 3-1　基于 IMS 域的 POC 方案

（5）提供群组会话中的策略强制执行；

（6）提供 POC 来话的策略处理，如接入控制，可用状态等；

（7）提供成员信息，如用户的 ID/绰号；

（8）收集和提供媒体质量信息；

（9）提供计费报告；

（10）群组和列表服务器（GLMS）。

POC 用户使用 GLMS 来管理群组和列表，如地址本和黑白名单。GLMS 有如下功能：

（1）提供列表管理操作，如群组和列表的创建，修改，读取和删除；

（2）提供群组和列表的存储；

（3）SIP/IP 核心。

基于 3GPP 和 3GPP2 的 IMS 的能力，主要包括一 些的 SIP 代理（proxy）和 SIP

登记服务器（registrars）。SIP/IP 核心有以下功能：

（1）负责 POC 客户端和服务器之间 SIP 信令的路由；

（2）提供地址解析与寻址功能；

（3）支持 SIP 压缩；

（4）根据用户的业务 profile 对 POC 客户端进行鉴权和授权；

（5）保持登记状态；

（6）提供计费信息。

在需求分析的基础上，结合设计模式的讨论，设计了 IM 接口的类图，并对鉴权和 Action 部分进行了进一步的细分析和设计。

3.2 PTT 用户信息管理系统的模型

用户信息管理是 PTT 总体解决方案的一部分，采用 B/S 结构，客户端可以是手机和任何其他设备，只要他们发送的消息遵循要求的格式。我们所要实现的就是一个服务器端应用程序，可以重头构建一个服务器应用程序，需要考虑多个客户端的请求，同时请求的压力；也可以采用现有的成熟服务器，如 Apache, IIS 等，在此基础上，按一定框架编写服务器应用程序，重点放在业务实现上。前一种需要大量的调试和性能测试，优点是处理单一，速度会有提高；后一种可以减少大量代码，可靠性高。在项目压力大的情况下，我们选择了后一种方案，在 IIS 上构建服务器程序，省去了编写处理多个客户端请求的繁重代码，极大提高了开发进度。这也体现了重用的思想，结构图 3-2 所示：

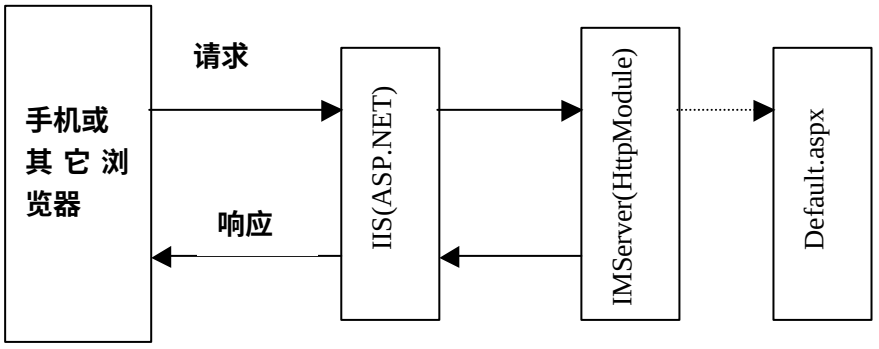

图 3-2　系统模型

客户端发送一条请求（该请求必须符合 HTTP/1.1 协议规范），被 IIS 传递到上层；在激发 HttpApplication 的 BeginRequest 事件的时候，扩展一个模块，用来处理我们的任务；处理完任务后，返回相应的结果，并关闭应用程序。流程如图 3-3：

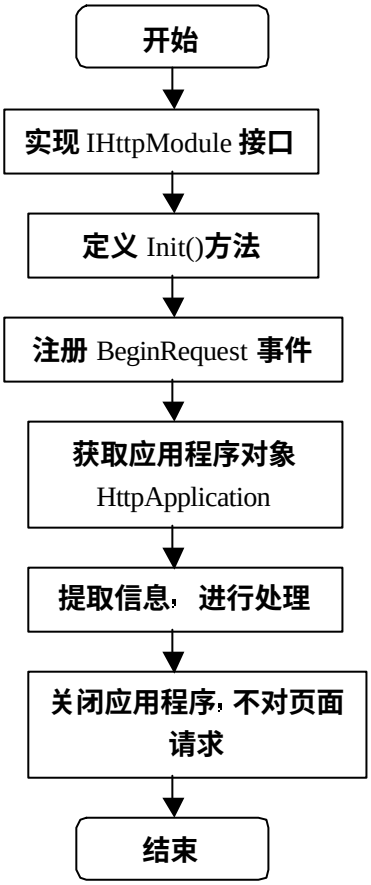

图 3-3　模型流程

3.3　系统总体结构

PTT 用户信息管理的总体结构可以分为五大模块，Log 模块、IMServer 模块、Authorization 模块、Action 模块以及数据访问模块，如图 3-4 所示。总体结构的设计体现控制分散的策略，IMServer 模块处理完后，控制权转移到 Authorization 模块；Authorization 模块处理完后，控制权转移到 Action 模块。避免出现超级控制类，使类设计复杂。

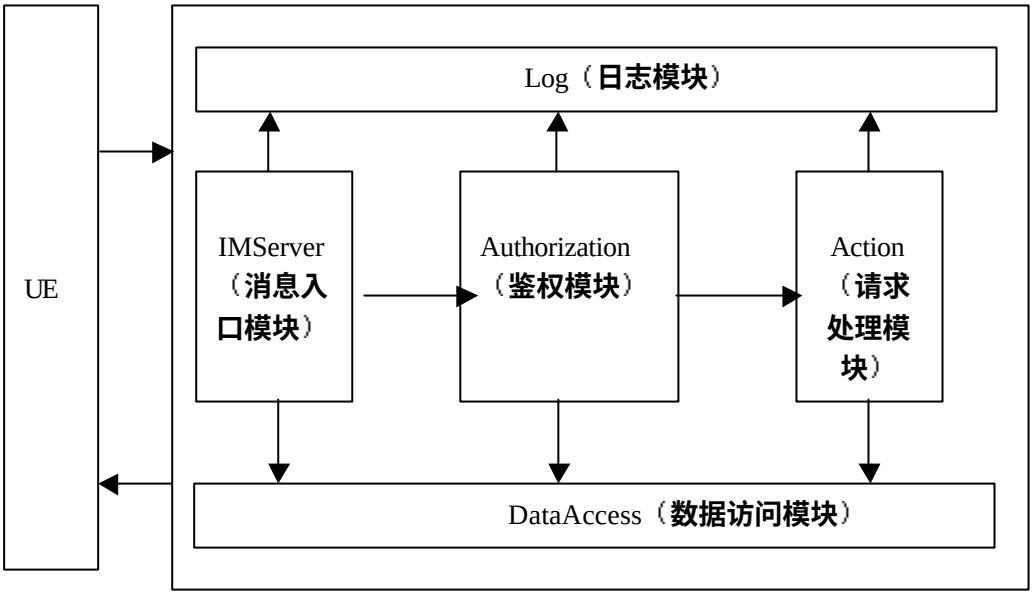

图 3-4　总体结构设计

IMServer 首先获得用户请求后，解析 uri 和鉴权头信息，根据配置鉴权方式，生成鉴权对象，控制权转移到鉴权模块；鉴权模块对该用户进行鉴权处理，成功后，生成相关的 action 处理对象进行处理。在没个过程中，都会对数据访问模块和日志模块进行调用，这两个部分是横向模块。流程图如 3-5 所示：

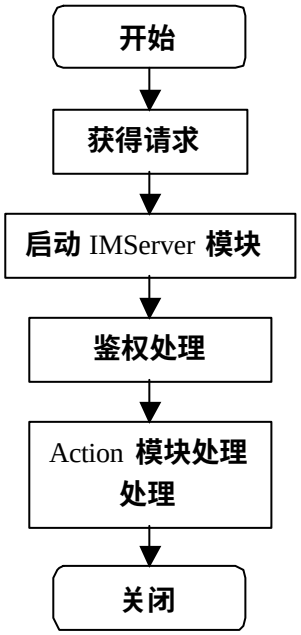

图 3-5　模块间流程处理

3.4 消息入口模块的设计

消息入口模块（IMServer）用于获取由 IIS 传上的用户请求，包括请求的 URI 和鉴权头信息，并负责解析。没有错误时，利用这些信息，创建鉴权对象，转移控制权。

IMServer 主要完成以下功能：

（1）检查请求是否为 GET 请求；

（2）解析 Uri，并判断域名是否合法；

（3）记录用户访问信息；

（4）有鉴权头时，解析鉴权消息；

（5）生成鉴权对象，控制权转移给鉴权。

请求中所用字段的规格如表 3-1 所示。

表 3-1 字段规格说明

字段	类型	长度（字节）
用户号码	整型	20
密码	字符串	16
群组号码	字符串	22
群组显示名称	字符串	48
聊天室类型	整型	1

3.5 鉴权模块的设计

鉴权模块（Authorization）完成对用户身份的验证，根据业务需要，分为有验证和无验证模式；当为无验证模式时，对用户不进行验证，直接送到下一模块处理。当为有验证模式时，对用户密码进行摘要访问鉴权（Digest Access Authentication）后，与用户携带的 Response 部分进行比较，以验证有效性。

3.5.1 鉴权流程

当客户端向服务端用 GET 请求受保护的文档的时候，而且客户端和服务端都知

道该文档的所有者和密码。第一次，客户端发送没有鉴权头的请求，服务端收到以后，返回如下信息给客户端，参数含义见表 3-2：

HTTP/1.1 401 Unauthorized

WWW-Authenticate: Digest

realm="10.10.0.1",

nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",

客户端收到第一次请求的结果后，提示用户输入用户名和密码，经过计算，发送新的请求，参数含义见表 3-2：

Authorization: Digest username="139",

realm="10.10.0.1",

nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",

uri="/dir/index.html",

response="6629fae49393a05397450978507c4ef1"

表 3-2　鉴权中的关键字

关键字	含义
Digest	表明是摘要鉴权
Realm	用户所在域
Nonce	MD5 算法产生的 48 位随机数
Username	用户私有 ID
Uri	请求消息
Response	MD5 计算和，含密码信息，48 位随机数

验证 response 通过后，对请求进行下一步处理；否则，产生新的 nonce 重复上面的步骤。如果三次不对，这对该用户锁定半个小时。流程见如图 3-6。

在程序中，第一次和第二次是一个状态管理，需要记录下来。在 ASP.NET 里，可以使用 cookie 和数据库保存等几种方式实现。但是，我们的客户端一般是手机，不具有 cookie 功能，所以我们采用数据库保存每个用户的办法，设计的状态表称为 nonce 表，其结构如表 3-3。

表 3-3 nonce 记录表

表示每个客户端	随机数	客户访问的时间
Ip	Nonce	Time

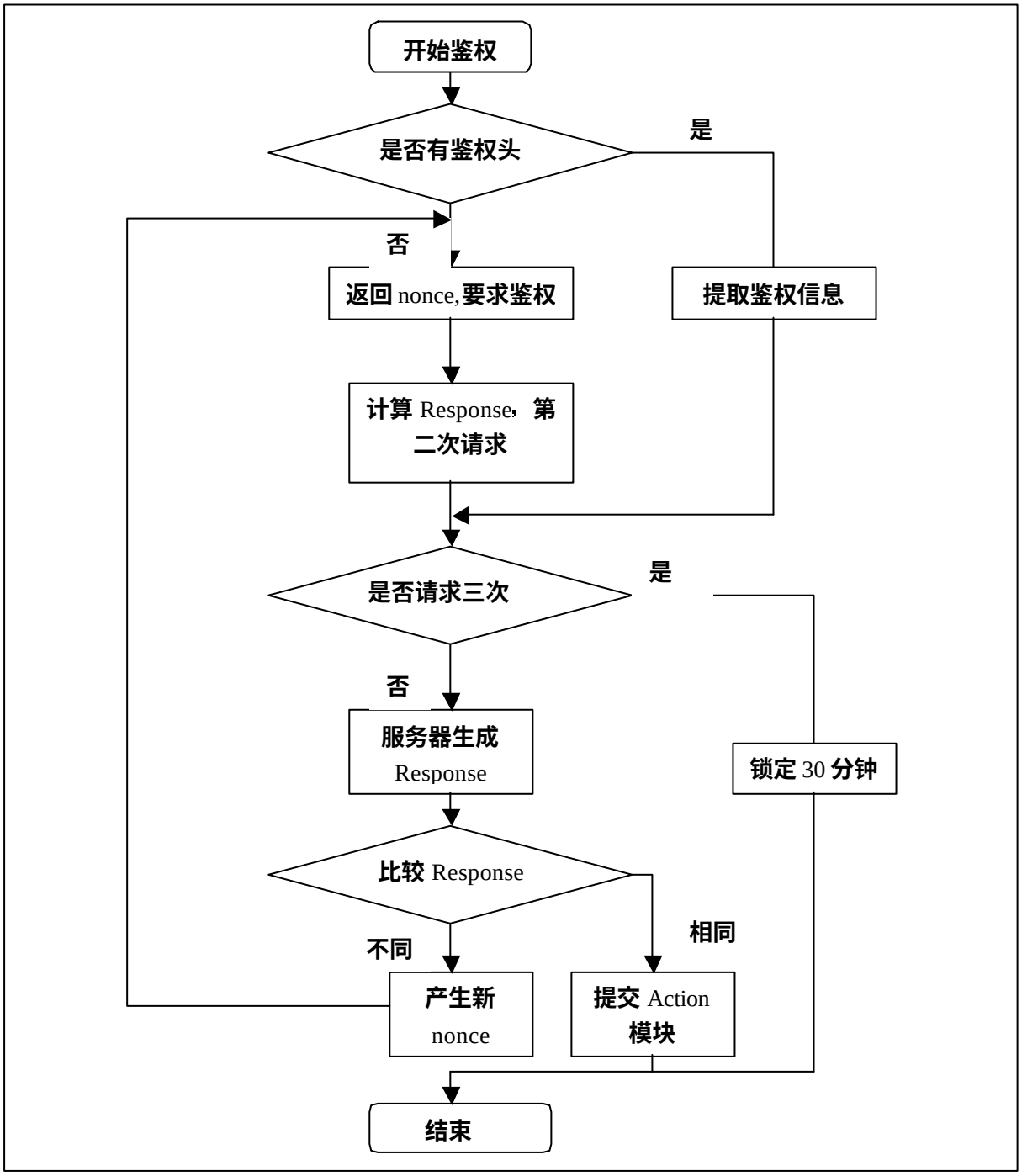

图 3-6　鉴权流程

nonce 表初始化为空，对客户端采用 ip 地址进行唯一识别。当有一个客户端访问时，查找 ip 地址是否已存在，如果不存在，则产生一条记录，插入到表中；如果已存在，则比较时间，如果当前时间和数据库记录中的时间间隔超过 5 秒，则该 nonce 失效，重新产生 nonce，要求客户端鉴权。

3.5.2 鉴权模块的详细设计

策略模式定义一系列的算法，把它们一个个封装起来，并且使它们可相互替换。使得算法可独立于使用它的客户而变化。

（1）适用性

当存在以下情况时使用 Strategy 模式：

当存在相关的类仅仅是行为有异。“策略”提供了一种用多个行为中的一个行为来配置一个类的方法[48]。

需要使用一个算法的不同变体。例如，你可能会定义一些反映不同的空间/时间权衡的算法。当这些变体实现为一个算法的类层次时，可以使用策略模式。算法使用客户不应该知道的数据。可使用策略模式以避免暴露复杂的，与算法相关的数据结构。

一个类定义了多种行为，并且这些行为在这个类的操作中以多个条件语句的形式出现。将相关的条件分支移入它们各自的 Strategy 类中以代替这些条件语句。结构如图 3-7。

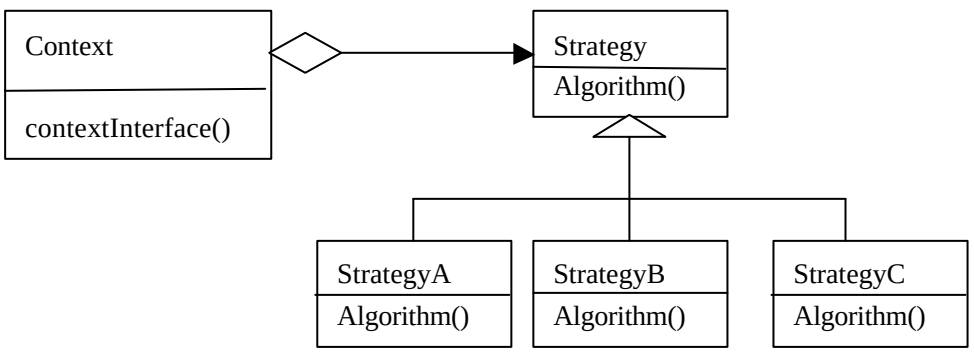

图 3-7　策略模式

（2）参与者

Strategy 定义所有支持的算法的公共接口。Context 使用这个接口来调用某 ConcreteStrategy 定义的算法。

ConcreteStrategy 以 Strategy 接口实现某具体算法。

Context 用一个 ConcreteStrategy 对象来配置，维护一个对 Strategy 对象的引用，可定义一个接口来让 Strategy 访问它的数据。

（3）协作

Strategy 和 Context 相互作用以实现选定的算法。当算法被调用时，Context 可以将该算法所需要的所有数据都传递给该 Strategy。或者 Context 可以将自身作为一个参数传递给 Strategy 操作。这就让 Strategy 在需要时可以回调 Context。

Context 将它的客户的请求转发给它的 Strategy。客户通常创建并传递一个 ConcreteStrategy 对象给该 Context；这样，客户仅与 Context 交互。通常有一系列的 ConcreteStrategy 类可供客户从中选择。

（4）效果

Strategy 模式有下面的一些优点和缺点：

①关算法系列 Strategy 类层次为 Context 定义了一系列的可供重用的算法或行为。继承有助于析取出这些算法中的公共功能。

②个替代继承的方法 继承提供了另一种支持多种算法或行为的方法。你可以直接生成一个 Context 类的子类，从而给它以不同的行为。但这会将行为硬性编制到 Context 中，而将算法的实现与 Context 的实现混合起来，从而使 Context 难以理解，难以维护和难以扩展，而且还不能动带地改变算法。最后你得到一堆相关的类，它们之间的唯一差别是它们所使用的算法或行为。将算法封装在独立的 Strategy 类中使得你可以独立于其 Context 改变它，使它易于切换，易于理解，易于扩展。

③除了一些条件语句 Strategy 模式提供了用条件语句选择所需的行为以外的另一种选择。当不同的行为堆砌在一个类中时，很难避免使用条件语句来选择合适的行为。将行为封装在一个个独立的 Strategy 类中消除了这些条件语句。

④现的选择 Strategy 模式可以提供相同行为的不同实现。客户可以根据不同时

间/空间权衡取舍要求从不同策略中进行选择。

⑤户必须了解不同的 Strategy 本模式有一个潜在的缺点，就是一个客户要选择一个合适的 Strategy 就必须知道这些 Strategy 到底有何不同。此时可能不得不向客户暴露具体的实现问题。因此仅当这些不同行为变体与客户相关的行为时，才需要使用 Strategy 模式。

⑥trategy 和 Context 之间的通信开销 无论各个 ConcreteStrategy 实现的算法是简单还是复杂，它们都共享 Strategy 定义的接口。因此很可能某些 ConcreteStrategy 不会都用到所有通过这个接口传递给它们的信息；简单的 ConcreteStategy 可能不使用其中的任何信息。这就意味着有时 Context 会创建和初始化一些永远不会用到的参数。如果存在这样的问题，那么将需要在 Strategy 和 Context 之间更进行紧密的耦合。

⑦加了对象的数目 Strategy 增加了一个应用中的对象的数目。有时你可以将 Strategy 实现为可供各 Context 共享的无状态的对象来减少这一开销。任何其余的状态都由 Context 维护。Context 在每一次对 Strategy 对象的请求中都将这个状态传递过去。共享的 Strategy 不应在各次调用之间维护状态。

鉴权模块的需求主要有 MD5 摘要加密和无鉴权，设计鉴权模块图 3-8。

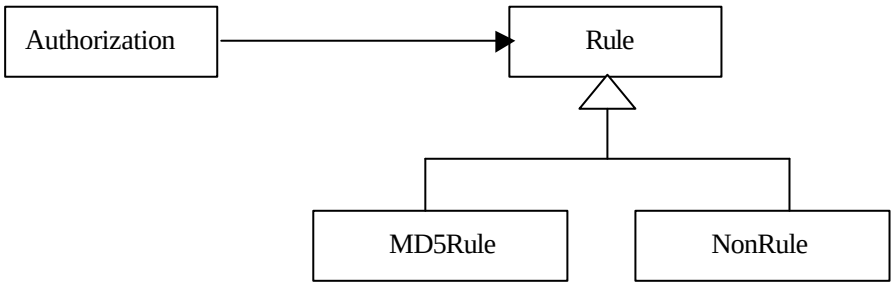

图 3-8　鉴权模块设计

图 3-8 是对鉴权模块的进一步设计，具体鉴权模块都从 Rule 类派生，重写其虚方法 DoAuth()。对不须鉴权模块，直接返回真。对 MD5 鉴权过程的实现比较复杂，要考虑到状态管理和超时等多种因素。IMServer 解析 uri 后，把必要信息传递给鉴权模块。鉴权模块首先判断是否含有鉴权头，如果没有产生 nonce，返回给客户端 401，未授权。如果客户端带有鉴权头，则查找该客户端的 ip 地址是否存在，如果存在。

随机数 nonce 的生成：nonce 的生成要考虑重复的可能性，尽量做到不重复；对

前时间戳和只有服务器端知道的私有数计算它们的 MD5 算法的和，nonce=H(time-stamp: private-key),其中 H 表示 MD5 哈希算法。计算得到随机的 HASH 数，即为 nonce 的值。但是这样仍然可能存在重复的 nonce，解决办法是对 nonce 进行生存期的限制对某个 nonce 的生存期限制在 5 秒内，在数据库中可以对此 nonce 和其产生时间记录在一起，以便比较。

Response 的生成分两个步骤，首先利用算法名称、用户名、域名、密码和 nonce 随机熟，生成哈希数 HA1;然后用请求方法名和请求 uri 计算哈希数 HA2;最后用 HA1 和 HA2 生成验证码 response。算式如下：

$$HA1=H(Alg,\ UserName,\ Realm,\ Password,\ Nonce) \tag{3.1}$$

$$HA2=H(method,\ uri) \tag{3.2}$$

$$Response=H(HA1,\ HA2) \tag{3.3}$$

流程如图 3-9：

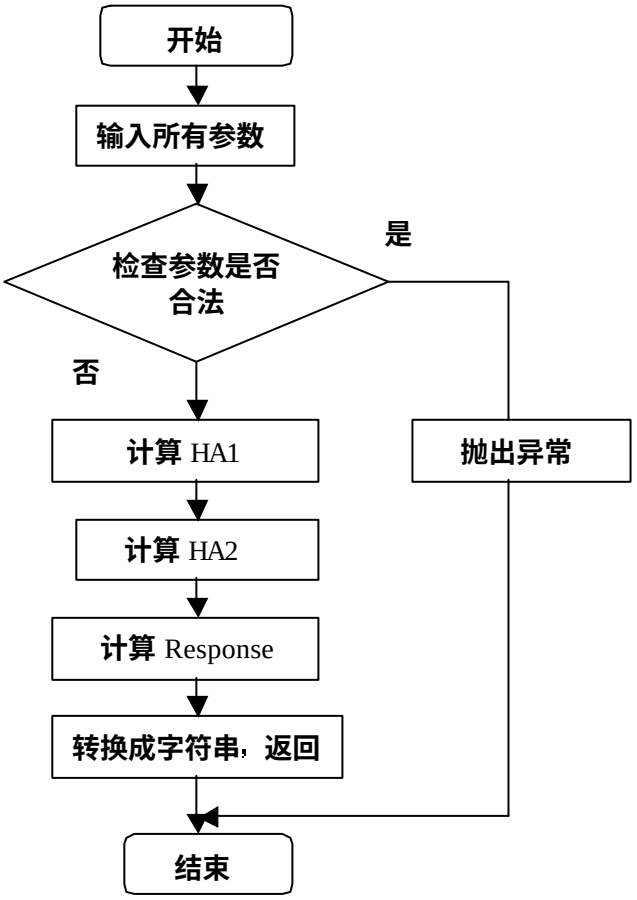

图 3-9 Response 计算流程

3.6 请求处理模块的设计

请求处理模块（Action）完成各个具体请求的相应处理。根据 Action 参数，生成不同对象，处理不同的请求，返回给客户端 XML 文本或者错误码。

3.6.1 接口定义

Action 接口定义主要用于和客户端之间的交互处理，定义基本参数。

（1）下载用户联系列表

当终端用户想获取自己管理的联系列表时，必须输入 action、owner 以及 realm 参数，见表 3-4。

表 3-4 下载用户联系列表

Action	Get_contactlist
Owner	139
Realm	10.10.0.1

（2）下载联系人

查看用户所管理的群组，必须输入 action、owner、realm 以及 contact_type，见表 3-5。

表 3-5 下载联系人

Action	Get_list_of_contactlist
Owner	139
Realm	10.10.0.1
Contact_type	ChatRoom

（3）下载群组成员

下载群组成员，必须输入 action、owner、realm 以及 group，群组使用其号码，不使用显示名称。见表 3-6 定义。

表 3-6　下载群组成员

Action	Get_buddy_of_group
Owner	139
Realm	10.10.0.1
Group	www

（4）创建群组

创建群组必须输入 action、owner、realm 以及 display name，其中 display name 为可选参数，用户可以输入，也可以选择不输入，这时系统应该提供群组号码作为默认显示名称。见表 3-7 中的定义。

表 3-7　创建群组

Action	create_group
Owner	139
Realm	10.10.0.1
Display Name	Group

（5）删除群组

删除群组必须输入 action、owner、group 以及 realm，见表 3-8 定义。

表 3-8 删除群组

Action	delete_group
Owner	139
Group	13901
Realm	10.10.0.1

3.6.2　模块 Action 的详细设计

简单工厂模式是类的创建模式，又叫静态工厂方法（static factory method）模式。简单工厂模式是一个工厂对象决定创建出哪一种产品类的实例[48]。

工厂模式专门负责将有大量共同接口的类实例化。工厂模式可以动态决定将哪

一个类实例化，不必事先知道每次要实例化哪一个类。工厂模式有以下几种形态：

简单工厂（simple factory）模式：又称静态工厂方法模式（static factory method）。

工厂方法(factory method)模式：又称多态性工厂（polymorphic factory）模式或虚拟构造子(virtual constructor)模式。

抽象工厂(abstract factory)模式：又称工具箱模式。简单工厂模式的一般性结构如图 3-10：

（1）角色与结构

简单工厂模式就是由一个工厂类根据传入的参量决定创建出哪一种产品类的实例。

简单工厂涉及到工厂角色，抽象产品角色以及具体产品角色等三个角色：

工厂类（creator）角色：担任这个角色的是工厂方法模式的核心，含有与应用紧密相关的商业逻辑。工厂类在客户端的直接调用下创建产品对象，它往往由一个具体类实现。

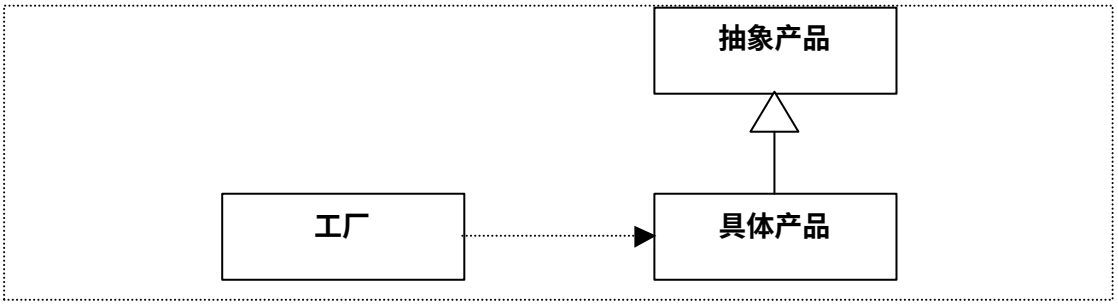

图 3-10　简单工厂模式

抽象产品(product)角色：担任这个角色的类是由工厂方法模式所创建的对象的父类，或它们共同拥有的接口。抽象产品角色可以用一个接口或者抽象类来实现。

具体产品(concrete product)角色：工厂方法模式所创建的任何对象都是这个角色的实例，具体产品角色由一个具体类实现。

（2）简单工厂模式的优点和缺点

模式的核心是工厂类。这个类含有必要的判断逻辑，可以决定在什么时候创建哪一个产品类的实例。而客户端则可以免除直接创建产品对象的责任，而仅仅负责消

费产品。简单工厂模式通过这种做法实现了对责任的分割。

缺点是，工厂类集中了所有的产品创建逻辑，形成了一个全能类。当产品类有不同的接口种类时，工厂类需要判断在什么时候创建某种产品。这种对时机的判断和对哪一种具体产品的判断逻辑混合在一起，使得系统在将来进行功能扩展时较为困难。参照上面的分析，设计 Action 模块如图 3-11 所示。

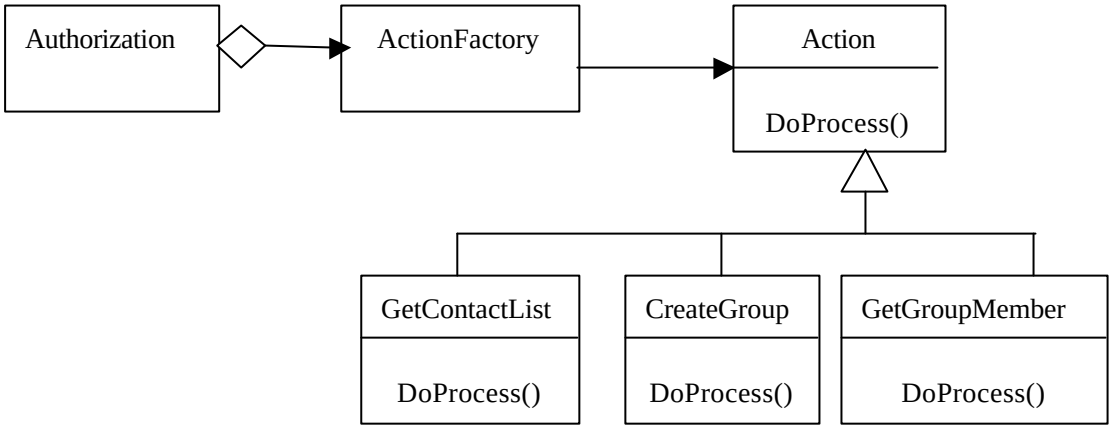

图 3-11　Action 模块设计

图 3-11 是对 Action 模块的进一步设计，系统中实现了 16 个 Action。每个 Action 都派生与基类 Action,实现虚函数，完成的功能和流程大致一样。类 GetContactList、类 CreateGroup 和类 GetGroupMember 都派生与同一个基类，实现其中的虚方法。首先类 ActionFactory 按照用户请求中的命令关键字，判断生成哪个具体的消息处理类，并把该对象返回给工厂类；在鉴权类中定义有消息处理类的基类，这样用基类代表具体类，实现多态性。这样在后续的开发中，可以方便的增加剩下的 16 个功能，而对现有模块产生最小的影响。

对单个模块的处理，先获得输入参数，对该函数进行入口的日志记录，并检查所有参数是否存在，如果有不存在的，则向客户端发送 400(Bad Request)错误码，并结束本次请求；如果成功，则调用数据访问模块，如果有异常，则返回 500 错误码给客户端；如果成功，则获取响应命令的处理码或则 XML 文本文件。最后对所有操作进行日志记录，并把结果发送给客户端。基本流程如图 3-12。

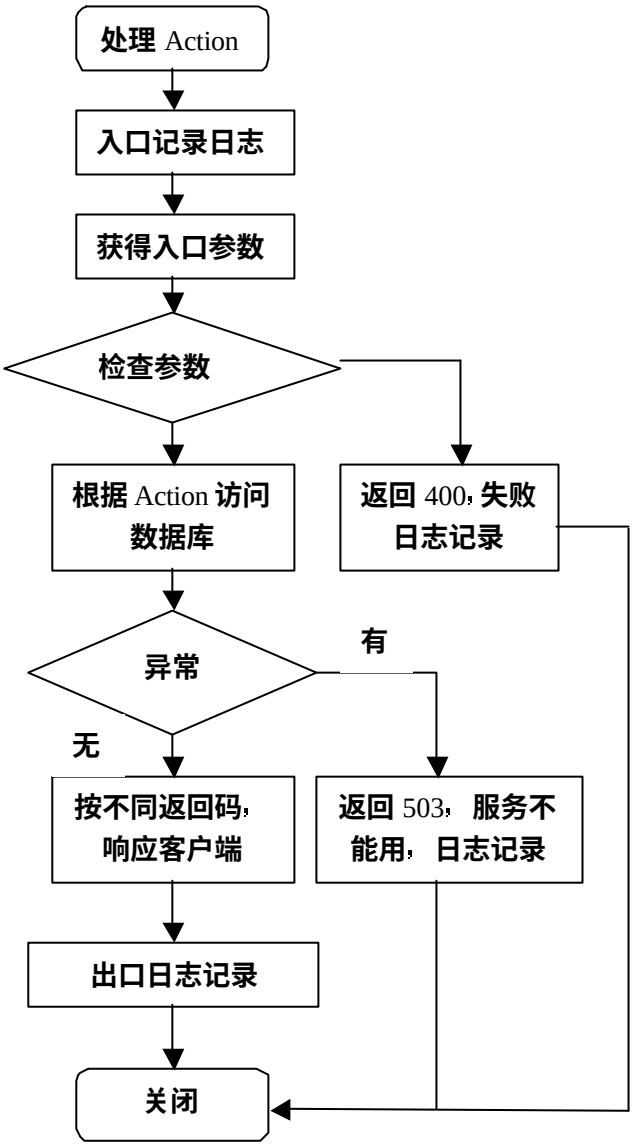

图 3-12　Action 模块的处理过程

3.6.3　错误处理

设计错误码的策略：每一种错误采用一种错误码；错误码的选择优先使用 HTTP 协议中出现的错误码和错误描述，如果没有对应错误码，则从 600 定义特定的错误码，定义如表 3-9 所示。

错误码的实现考虑：在应用程序中硬编码错误码，即对每个 Action 中的异常处理和由数据库返回的错误码都直接赋值，statusCode = 200、statusDescription = “OK”。

这样处理带来的问题是，修改错误码和错误描述时，每个 Action 模块用到的错误码都要修改，工作量大，不利维护。

解决的办法是采用定义错误码资源，生成动态链接库；在程序中查找错误码和对应的错误码描述。

表 3-9　错误码定义

错误码	摘要	描述
200	OK	操作成功
304	Not Modified	请求的内容未修改
400	Bad Request	请求中有错误，包括关键字错误，域名错误
401	Unauthorized	用户没有鉴权
403	Forbidden	对不属于自己的资源进行请求
408	Request Timeout	请求超时
503	Service Unavailable	程序错误，但只显示给用户服务不可用
600	Contactlist empty	联系列表为空
601	Group Empty	群组为空
602	Type not match	类型不匹配
603	Max number group per user	用户所创建的群组达到最大
604	User barred	用户被锁定

3.7　日志模块的设计

主要分为调试日志，跟踪日志和安全日志。提供给各个模块记录信息，方便调试和跟踪用户。

调试日志：主要记录异常信息和程序有可能出现错误的位置，例如 try 和 catch 块中的语句。

跟踪日志：对程序流程进行跟踪，了解程序的逻辑运行过程，方便调试和跟踪

错误。分为入口跟踪，出口跟踪以及内部函数入出跟踪。

安全日志：对鉴权时的安全信息进行记录。安全日志分为低、中、高三个级别，可对日志记录按级别进行过滤显示。用户每次鉴权都要进行记录，如果验证成功，则安全日志记录级别为低；如果验证不成功，则安全日志记录级别为高；达三次失败，则锁定用户 30 分钟。

日志存储位置：所有日志的存储都记录在 Windows 操作系统的事件查看器里。

日志记录内容：请求对象，响应对象，操作类型，操作时间，摘要，具体描述。

3.8 数据访问模块的设计

统指数据访问逻辑模块和物理数据库。数据库使用 Sql Server 2000，访问数据库主要使用 ADO.NET 来完成对数据库访问，功能主要包括获取用户密码，群组信息，更改用户等数据库操作。

数据访问层在全局上是一个组件，设计成一个类比较合适,里面包含各种数据库操作方法，分为查询、删除、增加以及修改，所有方法的调用都采用存储过程的方式。服务器端只提供数据业务，对与数据的处理和显示，由客户端决定；另外客户端平台也是多样的，比较适合采用 XML 作为数据业务的载体[47]。由与访问数据库得到的是数据集，需要把数据集记录转化成 XML 文件，这部分也放在数据访问模块里完成。其结构设计如下图 3-13：

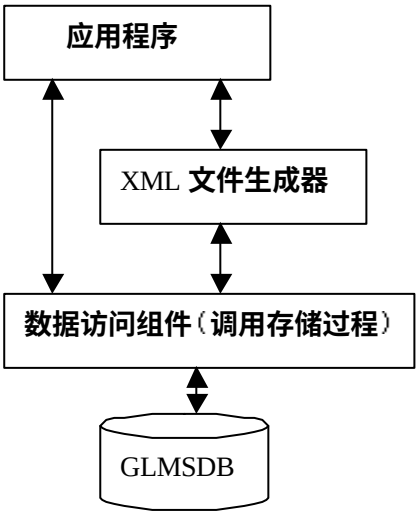

图 3-13　数据访问结构

XML 文件的生成过程：对于数据库的访问，采用的是 ADO.NET，返回的是一个数据集，需要在内存中重新构造 XML 文档。首先，创建 XML 文档，接着创建 XML 声明，然后对数据集进行循环访问，创建相应的节点。流程如下图 3-14：

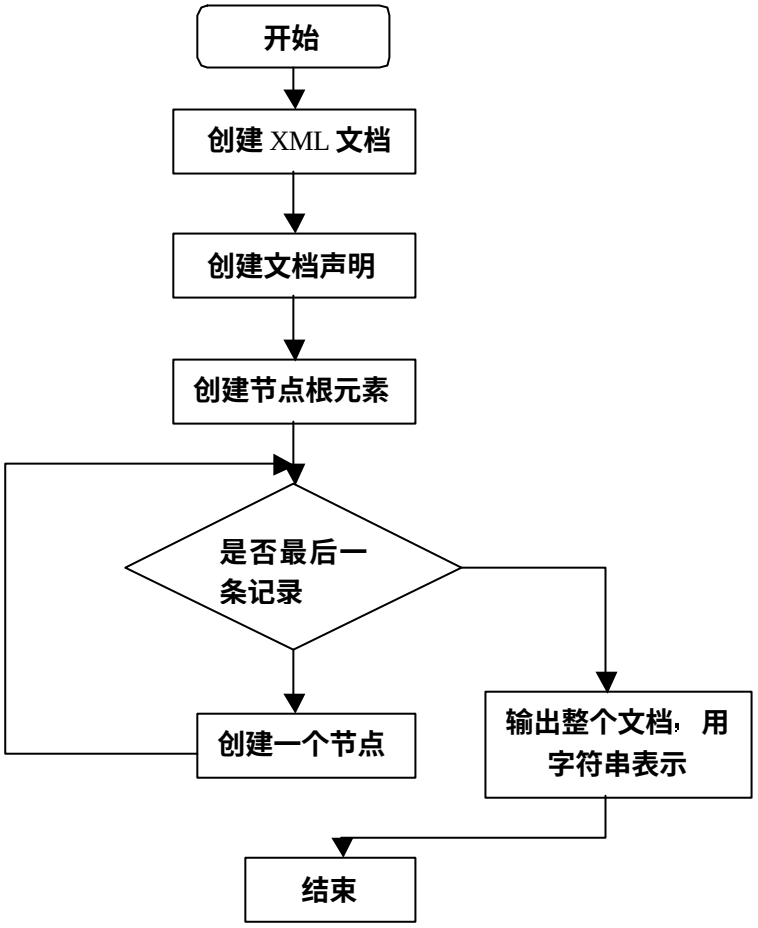

图 3-14　XML 文件的生成过程

3.9　小结

这一章我们首先讨论了 PTT 的总体结构，接着重点在总体类结构的设计，Action 模块以及鉴权模块的设计。总体结构突出控制分离的思想，不在一个模块里做所有控制，形成超复杂类。在 Action 模块，采用工厂方法，提高可扩展性；在鉴权模块，用到了策略模式，也起到了扩展的目的。

4 PTT 用户信息管理系统的实现

PTT 用户信息管理系统是一个 Web 服务程序，没有运行界面，主要是功能方面的实现。

4.1 应用程序配置

应用程序在以下方面能够进行灵活配置，达到客户的需求：

（1）配置IMServer

应用程序的配置，在标签<httpModules>和</httpModules>之间。Name表示自定义的http模块的友好名称，在这里注册了一个http模块，进行我们的业务处理。

（2）配置鉴权方式

鉴权方式分为两种，一种是进行摘要鉴权，另一种无须鉴权。“0”表示要进行摘要鉴权，其它任何字符表示不需要鉴权。

（3）配置数据库连接字符串

配置Sql Server数据库所在的位置，server表示数据库服务器的地址；UserID表示数据库用户，有一定的权限；Password表示该用户的密码；database表示所建的应用程序的数据库，即用户信息管理系统所访问的数据库。

（4）配置日志输出等级

日志等级分为三级，设定等级的目的是为了显示问题的严重程度，也方便对日志进行过滤显示。“0”表示不输出任何日志；“1”表示日志等级为信息级别；“2”表示级别为告警级别；“3”表示日志级别为严重级别，为最高级别，需要马上处理。

（5）配置日志服务器

日志的输出可以输出到本机的事件查看器中，也可以输出到其它远程机器上，需要指明日志服务器的地址。

下面是一份完整的配置文件，用XML描述：

<configuration>

```
<system.web>
    <httpModules>
      <add name="Module" type="IMServer.IMServer, IMServer" />
    </httpModules>
</system.web>
<appsettings>
      <add name="authentication_mode" value="1">
        <add name="connection_string" value="server=10.10.0.5;User
ID=PTTUser;Password=wqs;database=GLMSDB;Connection Reset=FALSE">
        <add name="debug_level" value="0">
        <add name="log_server" value="10.10.0.5">
</appsettings>
</configuration>
```

4.2 功能实现

手机用户可以发送各种服务器识别的命令来维护自己的数据。命令格式遵守 HTTP/1.1 **协议规范，所有命令使用** GET **方法。服务器端只提供数据服务，不涉及到界面设计。**

（1**）下载用户联系列表**

如果手机用户想查看自己所拥有的联系列表，可以使用下载联系列表的功能来完成。用户 139 **想查看自己的联系列表，可以使用** get_contactlist **操作来请求。**

请求命令：

```
GET  http://localhost/script?action=get_contactlist&owner=sip:139@
10.10.0.1 HTTP/1.1
Host: 10.10.0.1
Authorization:user=139;uri=http://localhost/script?action=get_contactlist&owner=si
p:139@10.10.0.1;realm=10.10.0.1;nonce=;response=
```

服务器响应：

<?xml version=”1.0” encoding=”UTF-8”?>

<contactlists>

<contactlist>

<ID>sip:139@211.71.17.221;contactlist=ChatRoom</ID>

<owner>139</owner>

</contactlist>

<contactlist>

<ID>sip:139@211.71.17.221;contactlist=NetworkGroup</ID>

<owner>139</owner>

</contactlist>

</contactlists>

XML **文件表明该用户有两个联系列表，联系列表的类型分别** ChatRoom(**聊天室**) **和** NetworkGroup（**即时通话组**）。

（2）**下载联系人**

用户如果想查看联系人，也就是联系列表管理的群组，则可以通过 get_list_of_contact_list **来完成。用户** 139 **想查看自己的联系列表** 139；contact_list=ChatRoom **里的群组，类型为聊天室，可以使用下面的命令。**

请求命令：

GET

http://localhost/script?action=get_list_of_contactlist&owner=sip:139@10.10.0.1&contactlist_uri=sip:139@10.10.0.1;contact_list=ChatRoom HTTP/1.1

Host:10.10.0.1

Authorization:

user=139;uri=h|tp://localhost/script?action=get_list_of_contactlist&owner=sip:139@10.10.0.1&contactlist_uri=sip:139@10.10.0.1;contact_list=ChatRoom;realm=10.10.0.1;nonce=;response=

服务器响应：

```
<?xml version=”1.0” encoding=”UTF-8”?>
  <groups>
        <group>
            <ID>123</ID>
            <name>John</name>
            <type>chatroom</type>
        </group>
        <group>
            <ID>124</ID>
            <name>Rose</name>
            <type>chatroom</type>
        </group>
</groups>
```

XML **文件表明，用户** 139 **在其聊天室类型的联系列表下面拥有两个群组，群组号分别为** 123（**名称为** John）**和** 124（**名称为** Rose）。

（3）**下载群组成员**

用户查看群组的成员，进行群组通信，可以使用命令 get_buddy_of_group **来完成。用户** 139 **下载他自己的群组** sip:www@10.10.0.1 **里的用户成员。**

请求命令：

```
GET
http://localhost/script?action=get_buddy_of_group&owner=sip:139@10.10.0.1&gro
up=sip:www@10.10.0.1 HTTP/1.1
Host:10.10.0.1
Authorization:                                                    u
user=139;uri=http://localhost/script?action=get_buddy_of_group&owner=sip:139@1
0.10.0.1&group=sip:www@10.10.0.1;realm=10.10.0.1;nonce=;response=
```

服务器响应：

<?xml version=”1.0” encoding=”UTF-8”?>

<group>

<member>

<ID>123</ID>

<name>John</name>

</member>

</group>

XML **文件表明，群组** sip:www@10.10.0.1 **里有一个成员，成员的** ID **号为** 123，**显示名称为** John。

（4）创建群组

用户创建一个群组时，使用 create_group **命令来完成。用户** 139 **创建自己的一个群组，其号码自动生成。**

请求命令：

GET http://localhost/script?action=create_group&owner=sip:139@10.10.0.1 HTTP/1.1

Host:10.10.0.1

Authorization:user=139;uri=http://localhost/script?action=create_group&owner=sip:139@10.10.0.1;realm=10.10.0.1;nonce=;response=

服务器响应：

200 OK HTTP/1.1

表明创建群组成功，用户可以通过上面介绍的命令对这个群组进行查看。

（5）删除群组

用户想删除一个群组，可以通过 delete_group **命令来完成。用户** 139 **想删除自己的一个群组** 13901，**可以发送下面的命令。**

请求命令：

GET http://localhost/script?action=delete_group&group=sip:13901@10.10.0.1

HTTP/1.1

Host:10.10.0.1

Authorization:

user=139;uri=http://localhost/script?action=delete_group&group=sip:13901@10.10.

0.1&group=sip:www@10.10.0.1;realm=10.10.0.1;nonce=;response=

服务器响应：

200 OK HTTP/1.1

响应表明删除群组 13901 **成功，用户可以通过相关查看命令来查看验证。**

4.3 小结

本章从实现的角度对系统进行了说明。首先讲了系统的配置所涉及到的参数，然后对系统所实现的功能进行了说明，包括下载用户联系列表、下载联系人、下载群组成员、创建群组以及删除群组。

5 结束语

软件开发是一项实践性很强的一门学科，本论文的成型来源于本人在实际开中的体会和设计。但没有涉及到测试和维护，这在软件开发中占相当大的比例。通过实际开发，对软件的各个方面有了较深的理解和认识。

在技术方面，首先介绍了 PTT 业务的技术定位，标准和规范，对当前 PTT/POC 核心技术进行了分析，包括体系结构，信令协议以及服务器和终端的实现方式。项目只实现了 PTT 体系结构中的部分模块，主要是用户数据的管理部分。然后对 XML，ASP.NET，设计模式等技术围绕软件开发进行了介绍和分析。

设计上，系统结构分为 5 大模块，消息入口模块、鉴权模块、请求处理模块、日志模块和数据访问模块。其中日志模块和数据访问模块设计成系统中的横向模块，为其它三个模块提供日志记录、访问数据库和生成 XML 文件的服务；消息入口模块、鉴权模块和请求处理模块是系统的逻辑部分，分三个步骤依次处理用户发来的请求，实现控制的分离。消息入口模块完成对用户的请求的检查，并解析该请求和鉴权头消息；鉴权模块负责对用户进行鉴权，主要采用摘要访问控制验证策略，计算用户密码的哈希散列值，用于网上传输；请求处理模块实现对用户的请求的处理，调用数据访问层，根据返回结果，发送不同的处理码和描述信息给用户；日志模块分为安全日志、调试日志和跟踪日志；数据访问模块利用存储过程访问数据库，返回错误码或者 XML 文件给调用者。由于鉴权模块和请求处理模块的设计相对比较复杂，在详细设计中，分别采用了策略模式和工厂方法模式，提高了系统的扩展能力和灵活性。

这次开发中，设计也有些不足的地方，给后继的测试、维护带来了一定的困难，是需要该进的地方。在日志模块的设计中，输出信息不明确，调试日志、安全日志以及跟踪日志，分类比较混乱，导致不能很快的定位问题；另外，对象在结构上划分不明确，数据和操作分离。这些是在开发中存在的一些问题。希望在重构过程中，完善这些方面。

致　谢

在论文完成之际，我要感谢我的导师陆永忠教授。从理论学习、选题、软件工程实践到论文的撰写，都得到了陆老师的精心指导。陆老师严谨的治学态度，一丝不苟的工作作风，对我在学业上严格的要求和生活上无微不至的关怀，都是我终身难忘。特别是论文紧张的撰写过程中，是陆老师的严格要求和悉心指导，使我得以顺利完成论文。

感谢软件学院的领导和老师们。作为首届 MSE 毕业的学生，我的学习、工程实践和论文工作得到院所有领导和老师的极大关心和支持，正是有了他们的帮助，我的学业才得以顺利完成。

感谢华为公司的周建武产品经理，王华项目经理等领导，他们提供的难得的软件工程实践机会使我学到了课堂上难以学到的软件项目管理和软件工程过程的实践知识。同时也从他们身上学到职业软件人的职业素养。

感谢 GLMS 项目组的车飞，王锦亮，欧阳志，蔡志良，李刚等软件工程师在软件工程实践过程中热情的帮助和建议。

感谢我的家人，他们对我无私的关爱和莫大的支持是我克服困难的最大动力，正是有了他们的默默奉献才成就了我的这次学业。

最后感谢各位专家评委对我的论文的悉心批评和指正！

参考文献

[1] 许童，廖建新. PTT/POC 技术综述. 电信工程技术与标准, 2004(5): 56~60

[2] 王瑜，乐正友. 基于 SIP 协议的 IP 电话增值业务实现技术. 电讯技术, 2003(2): 114~119

[3] 朱扬荷. 几种 PTT 比较. 通信世界, 2004(6): 36~37

[4] 郑国敏. 基于四类移动网络的 Push-To-Talk 解决方案和比较. 通信世界, 2003(9): 52~54

[5] Ken Rehbehn. Push to Talk. Telecommunications Americas. Norwood, 2004, 38(3): 22~24

[6] Tanner, John C. Pushed and Shoved on Push-to-talk. America's Network, 2004, 108(8): 16~18

[7] 曹玖新，张得运. VoIP 实现技术研究. 计算机工程, 2000, 26(10): 497~500

[8] 何斌，黄本雄，胡广等. 基于蜂窝移动通信的 PTT 研究与实现. 计算机应用, 2004, 24(9): 82~84

[9] 孟旭东，许岚. 基于 SIP 的 IP 会话控制技术. 南京邮电学院学报, 2000, 20(4): 48~52

[10] 李琳，王新刚，柴乔林. SIP 协议在开放的 VoIP 模型中的实现. 计算机工程, 2002, 28(8): 222~226

[11] Webb William. Introduction to Wireless Local Loop. Second Editions. Boston: Artech House, 2000. 40~42

[12] Xia Qingguo, Gao Deyuan. Study of Distributed Gateway Architecture based on VoIP. Journal of Dalian University of Technology, 2003, 43(10): 126~128

[13] Joseph Palenchar. Nokia Pushing For Push-To-Talk. TWICE New York, 2003, 18(11): 6~7

[14] 张旭光，平玲娣，潘雪增. 基于 SIP 协议的分布式 VoIP 体系结构的设计与实现.

计算机工程, 2004, 30(14): 95~97

[15] Webb, William. CDMA for WLL. Mobile Communications International. Jan 1999: 61~62

[16] Andersen. Telemate Mobile Consultants. The GSM-CDMA Economic Study, 1998,(12): 30~31

[17] Brodsky Ira. 3G Bussiness Model. Wireless Review, 1999(6): 42~43

[18] Daniels Guy. A Brief History of 3G. Mobile Communications International, 1999, 65(8): 106~107

[19] Gull Dennis. Spread-Spectrum Fool's Gold? Wireless Review, 1999(1): 37~38

[20] Homa Harri, Antti Teskala. WCDMA for UMTS. John Wiley & Sons, 2000(3): 40~41

[21] Smith Clint. Practical Cellular and PCS Design. McGraw-Hill, 2000(1): 40~43

[22] Smith Gervelis. Cellular System Design and Optimization. McGraw-Hill, 2001(6): 50~51

[23] **张柯**，**黄永峰**，**李星**．**基于**. NET **的** SIP **实现**．**计算机工程**, 2004, 30(10): 148~150

[24] **瞿裕忠**，**张剑锋**，**陈峥等**. XML **语言及相关技术综述**．**计算机工程**, 2000, 26(12): 4~7

[25] **毛勇**，**陈奇**．XML: **一种将广泛应用的数据格式描述语言**．**计算机应用研究**, 1999(3): 4~6

[26] **刘俊彦**. HTTP: World Wide Web **的基本协议**．**个人电脑**, 1997(1): 147~148

[27] **李平**. GPRS **网络如何引入** Push to Talk **业务**．**通讯世界**, 2003(4): 64~67

[28] **乐燕群**，**程时端**. IP **电话信令协议——**SIP．**电信技术**, 2000(2): 24~26

[29] **王宗宁**，**高传善**，**吴群**．**应用** SIP **进行多媒体初始化**．**计算机工程**，1997，25(7): 33~35

[30] **叶锡君**，**吴国新**，**许勇**．**超文本传输协议的认证技术研究与实现**．**计算机集成制造系统**, 2001, 7(3): 49~52

[31] Clint Smith, Daniel Collins. 3G Wireless Networks．**第一版**．**李波**，**卢树军**，**张志**

龙等译. **北京**: **人民邮电出版社**, 2003. 238~269

[32] Richard Leinecker. ASP. NET **实用全书**. **第二版**. **北京**: **电子工业出版社**, 2003. 428~447

[33] Lautenbach, Berin. Introduction to XML Encryption and XML Signature. Information Security Technical Report, 2004, 9(3): 6~18

[34] Sperberg-McQueen C.M. Managing Technology: XML and the Future of Digital Libaries. The Journal of Academic Librarianship, 1998, 24(4):314~317

[35] Buneman Peter, Davidson Susan, Fan Wenfei et al. Reasoning about keys for XML. Information Systems, 2003, 28(8): 1037~1063

[36] Guillaume Damien, Murtagh Fionn. Clustering of XML Documents. Computer Physics Communications, 2000, 127(2): 215~227

[37] Fan Wenfei, Kuper Gabriel M. A Unified Constraint Model for XML. Computer Networks, 2002, 39(5): 489~505

[38] Mello Ronaldodos Santos, Castano Silvana, Heuser Carlos Alberto. A method for the Unification of XML Schemata. Information and Software Technology, 2002, 44(4): 241~249

[39] **李波**, **朱艳云**. PUSH-TO-TALK **背后的多媒体子系统剖析**. **通讯世界**, 2004(3): 68~69

[40] **李欣**. **一键通话**. **现代通信**, 2004(4): 23~24

[41] Deutsch, Alin, Fernandez. A Query Language for XML. Computer Networks, 1999, 31(11): 1155~1169

[42] Brown, Allen, Fuchs et al. MSL: a Model for W3C XML Schema. Computer Networks, 2002, 39(5): 507~521

[43] Kristensen Anders. Formsheets and XML Forms Language. Computer Networks, 2002, 31(11): 1189~1201

[44] Erdmann Michael, Studer Rudi. How to Structure and Access XML Documents with Ontologies. Data and Knowledge Engineering, 2001, 36(3): 317~335

[45] Egyedi T M, Loeffen. Succession in Standardization: Grafting XML onto SGML. Computer Standards and Interfaces, 2002, 24(4): 279~290

[46] Teng Lv, Ning Gu, Ping Yan. Normal Forms for XML Documents. Information and Software Technology, 2004, 46(4): 839~846

[47] Chung Tae-Sun, Kim Hyoung-Joo. XML Query Processing Using Document Type Definitions. Journal of Systems and Software, 2002, 64(3): 195~205

[48] Erich Gamma, Richard Helm, Ralph Johnson, et al. **设计模式**. **第一版**. **李英军**, **马晓星**, **蔡敏等译**. **北京**: **机械工业出版社**, 2002. 208~214