独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：李福东　　日期：2008.3.28

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在__年解密后适用本授权书。非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：李福东　　日期：2008.3.28

导师签名：吴伟明　　日期：08.03.28

移动办公平台架构研究与实现

摘 要

随着移动通信技术的发展，办公人员可以摆脱办公室的限制进行移动办公。然而，目前的移动办公还只是通过短信、彩信、WAP 方式完成如邮件、公文到达提醒或者新闻浏览等简单的业务，还不具备多媒体视频业务以及 OA 系统与 ERP、CRM、SCM 等应用系统的集成互通等固定办公中具有的功能，因此迫切需要对移动办公相关技术进行研究来推动移动办公的发展。

IMS 是 3G 核心网的子系统，可以解决移动办公中对多媒体视频数据传输的网络带宽问题，同时 IMS 可以融合 3G 网络、因特网、PSTN、WLAN 等多种网络，可以解决移动办公终端与其他办公终端之间协同办公的问题。SOA 可以实现多个应用系统之间松耦合的架构形式，支持移动办公系统与其它应用系统之间的集成互通。J2EE/J2ME 采用 Java 虚拟机技术屏蔽了不同的硬件和操作系统的差异性，使移动办公业务能够很好地适应外界环境。

本课题重点研究移动办公平台架构模型。首先分析了本课题的研究背景、国内外发展现状、研究的目的及意义，然后介绍了实现移动办公平台的关键技术，接着提出了新型的移动办公平台架构模型并在此基础上给出了移动办公平台架构设计方案，最后通过一个移动会议系统对移动办公平台架构模型进行了测试验证。本文提出的移动办公平台架构模型对于移动办公平台架构设计以及移动办公业务的发展都具有非常重要的意义。

关键词： 移动办公　协同办公　移动办公平台

RESEARCH AND IMPLEMENTATION ON ARCHITECTURE OF MOBILE OA PLATFORM

ABSRTACT

With the development of mobile telecommunication technology,office workers can break away from the limitation of office and work in mobile state. However,current mobile OA only has simple bussiness such as arrival reminder of email or document by way of short message or wap, still has not multimedia video bussiness and integration between OA system and other system such as ERP,CRM,SCM, therefore it is impedence to research on techniques about mobile OA to drive the development of mobile OA.

IMS is a subsystem of 3G core network and can solve network bandwith of multimedia video in mobile OA and it can convenge many networks including 3G,Internet,PSTN,WLAN and it can also solve collabrative work between mobile terminals and other work terminals. SOA can make loose coupling among many application system, support integration between mobile OA system and other application system. J2EE/J2ME mask the difference of different hardware and operation system by way of Java virtual machine, this make mobile OA bussiness adapt environment outside smoothly.

The emphasis of this paper is architecture model of mobile OA.Firstly, analyze background,status in and out,purpose and signifance of the research,then introduce key techniques of the implementation of mobile OA platform and bring forward a new kind of architecture model of mobile OA platform,give the design solution of the architecture of mobile OA platform based on it in succession. At last, test and validate the architecture model of mobile OA platform by virtue of a mobile conference system. The architecture model of mobile OA platform put forward in this paper is most significant to the architecture design of mobile OA platform and the development of mobile OA bussiness.

KEY WORDS: mobile oa collabrative oa mobile oa platform

第一章 绪论

1.1 课题研究背景

移动通信技术的发展和移动终端的出现改变了传统的办公模式，使人们处理公文、收发电子邮件等工作不再局限于办公室内，办公人员可以使用移动办公设备随时随地处理工作中的事务。

移动办公是人们在办公方式上的一种新的尝试，它增强了办公能力，提高了办事效率，但目前的移动办公业务还比较简单，只有邮件到达提示、公文办理提醒和信息浏览等，接入的方式也只局限于短信、彩信和 WAP。随着人们对于移动办公能力的要求不断提高，现有的移动办公业务越来越暴露出其局限性。

目前，在互联网百兆甚至千兆网络带宽的支撑下，加之由于互联网开放的特点形成了在对其研究的技术、方法方面的领先性与成熟性，使得企事业单位基于互联网建立的 OA 系统已经变得非常成熟。然而移动办公业务由于移动通信技术和移动终端能力的制约，在许多方面还远远落后于固定办公。

随着第三代移动通信在无线网络带宽上的提升，移动终端性价比的提高以及各种先进成熟的设计方法在移动领域的引进，为移动办公的进一步发展创造了条件。

移动办公业务系统的支撑平台对于移动办公业务的发展起到非常关键的作用，而目前的移动办公平台还是封闭的呈“烟筒”形的架构方式，一方面建立它建立在 2G/2.5G 的移动通信网络之上，没有传输多媒体视频数据的网络带宽，另一方面也很难与现有的业务系统集成互通，因此很有必要研究一种统一开放的新型移动办公平台架构，这就是本课题的研究背景。

1.2 国内外研究现状

中国的两大移动运营商中国移动和中国联通推出了多个移动办公业务。

中国移动的移动 OA 是通过部署 MAS（Mobile Application Server）服务器的方式与集团客户原有的 OA 系统进行耦合，主要为集团客户提供基于移动无线网络访问单位内部 OA 系统的解决方案。客户能够通过移动终端随时随地的接入公司或者单位内部的 OA 系统，实现公文处理、邮件提醒和集团通讯录等的业务流程。

中国移动的移动 OA 的功能主要包括公文处理、公告发布、集团通讯录、信息查询、日程管理和邮件提醒等。其中公文处理为主要功能包括新建公文、公文处理批复、公文流转、公文查阅和建立公文列表等。

中国联通提出“随时随地，轻松办公”的办公理念，无论何时何地 ,只要在 CDMA 网络信号覆盖的地方，就可以利用手机、PDA、笔记本电脑等移动终端设备通过短信、WAP、BREW 等多种方式与企业的 OA 办公系统进行连接，从

而将公司内部局域网扩大成为一个安全的广域网。中国联通的移动办公的主要方式有：

1）短信方式：实现公文、邮件到达提醒服务。当企业办公系统内的个人公文、电子邮件到达时，会通过短信直接将标题信息或内容提要发送到个人手机上，进行及时的提醒服务。

2）WAP、BREW 方式：浏览详细公文、邮件内容。企业员工可以使用手机通过 WAP 界面的方式访问公司 OA 办公系统，进行公文、邮件等详细信息的浏览。

3）“掌中宽带”上网方式：实现审阅公文、任务指派及邮件传递。

上海贝尔阿尔卡特公司提出了一体化视频会议解决方案，这是是为未来 3G 运营商定做的商务应用解决方案，该方案将实现使用不同的终端的用户(包括 3G 终端、PDA 终端、带摄像头的宽带 PC 机用户、NGN 固定终端等用户)参加同一个视频会议。

可以看出，国内移动运营商推出的移动办公业务还是基于 GPRS 网络的简单业务，各个研究机构对于移动办公的研究还处于实验阶段，没有形成统一的标准与规范。

1.3 课题研究的目的与意义

随着整个社会信息化水平的不断提高，移动办公在帮助办公人员更加方便地处理大量的日常工作的同时，对于企业的核心竞争力也有着日益重要的影响，而目前的移动办公业务已经不能满足人们的办公要求，因此对于移动办公平台架构的研究是很有必要的。

本课题研究以移动通信技术 IMS 网络为支撑，以成熟的 J2EE 和 J2ME 为应用架构基础，结合先进的设计方法 SOA，提出了一种新型的移动办公平台架构模型，达到为移动办公平台架构设计提供参照模型和实现方法的目的。

移动办公平台架构相当于系统实现的核心支撑框架，不但对于系统的可扩展性、可靠性、可维护性和可伸缩性等有着直接影响，而且对于移动办公业务的类型、功能强弱、终端适应能力和实现周期等起着至关重要的作用，因此对于它的研究将促进移动办公相关技术和应用业务的发展，对于提供整个社会的办公能力都有着重要的意义。

1.4 论文内容安排

本论文第一章为绪论，包括课题的研究背景、国内外发展现状、课题研究的目的与意义；第二章讲述实现移动办公平台的关键技术，如 IMS、SOA、SDO、J2EE、J2ME、SIP 等；第三章讲述移动办公平台架构设计方案，包括总体设计方案、服务器端和移动客户端详细设计方案；第四章根据移动办公平台设计方案，结合当今成熟的各种技术和产品，给出了移动办公平台架构的实现方法；第五章通过分析、设计和实现一个移动会议系统对移动办公平台架构的先进性进行测试并给出测试的结果；第六章对本课题研究进行总结和展望。

2 第二章 实现移动办公平台的关键技术

本章包括的关键技术主要是移动办公平台相关的，包括 3G 核心网中 IMS 技术架构，面向服务架构 SOA、Java 相关的 J2EE 和 J2ME 技术架构和 SIP 协议。

2.1 IMS

IMS 即 IP 多媒体子系统（IP Multimedia Subsystem）。它是 3GPP（Third Generation Partnership Project，第三代合作伙伴计划）在 R5（第五版，Release 5）中提出的，其目的是实现网络融合和业务融合。

移动通信技术从第一代的模拟通信技术发展到第二代的数字通信技术 GSM。GSM 仅仅支持语音业务，后来又出现了俗称 2.5G 的 GPRS（通用分组无线业务，General Packet Radio Service）技术，它可以满足数据业务日益增长的要求。

GPRS 网络虽然能够提供数据业务，但是由于网络带宽的限制（最高 384kbps），因而对于视频业务支持不够。到了第三代移动通信（简称 3G）时，带宽最高达到 2Mbps，因此就具有支持多媒体视频业务的能力。

3G 移动通信网中的核心网络就分为 CS（电路交换，Circuit Switched）、PS（Packet Switched）和 IMS 三个域。其中 IMS 域建立在 PS 域之上，采用分层的网络结构，实现了业务与呼叫控制，呼叫控制与承载，承载与接入的分离。它的层次结构图如下所示：

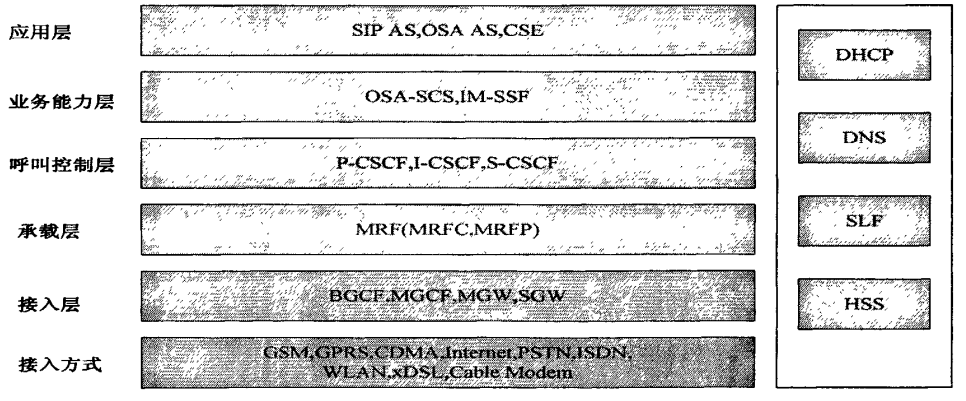

图2-1 IMS层次结构图

上图中，应用层包括各种应用服务器：SIP AS（SIP 应用服务器），OSA AS（开放业务接入应用服务器），CSE（CAMEL 业务环境，Camel Service Environment）。业务能力层包括 OSA-SCS（开放业务接入业务能力服务器），IM-SSF（IMS 业务交换功能）。呼叫控制层包括 3 个 CSCF（呼叫会话控制功能，Call Session Control Function）网元：P-CSCF 为代理 CSCF（Proxy CSCF），I-CSCF 为查询 CSCF（Interrogating CSCF），S-CSCF 为服务 CSCF（Serving CSCF）。承载层包括 MRF（媒体资源功能，Media Resource Function），MRF 分为 MRFC（媒体资源控制功能）和 MRFP（媒体资源处理功能）。接入层包括 BGCF（边界网

关控制功能，Breakout Gateway Control Function），MGCF（媒体网关控制功能，Media Gateway Control Function），MGW（媒体网关，Media Gateway），SGW（信令网关，Signal Gateway）。接入方式包括 GSM，GPRS，CDMA（WCDMA，CDMA2000，TD-SCDMA），Internet，PSTN，ISDN，WLAN，xDSL，Cable Modem 等。另外，HSS 为归属用户服务器（Home Subscriber Server），SLF 为用户定位功能（Subscriber Location Function）。

SIP AS 的功能是提供应用业务；OSA AS 的功能是提供开放业务，可以基于 Parlay API 在不需要了解通信网络细节的情况下快速开发出电信业务，OSA-SCS 相当于 S-CSCF 和 OSA AS 的中介网关；CSE 支持为传统智能网业务，IM-SSF 相当于 S-CSCF 和 CSE 之间的中间网关。

CSCF 是 IMS 中的核心网元，实现呼叫控制的功能。它分为 3 个网元，P-CSCF 是外界接入 IMS 的第一个网元，负责转发来自外部的会话控制并转发 IMS 到外部的呼叫会话控制，另外还进行信令的压缩、解压缩，计费等功能。I-CSCF 的功能是为呼叫方找到合适的 S-CSCF，同时还具有计费和屏蔽 IMS 网络拓扑结构的功能。S-CSCF 根据业务过滤规则触发应用服务器，从而实现呼叫控制与业务的分离，同时还具有计费等功能，可以充当代理服务器、重定向服务器，背靠背服务器和终端服务器。

MRFC 的功能是对媒体网关进行控制，包括媒体流的建立、修改和释放；MRFP 的功能主要是对媒体流的合并、媒体格式的转换等。

BGCF、MGCF、SGW、MGW 的功能是与 PSTN/ISDN/CS 域进行互通。BGCF 的功能是找到合适的 CS 域；MGCF 的功能是控制媒体网关；SGW 的功能是信令转换，及 SIP 与 ISUP（ISDN 用户部分）进行转换；MGW 的功能包括媒体格式转换等。

IMS 中各网元之间采用 SIP、DIAMETER 等协议进行交互，它的逻辑结构图如下所示：

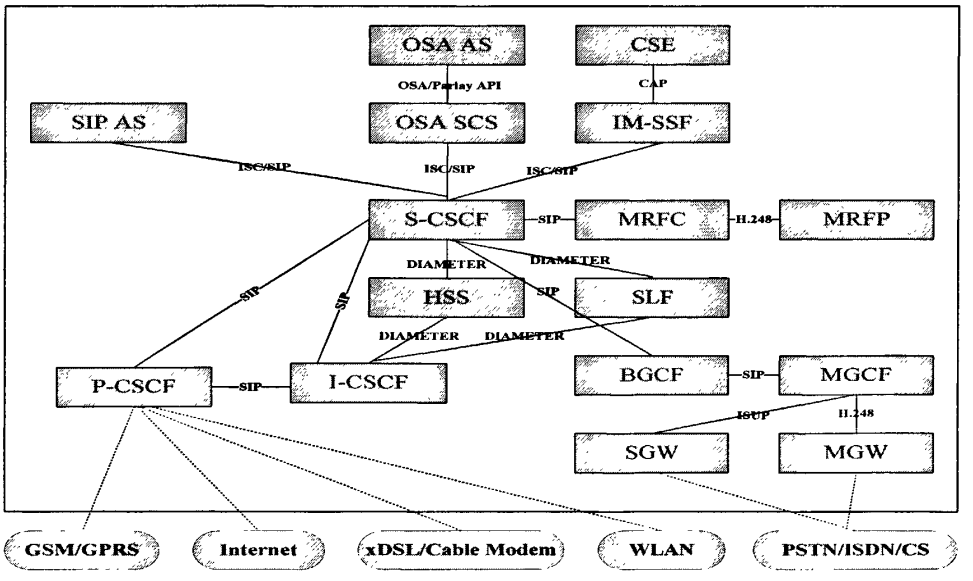

图2-2　3GPP IMS的功能实体及接口

从上图中可以看出，OSA SCS 与 IM-SSF 相当于 OSA AS 和 CSE 之间的中介网关，S-CSCF 与 SIP AS、OSA-SCS 和 IM-SSF 之间统一使用 ISC/SIP 接口；MRFC

与 MRFP 之间和 MGCF 和 MGW 之间采用 H.248 协议；S-CSCF 和 I-CSCF 与 HSS 或者 SLF 之间采用 DIAMETER 协议。IMS 其它网元之间均采用 SIP 协议。

移动办公平台架构在网络层选择 IMS 并通过 SIP 协议直接接入 IMS 中的 S-CSCF，通过 DIAMETER 协议访问 IMS 中的 HSS，因而是移动办公平台支持多媒体视频业务同时支持多个网络中的办公终端协同工作。

2.2 SOA

2.2.1 概述

人们不断提出新的软件方法论来解决现实中遇到的问题，从上世纪 70 年代提出的面向对象方法论发展到后来的面向组件再到今天的面向服务方法论。

面向对象方法论改变了人们在开发软件过程中分析问题的思维方式，在对软件系统的分析、设计时将问题域中的对象抽象成类。

面向组件方法论是把多个软件功能模块封装成一个组件，由组件提供接口供外部调用。主流的组件技术有 Sun 公司的 EJB（Enterprise Java Bean）、微软公司的 COM（Component Object Model）和由 OMG（Object Management Group）推出的 CORBA（Common Object Request Broker）。

许多企业为了提高工作效率，纷纷建设各种 IT 系统，比如 OA（办公自动化，Office Automation）系统、HR（人力资源，Human Resource）系统、SCM（供应链管理，Supply Chain Management）系统等，但是各个系统都采用自己的私有协议，无法与其他系统之间通信。为了解决这个问题，人们提出了 EAI（企业应用集成，Enterprise Application Integration）的概念，其方法就是在各个系统中建立接口适配器来完成系统之间的集成。由于日前企业系统会根据需求经常变动，因此如果 IT 系统做了改动，相应的接口也要做改动，这样为系统的维护带来很大的麻烦，同时大大增加了系统的维护成本。

EAI 无法根据用户需求做灵活改动的原因在于系统之间不是采用统一的通信方式。SOA（面向服务架构，Service Oriented Architecture）是一种新的软件架构方法，可以有效地解决这一问题。

由于 SOA 可以很好地解决系统之间的互操作问题，同时还具备模块化和接口标准化的特点，IBM、微软、HP、Oracle、BEA、Sun、SAP 都纷纷推出 SOA 产品和解决方案，这也可见 SOA 在未来的良好发展前景。

SOA 思想其实也很简单，它好比人们之间的语言沟通。如果人们之间都采用本地方言，那么也许彼此都无法理解对方，反之，如果大家都说普通话，那么彼此就很容易领会对方的意思。

实现 SOA 这一思想有很多技术，其中以 Web Services 技术为主流。Web Services 技术实现 SOA 主要包括 3 个方面：服务提供方（Service Provider）、服务注册方（Service Register）和服务请求方（Service Requestor）。服务提供方通过 WSDL（Web Services Definition Language）来描述服务；服务注册方可以发布服务，同时，调用方也可以通过 UDDI（Universal Description，Discovery and Integration）对所需服务进行查询；服务请求方就是使用 Web 服务的客户端，它可以通过 SOAP（简单对象访问协议，Simple Object Access Protocol）通过 UDDI 在服务注册方进行访问查询，查询到所需服务后再与服务注册方进行绑定，然后

就可以使用服务了。服务提供方、服务注册方和服务请求方 3 方的关系如下图所示：

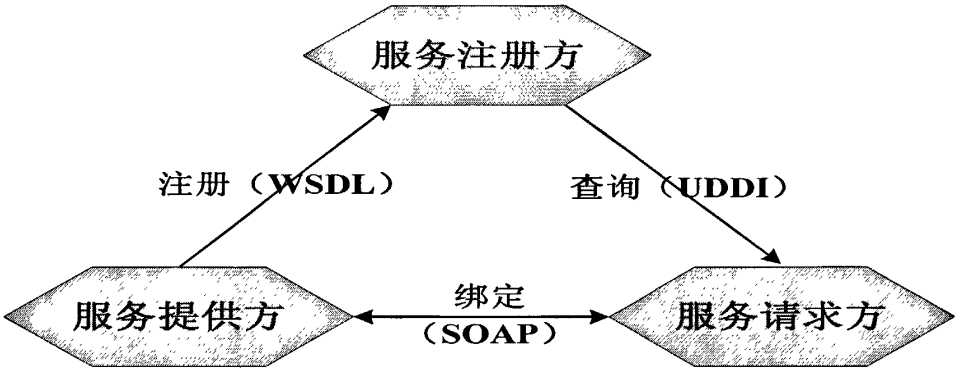

图2-3 服务提供方、服务注册方和服务请求方的关系

SOA 架构采用松耦合的方式，当业务变化是，对于其它业务系统没有什么影响，因此大大增加了应用集成的灵活性和可扩展性。SOA 能够达到松耦合架构的原因是它不像传统系统之间采用私有协议进行互操作，而是采用 SOAP、WSDL、UDDI 这样的公用协议进行互通，因此当某个系统的业务发生变化时，并不影响其它系统和它的互操作。

SOA 架构内部采用模块化的方式，各个服务以模块形式存在并且以标准化接口方式与其他模块进行通信。各个服务模块有自己的属性和服务接口，同时也可以引用其他服务。各个模块可以随意组合成新的服务模块。因此大大提高了服务生成的灵活性，提高了服务的复用性。

SOA 的另外一个优点就是可以最大化复用现有 IT 资产。在没有 SOA 之前，IT 资产被重用的方式是通过编写私有的适配器接口来实现，这样做不但要耗费大量的资金，而且对于项目的交付期限也无法保证。如果采用 SOA 方式，可以在企业现有 IT 资产的基础上进行改造，改造成面向服务的架构形式，供其它系统调用，大大提高了企业 IT 资产的复用性。

SOA 的最大好处是使得企业的 IT 系统能够以业务为核心。当今世界，企业之间竞争越来越激烈，如何在第一时间为客户提供所需业务，迅速占领市场是最为重要的。SOA 方式通过组合现有服务或者创建新的服务的方式快速生成新的业务，使得企业关注的焦点从生成各种 IT 服务转变为关注客户所需的业务上。

2.2.2 SCA

SCA 即服务组件架构（Service Component Architecture），是由 BEA、IBM、HP、Microsoft 等公司共同制定的一种 SOA 规范。

SCA 独立于具体的实现方式，它不关心服务组件使用 Java 还是用.NET 实现的。SCA 规范包括多个方面，最基本的是对一个服务组件的定义。一个服务组件通常包括服务接口，属性和引用 3 部分。

SCA 组件的服务接口以 WSDL 描述。SCA 组件的属性用于在组件初始化时，根据组件属性值的不同对组件进行相应的操作。SCA 组件的引用描述对于其它组件的引用。

另外，多个 SCA 组件还可以形成 SCA 组合，即 SCA Composite，多个 SCA

组合形成 SCA 域（Domain）。

在 SCA 内部，支持 SCA 的中间件可以同时作为 Java SCA 容器，BPEL SCA 容器或者 Spring SCA 容器等。在使用 SCA 中的组件时，可以有 Web Services、JMS、EJB Session Bean、SCA 等多种绑定方式。如下图所示：

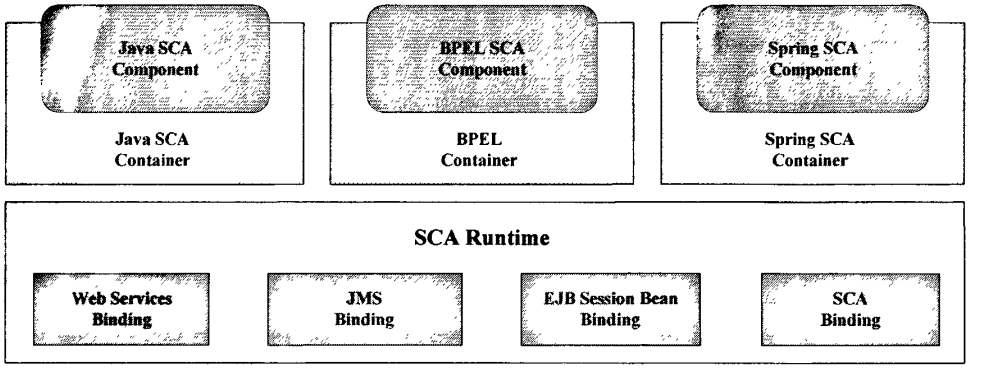

图2-4 构建SCA Runtime的一种方法

在 SCA 规范中，组件、组合的配置均在一个基于 XML 的组件定义语言（SCDL，Service Component Definition Language）进行描述。Component 项描述了组件的名称、实现该组件所使用的语言等信息，如下所示：

```
<composite ...>
  <component name="Component1" >
    <implementation.java ...>
  </component>
  <component name="Component2" >
    <implementation.bpel ...>
  <component name="Component3" >
    <implementation.spring ...>
</composite>
```

2.2.3 SDO

SDO 即 Service Data Object，是由 BEA、IBM、Microsoft、Oracle 等公司共同制定的一种数据访问规范。随着数据源类型的增多，访问数据的方式也逐渐增多，比如访问关系型数据库采用 JDBC、ODBC 的方式，访问 XML 文件数据库采用 Xquery、Xpath、XLST 等方式，访问 Web Services 数据源采用 Web Services 方式，与其它系统进行数据连接的 JCA（Java Connector Architecture）等方式。SDO 可以实现数据源访问的统一性，同时体现了数据源访问的透明性，如下图所示：

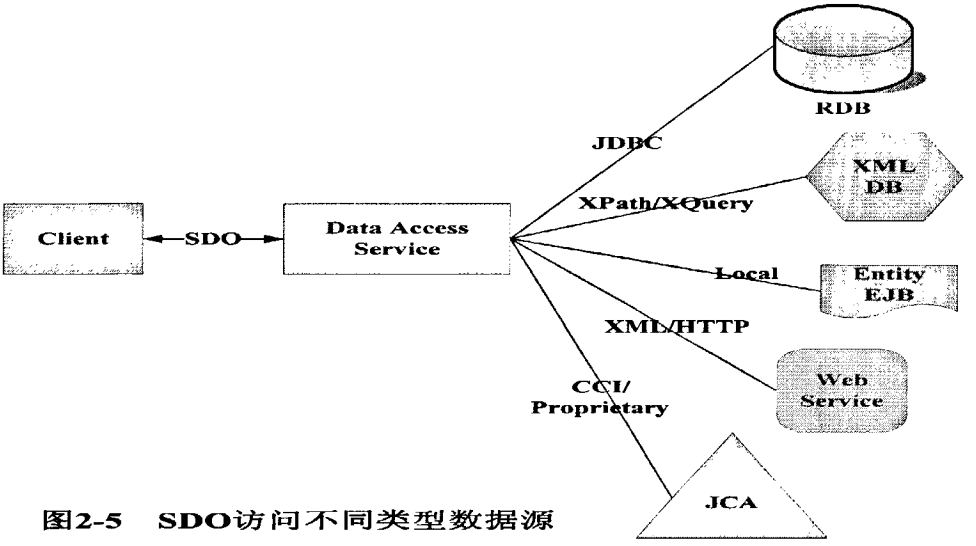

图2-5 SDO访问不同类型数据源

2.3 J2EE

2.3.1 概述

J2EE 即 Java 2 Platform for Enterprise Edition，是由 Sun 公司与 1997 年提出的针对于不同应用领域的 3 个 Java 平台之一。J2EE 是针对 Java 企业级应用而设计。

J2EE 架构包括 4 个层次：客户层、Web 层、业务逻辑层和数据层/基础层。如下图所示：

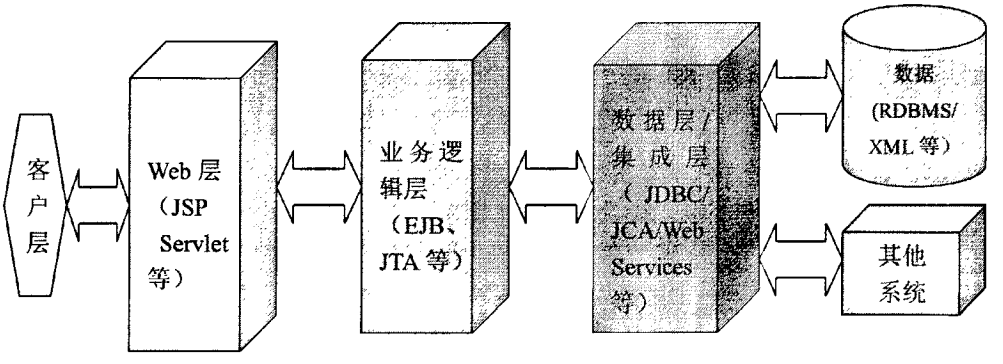

图 2-6 J2EE 层次关系

客户层主要指浏览器，用户可以通过浏览器输入网址访问基于 J2EE 的应用系统，比如新浪、搜狐、北京邮电大学的主页等。

Web 层是用户访问企业应用系统的第一层，也被称为呈现层。它主要由 JSP、Servlet 等来实现。

业务逻辑层实现了应用系统的业务逻辑部分，通常采用 EJB、JTA（Java 事

务 API，Java Transaction API）等来实现。业务逻辑层可以采用分布式部署方式，它通过 JTA 来保证业务逻辑的正常执行。

数据层或者集成层是基于 J2EE 架构的应用系统与数据库或者其他应用系统系统的互通层，由 JDBC、JCA、Web Services 等协议组成。可以通过 JDBC 访问关系型数据库，通过 Web Services 与其它系统集成。

在移动办公平台架构中，Web 层通过 SIP Servlet 辅助实现，保证 SIP 消息与业务处理类中的方法之间的映射。业务逻辑层通过 SCA 辅助实现，保证业务逻辑层实现的灵活性。数据层通过 SDO 辅助实现，保证数据源访问的透明性。

2.3.2 JSP

JSP(Java Server Pages)是由 Sun 公司倡导、许多公司参与一起建立的一种动态网页技术标准，用以帮助 Web 内容开发人员创建动态网页，并且只需要相对较少的代码。JSP 页面由 HTML 代码和嵌入其中的 Java 代码所组成。服务器在页面被客户端所请求以后对这些 Java 代码进行处理，然后将生成的 HTML 页面返回给客户端的浏览器。

2.3.3 Servlet

Servlet 是一种小型的 Java 程序，它扩展了 Web 服务器的功能。作为一种服务器端的应用，当被请求时开始执行，这和 CGI Perl 脚本很相似。Servlet 和 CGI 脚本的一个很大的区别是：每一个 CGI 在开始的时候都要求开始一个新的进程，而 Servlet 是在 Servlet 容器中以分离的线程来运行的。因此 Servlet 在可伸缩性上提供了很好的改进。Servlet 提供的功能大多与 JSP 类似，不过实现的方式不同。JSP 通常是大多数 HTML 代码中嵌入少量的 Java 代码，而 Servlet 全部由 Java 写成并且生成 HTML。

2.3.4 JDBC

JDBC(Java Database Connectivity)以一种统一的方式来对各种各样的数据库进行存取。和 ODBC 一样，JDBC 为开发人员隐藏了不同数据库的不同特性。另外，由于 JDBC 建立在 Java 的基础上，因此还提供了数据库存取的平台独立性。

2.3.5 EJB

EJB 即 Enterprise Session Bean，它在 J2EE 技术架构中主要完成业务逻辑处理功能。

EJB 与 JavaBean 是不同的。EJB 是一种分布式组件，可以完成分布式事务处理；JavaBean 是一个简单组件，可以封装对象的属性和方法，不支持分布式事

务处理，一般用于 J2EE 的 Web 层实现简单的业务逻辑。

EJB 包括 3 中类型：会话 Bean（Session Bean）、实体 Bean（Entity Bean）和消息驱动 Bean（MDB，Message Driven Bean）。

Session Bean 在业务处理中完成会话功能，分为有状态（stateful）和（stateless）两种类型。有状态的 Session Bean 在建立会话过程中记录 Session 的状态，比如这次 Session 中包含的用户名、密码、联系方式等，Stateful Session Bean 的优点是可以记录会话的状态。Stateless Session Bean 在会话过程中不记录会话状态，只执行相应的业务逻辑，因此占用内存空间少，执行效率高。

Entity Bean 代表一个实体对象，通常完成与数据库中表之间的数据操作，它分为容器管理的持久化（CMP，Container Managed Persience）和 Bean 管理的持久化（BMP，Bean Managed Persience）。CMP 指的是数据库的持久化工作有支持 Entity Bean 运行的中间件容器来管理，这样减轻了开发者的工作量。BMP 指的是数据库的持久化工作由支持 Entity 运行的中间件容器来管理，这给 Bean 的开发者带来了复杂性，但是开发者可以根据自身的需要开发出功能比 CMP 强大的 Entity Bean。

MDB 是专门用于基于消息的应用的。MDB 需要与 JMS（Java Message Service）配合才能够实现，

在多媒体视频会议系统中，采用 Stateless Session Bean 来实现控制类，采用 CMP 实现系统中的实体类。

2.4 J2ME

2.4.1 概述

J2ME 即 Java 2 Platform for Micro Edition（Java 2 平台袖珍版），由 Sun 公司 1999 年推出，它是针对消费类电子设备、嵌入式设备设计的。Sun Microsystems 将 J2ME 定义为“一种以广泛的消费性产品为目标的的高度优化的 Java 运行时环境，包括寻呼机、移动电话、可视电话、数字机顶盒和汽车导航系统.

2.4.2 J2ME 总体架构

J2ME 使用配置和简表定制 Java 运行时环境 (JRE)。作为一个完整的 JRE，J2ME 由配置和简表组成，配置决定了使用的 JVM，而简表通过添加特定于域的类来定义应用程序。

配置将基本运行时环境定义为一组核心类和一个运行在特定类型设备上的特定 JVM。我们将在 J2ME 配置一章中详细讨论配置。

简表定义应用程序；特别地，它向 J2ME 配置中添加特定于域的类，定义设备的某种作用。我们将在 J2ME 简表一章中深入介绍简表。

下面的图表描述了不同的虚拟机、配置和简表之间的关系。它同时把 J2SE API 和它的 Java 虚拟机进行了比较。虽然 J2SE 虚拟机通常被称为一种 JVM，但是 J2ME 虚拟机、KVM 和 CVM 都是 JVM 的子集。KVM 和 CVM 均可被看作是一种 Java 虚拟机 -- 它们是 J2SE JVM 的压缩版，并特定于 J2ME。

2.4.3 配置简述

配置将基本运行时环境定义为一组核心类和一个运行在特定类型设备上的特定 JVM。本文只介绍连接限制设备配置 (CLDC)，CLDC 是与 KVM 一起用于内存有限的 16 位或 32 位设备。这是用于开发小型 J2ME 应用程序的配置（虚拟机）。CLDC 同时还是用于开发绘图工具应用程序的配置。 Palm 电脑便是一个运行小应用程序的小型无线设备的示例。

2.4.4 简表简述

简表定义了您的应用程序所支持的设备类型。特别地，它向 J2ME 配置添加了特定于域的类来定义设备的某种作用。简表建立在配置的顶部。已经为 J2ME 定义了两种简表：KJava 和移动信息设备简表 (MIDP)，它们也被建立在 CDLC 上。这两种简表适用于小型设备。

有一种纲要简表，您可以在它的上面创建自己的简表，这种纲要简表也称为基础表，可供 CDC 使用。然而，在本教程中，我们只重点介绍建立在 CLDC 顶部，适用于小型设备的简表。我们将在后面的文章中讨论上述这些简表，还会使用 KJava 和 MIDP 建立一些示例应用程序。

2.4.5 J2ME 目标设备

使用 CLDC 开发的 J2ME 应用程序的目标设备通常具有以下特征：

- 可供 Java 平台使用的 160 到 512 千字节的总内存
- 功率有限，常常是电池供电
- 网络连通性，常常是无线的、不一致的连接并且带宽有限
- 用户接口混乱，程度参差不齐；有时根本就没有接口
- 一些 CLDC 支持的设备，包括无线电话、寻呼机、主流个人数字助手 (PDA)，以及小型零售支付终端。

2.5 SIP

SIP (Session Initiation Protocal)称为会话初始协议，是由 IETF(Internet Engineering Task Force)组织于 1999 年提出的一个在基于 IP 网络中，特别是在 Internet 这样一种结构的网络环境中，实现实时通讯应用的一种信令协议。它借鉴在因特网中得到成功应用的 HTTP 和 SMTP 的优点发展而来，因此具有简单、灵活的特点。由于其采用文本形式，并且具有良好的扩展性，因此无论移动网推崇的 IMS，还是固网拥护的 NGN，都采用其作为网络内部的通信协议。

而所谓的会话(Session)，就是指用户之间的数据交换。在基于 SIP 协议的应用中，每一个会话可以是各种不同类型的内容，可以是普通的文本数据，也可以是经过数字化处理的音频、视频数据，还可以是诸如游戏等应用的数据，应用具有巨大的灵活性。

作为一个 IETF 提出的标准，SIP 协议在很大程度上借鉴了其他各种广泛存在

的 Internet 协议，如 HTTP(超文本传输协议)、SMTP(简单邮件传输协议)等，和这些协议一样 SIP 也采用的基于文本的编码方式，这也是 SIP 协议同视频通讯领域其他现有标准相比最大的特点之一。

SIP 协议的提出和发展，是伴随着 Internet 的发展而发展的，到目前为止它走过了一下几个阶段： n1996 年首先出现了 SIP 的概念，这时 SIP 的主要应用是针对 Internet 上的各种文本应用，如电子邮件、文字聊天等； n1999 年 3 月，ITEF 的多方多媒体会话控制(MMUSIC)工作组提出了 RFC2543 建议，供各厂商和机构讨论； n1999 年 9 月，SIP 工作组从 MMUSIC 中分离并独立出来，成立了 SIP 工作组，并与 2000 年 7 月发表了 SIP 的草案； n2002 年 6 月，ITEF 的 SIP 工作组又发表了 RFC3261 建议，以取代 RFC2543。

由于网络环境以及相关多媒体技术的不足，在 SIP 协议首次提出的时候，仅仅针对各种文本应用，随着技术的发展，并通过和 IETF 中 IP 电话工作组(IPTEL)、IP 网中电话选路(TRIP)工作组等兄弟工作组配合工作，在 SIP 协议中大大加强了对多媒体通讯的支持。

由于 Internet 的飞速发展，在最近的两年时间内，SIP 已经开始被 ITU-T SG16、ETSI TIPON(欧洲标准化组织)，IMTE 等各种标准化组织所接受，并在这些组织中成立了与 SIP 相关的工作组。特别是作为 ITU-T SG16 主要成员，在多年发展 H323 应用的基础上，针对 SIP 应用在视频领域的特点，提出了 SIP 的应用指导，并推出了相应的 SIP 协议栈，使得 ITU 的成员实现了这两种协议之间的互通性。

SIP 协议采用类似于 Email 地址的形式表示会话双方的地址，例如 sip:lifudong@bupt.edu.cn。采用文本方式描述会话信息，包括起始行，消息头，消息体三部分，如下所示：

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <bob@biloxi.com>
From: Alice <alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Date: Thu, 21 Feb 2002 13:02:03 GMT
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 147

v=0
o=UserA 2890844526 2890844526 IN IP4 here.com
s=Session SDP
c=IN IP4 pc33.atlanta.com
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

起始行

消息头

回车换行

消息体

SIP 消息的起始行分为两种类型：请求行和响应行。请求消息分为 6 种基本类型：REGISTER、INVITE、ACK、BYE、CANCEL、OPTIONS，这些是必须

支持的。另外还有一些扩展信令，比如 PRACK、SUBSCRIBE、INFO、MESSAGE 等，它们都是根据特定需要拓展的。REGISTER 信令的功能是完成 SIP 用户的注册；INVITE 用于呼叫被叫用户；ACK 用于对被叫摘机返回的 200OK 进行确认，从而完成 3 次握手；BYE 信令用于完成通话后发出挂机消息；CANCEL 用于主叫用户为建立通话连接而取消呼叫；OPTIONS 用于查询被叫用户的能力信息。

SIP 消息的响应分为 6 中类型：1xx、2xx、3xx、4xx、5xx、6xx。1xx 消息代表临时响应消息，比如 100 trying，180 ringing 等；2xx 代表成功的消息，比如 200 OK；3xx 代表重定向消息；4xx 代表客户端请求有错；5xx 代表被请求的服务端出错；6xx 代表全局错位。

SIP 的消息体可以是普通文本，也可以是 SDP（会话描述协议、Session Description Protocol）。SDP 描述发起呼叫的一方的媒体能力，比如描述音频编码、音频数据传输协议、传输端口等。

2.6 本章小结

本章全面介绍了设计和实现多媒体会议系统用到的关键技术。IMS 是实现移动办公平台的网络基础，体现了不同网络中的终端的协同性；SOA 是系统集成的以服务为中心的松耦合的软件架构方式，SCA 是面向服务架构中组件的实现规范，SDO 是面向服务架构中数据访问的规范；J2EE 和 J2ME 分别是是 Java 技术针对企业级和移动设备级的两种技术架构，基于它们建立的应用都具有跨平台性和良好的可扩展性；SIP 是 IMS 网络内部网元之间通信的标准协议，同时它也可以用于 IMS 网络外部，SIP 用于应用系统和 IMS 网络之间可以方便应用系统平滑接入 IMS 网络，用于移动终端和 IMS 网络之间可以提高移动业务的智能性；WebLogic SIP Server 是基于 Java 技术的中间件，可以辅助开发和运行，它实现了 J2EE 规范，同时支持 SIP 和 DIAMETER 等协议。

3 第三章 移动办公平台架构设计

本章通过分析移动办公业务及平台的发展要求，结合 IMS、SOA、SCA、SDO、J2EE、J2ME、SIP 等先进的技术和设计方法，提出了移动办公平台架构模型。接着把移动办公平台总体架构模型中分为服务器端和客户端两部分进行详细的分析并给出设计方案。

3.1 架构设计分析

当前的移动办公平台有可伸缩性、可重用性、可维护性、可靠性等几个方面的基本要求，同时要求新型的移动办公平台能够支持多种办公终端协同办公、支持多媒体视频业务、跨平台性、新业务生成快速灵活、方便与其它业务系统集成互通以及移动终端业务有一定的智能性。

要达到以上要求，首先要求移动通信网络传输带宽高到足以传输多媒体视频数据，同时网络架构灵活，实现接入层、控制层和业务层的分离，为移动办公终端协同办公以及新业务的快速形成打下基础。IMS 是 3G 核心网的子系统，3G 无线网络在用户步行时支持 384Kbps 的速率，室内支持高达 2Mbps 的速率，能够满足传输多媒体视频数据的要求。另外，IMS 采用分层的架构方式，将核心网子系统 IMS 分为接入层、承载层、控制层和业务应用层，便于移动办公平台接入 3G 网络，支持 GSM、GPRS、3G、Internet、WLAN、PSTN、ISDN 等多种网络中的办公设备协同工作。

其次，要到满足对于移动办公平台的跨平台性要求，基于 Java 技术的 J2EE 和 J2ME 一方面采用虚拟机技术屏蔽了硬件和操作系统的差异性，为移动办公平台的跨平台性提供了前提，另一方面，J2EE 和 J2ME 采用分层的架构方式，J2EE 实现了表现层、业务逻辑层、数据层/集成层的分离、J2ME 实现了配置层、简表层和业务应用层的分离，为适应移动办公业务的变化和新业务的实现打下了基础。

再次，应该采用成熟先进的应用中间件，一方面这些中间件能满足系统可伸缩性、可重用性、可维护性和可靠性几个方面的基本要求，同时应用中间件实现了 SIP、J2EE 和 J2ME 等技术规范，对于移动办公平台接入 IMS 和移动办公业务的快速实现创造条件。

应用中间件处于硬件和操作系统之上，通过软件集群方式达到可伸缩性、可靠性和可扩展性几个方面的要求。当随着系统的用户增多，现有系统无法处理更多的客户请求，这时可以通过配置和部署软硬件设备和应用软件来提高系统的性能和吞吐量。当集群中有某个硬件或者软件宕机后，集群中其它正常工作的中间件系统能够自动切换过来，实现应用级会话和业务状态的自动迁移，从而达到提高系统可靠性的目的。由于应用中间件支持的 J2EE 或者 J2ME 技术规范采用分层的架构方式，因而当修改现有业务或者开发新的业务时，只需要和业务实现有关的相应层做小的修改即可，提高了系统的伸缩性和扩展性。

最后，应该设立专门的组件层来满足系统复用性和与其他应用系统集成的要

求。目前的组件规范主要有微软公司的 COM、Sun 公司的 EJB 和 OMG（Object Management Group，对象管理组织）的 CORBA，这些组件规范虽然提高了系统的复用性，但是 COM 和 EJB 局限在某一种技术平台之上，而 CORBA 规范虽然走中立路线，但是并没有得到广泛支持与应用。SOA 是由 IBM、微软、BEA、Oracle 等跨国软件公司提出的概念和相关规范，其主要思想是它可以使得软件系统之间采用松耦合的架构形式，同时遵循基于 XML 语言的协议标准，为系统互通和集成提供了便利，目前 SOA 的主要实现技术是 Web Services。在移动办公平台的架构中，组件层采用 SOA 中的两个规范 SCA 和 SDO。SCA 和 SDO 都是采用面向服务的方式，SCA 规范是服务组件设计规范，主要目标是在面向服务的架构下采用组件的形式实现软件的复用，是面向服务与面向组件设计思想的结合，SDO 规范是服务数据的设计规范，其主要目标是在面向服务的架构下实现客户端对于数据源访问的透明性，这些数据源可以是 RDBMS，也可以是 XML、Web Services 等其它数据源。

移动办公平台架构中还应当包括对安全管理、QoS 管理和计费管理等方面的研究，这些并不在本文讨论的范围之内。

3.2 总体架构设计

移动办公平台架构在总体上包括服务器端和客户端两部分，分为为网络层、硬件与操作系统层、中间件层、组件层和业务应用层。

在移动办公平台架构中的网络层主要考虑到两方面的要求：第一、网络带宽能够传输多媒体视频数据。第二、移动办公平台实现与移动通信网络的无关性。

移动办公平台架构中的应用中间件层主要考虑以下三个方面的要求：第一、要方便移动办公业务的开发、配置和部署。第二、支持移动办公平台的可伸缩性、可维护性、可靠性和可扩展性。第三、支持最新 J2EE、SIP、DIAMETER 和 J2ME 规范的实现。

在移动办公平台架构中的组件层主要考虑以下三个方面的要求：第一、满足系统高复用性的目标。第二、实现面向服务的架构形式。第三、数据源访问的透明性。

通过以上分析，构想的移动办公平台总体架构图如下所示：

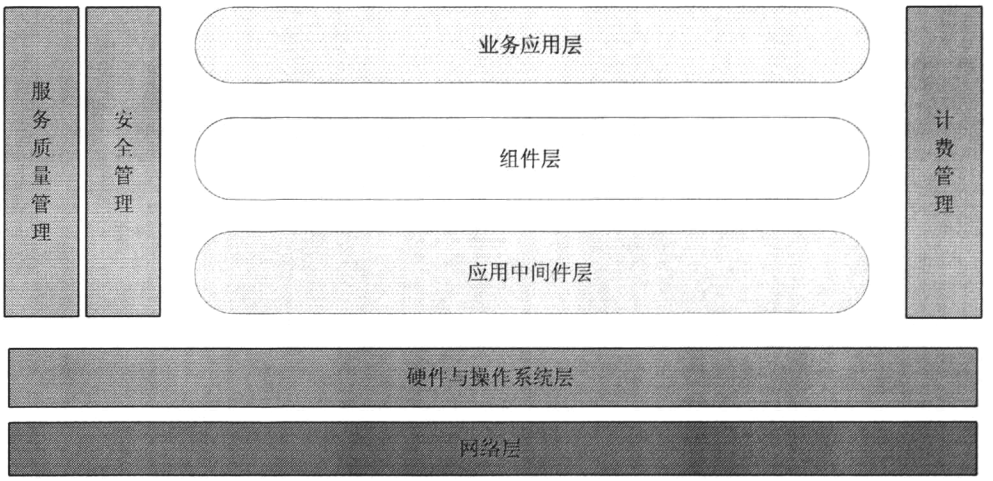

图3-1 移动办公平台总体架构

在移动办公平台中包括服务器端和客户端两部分，它们之间采用 SIP 协议通

信。一方面，移动办公平台服务器端通过 SIP 协议接入 IMS 网络（第一个接入点是 IMS 网络中的 S-CSCF 网元），另一方面移动办公平台客户端位于移动终端之上，通过 3G 无线接入网接入 IMS 网络（第一个接入点是 IMS 为了中的 P-CSCF）。SIP 协议基于文本，具有的简单、可扩展性好以及可以动态修改会话控制的特性，支持基于移动终端开发智能化的移动办公业务。

总之，移动办公平台架构模型具有的先进性如下：第一、采用分层架构形式，使应用开发、配置和部署具有很大的灵活性。第二、基于 IMS 技术架构设计，支持多种网络中的办公终端协同办公，同时支持多媒体视频业务。第三、应用中间件层实现了 J2EE 和 J2ME 技术架构，使得移动办公平台具有跨平台性、可扩展性、可维护性、可伸缩性和高可靠性，以上特性也是成熟的中间件应当具有的。除此之外，它还支持 Web Services 技术规范，可以与其他应用系统之间建立松耦合的面向服务架构，从而为移动办公应用系统与其它应用系统之间集成互通打下了基础。第四、在应用中间件层之上专门设计的组件层，可以通过预先设计的通用组件提高系统的复用性，也可以通过 SDO 的访问方式，实现数据源访问的透明性。

3.3 服务器端设计

移动办公平台服务器端分为网络层、硬件和操作系统层、应用中间件层、组件层和业务应用层。

网络层基于 3G 核心网子系统 IMS，可以融合多种接入网，因此支持移动办公终端与其它办公终端协同办公，同时具有支持多媒体视频业务的能力。

硬件和操作系统层不限定哪一种服务器硬件和操作系统，只要其上有适应该服务器硬件和操作系统的 Java 虚拟机即可。

应用中间件层处于服务器硬件和操作系统之上，应当实现 J2EE 规范，同时实现保证移动办公平台接入 IMS 网络的 SIP、DIAMETER 等规范。

组件层基于 SCA 组件服务和 SDO 数据服务两种规范实现，SCA 组件服务保证系统功能组件级的复用，SDO 数据服务保证组件能够在外界数据源发生变化时不做改动。

业务应用层是基于应用中间件层和组件层实现的移动办公业务，包括移动多媒体视频会议、网页同址浏览、移动电子白板共享等业务。

通过以上分析，构想的移动办公平台服务器端架构如图 3-2 所示：

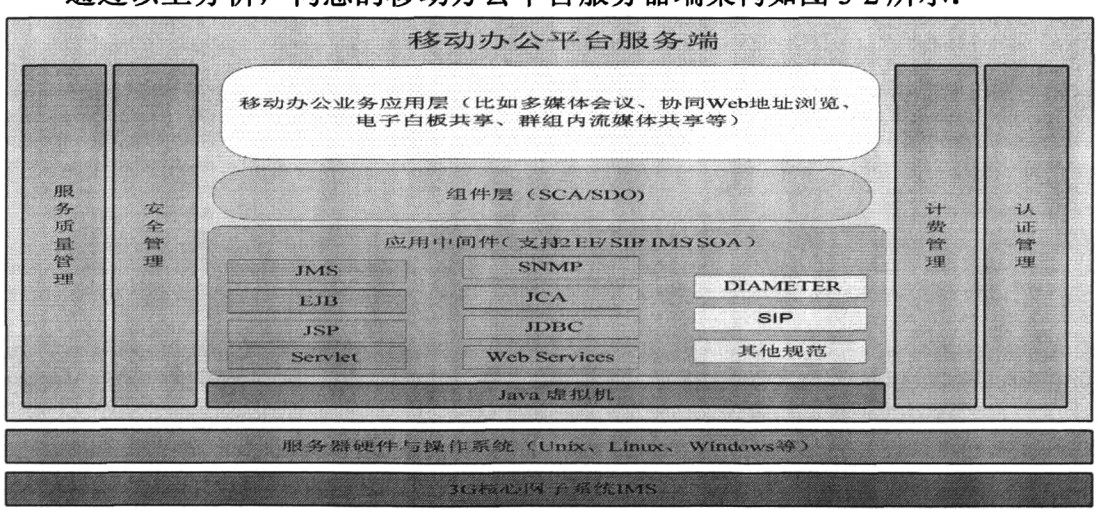

图3-2 移动办公平台服务器端架构

从上图中可以看出，3G 核心网子系统 IMS 处于移动办公平台架构的最底层，IMS 一方面支持包括移动终端在内的多种终端接入，另一方面也支持移动办公平台的接入。在 IMS 网络之上是服务器硬件和操作系统，包括 Unix、Linux 或者 Windows 这样的操作系统。

在操作系统之上是由 Java 虚拟机支撑的应用中间件层，它使得移动办公平台具有操作系统平台无关性，同时支持 J2EE 规范和 SIP 规范，在应用中间件之上可以建立 B/S 结构 Internet 应用，也可以建立 C/S 结构的应用。

在应用中间件层之上是组件层，组件层采用 SCA 和 SDO 规范实现。一方面可以建立服务组件，方便系统内部调用，提高系统的复用性，另一方面可以建立数据服务，实现各种数据源访问的透明性。

3.4 客户端设计

移动办公平台客户端存在于移动终端嵌入式操作系统平台之上，主要功能是辅助开发个性化的移动办公业务。

本课题中的移动办公平台架构设计主要达到跨平台性和高复用性两个目标。在跨平台性方面，客户端架构中应用中间件层实现了 J2ME 技术规范。由于 J2ME 基于 Java 虚拟机，它屏蔽了各种嵌入式操作系统的差异性，因而移动办公业务应用具有平台无关性。另外，J2ME 架构内部采用分层的方式，分为配置层、简表层和应用层，只要遵循 JTWI（Java 技术 for 无线工业）标准就可以使应用系统具有良好的移植性。

在应用中间件层之上是组件层。设置组件层的目的是为了通过预先开发的通用业务组件提高系统的复用性进而缩短业务生成的时间。

在移动办公平台客户端两侧有安全管理和认证管理功能，主要目的是提高移动办公业务的安全性，这并不属于本课题研究的主要内容。

通过以上分析，构想的移动办公平台客户端架构如下图所示：

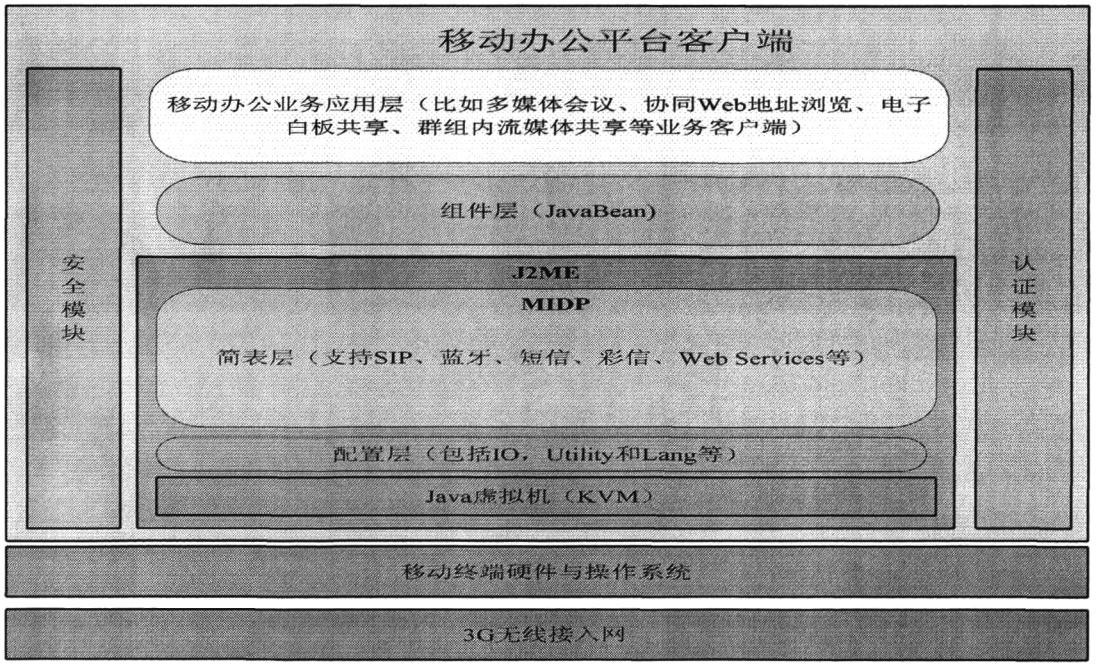

图3-3 移动办公平台客户端架构

从图 3-3 可以看出，移动办公平台客户端架构中的网络层基于 3G 无线接入网，而 3G 无线接入网可以提供足以传输多媒体视频数据的网络带宽，因此为承载多媒体视频业务提供了条件。

网络层之上是移动终端硬件和操作系统，移动终端操作系统有 Symbian、Linux、Windows Mobile 这样的嵌入式操作系统，移动终端形式有 PDA、智能手机、普通手机等。

硬件和操作系统层之上是应用中间件层，在移动办公平台客户端架构中采用支持 J2ME 规范的中间件。J2ME 的配置层包括 IO、Lang、Utility 等基本的功能，同时包括的 KVM 屏蔽了嵌入式操作系统的差异性，使得移动办公业务客户端具有可移植性。简表层主要是特定于移动终端的特性，为开发个性化移动办公业务提供支持，但是只要应用中间件实现的开发工具包符合 JTWI 标准，就能够在不同的移动终端之间移植，在设计和开发移动终端客户端软件时要注意遵循这一标准，提高移动办公业务的可移植性。

3.5 本章小结

本章首先由移动办公业务的发展要求和移动办公平台架构设计的要求进行分析，结合当今先进的移动通信技术、信息技术和架构设计方法，给出了移动办公平台架构的总体设计方案。移动办公平台的总体架构模型分为网络层、应用中间件层、组件层和业务应用层。

移动办公平台的架构分为服务器端和客户端两部分。服务器端架构在网络层采用 3G 核心网子系统 IMS。应用中间件层采用实现 J2EE、SIP、DIAMETER 等规范的中间件，不仅能够无缝接入 IMS 网络，同时支持 B/S 结构的 Internet 应用，还能体现移动办公平台的可靠性、可伸缩性、可维护性和可扩展性。组件层的目的是提高系统的复用性。移动办公平台架构客户端主要实现了跨平台性和复用性两个目标。

4 第四章 移动办公平台架构实现

上一章关于移动办公平台架构模型的抽象概念，目的是为移动办公平台的搭建提供一种指导。本章根据移动办公平台架构模型的各个部分的要求，首先给出办公平台接入 IMS 网络的方法，然后结合当前成熟的中间件产品给出移动办公平台的实现方案。

4.1 接入 IMS 的方法

移动办公平台架构中的网络层选择了 3G 核心网子系统 IMS 下面说明移动办公平台服务器端是如何接入 IMS 网络的。

IMS 技术规范描述了 IMS 具有融合包括 GSM、GPRS、WCDMA、PSTN、Internet、WLAN 等多种不同类型的网络的能力，这为处于不同网络内部的终端的协同办公创造了条件。为了便于叙述移动办公平台的协同性，在这里只描述处于 3G 网络中的移动办公终端（3G 手机）与处于因特网中的桌面办公终端（PC 机）为实现协同办公所经历的过程。

3G 手机如果通过 IMS 网络与 PC 终端进行协同办公，首先要经过 3G 无线接入网，3G 无线接入网通过 3G 核心网的分组交换域接入 3G 核心网的 IMS 域的第一个网元 P-CSCF。IMS 通过网元 S-CSCF 和 HSS 接入移动办公平台。其次，移动办公平台根据业务类型执行相应的业务逻辑。最后，移动办公平台依次通过 IMS、3G 分组域网络、Internet，在 Internet 中通过路由器和交换机等设备找到对方的 PC 办公终端。3G 手机借助 IMS 网络协同办公的详细流程如图 4-1 所示：

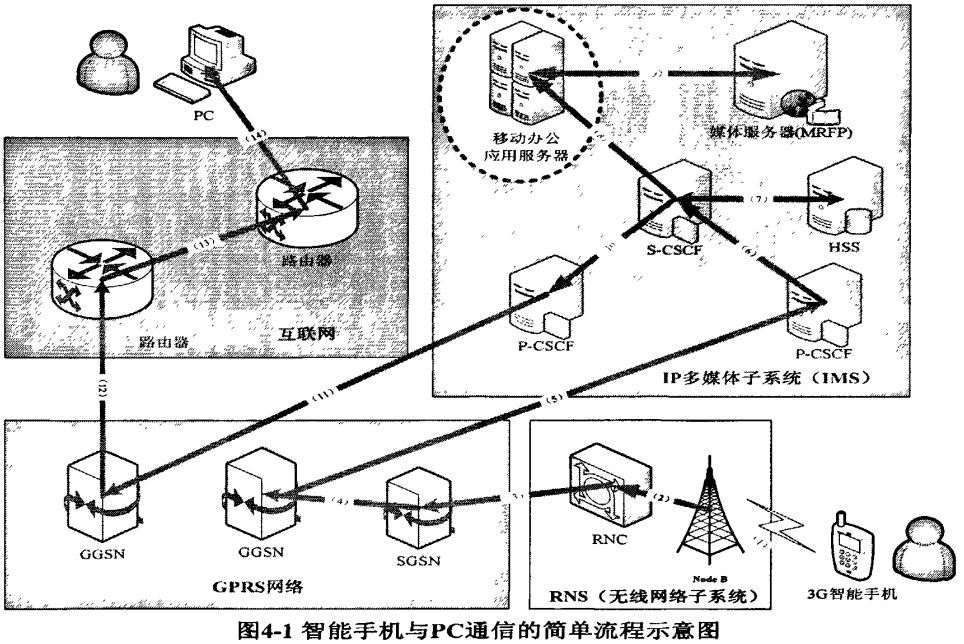

图4-1 智能手机与PC通信的简单流程示意图

4.2 服务器端实现

移动办公平台服务器端架构中的应用中间件层不但实现了 J2EE 技术规范，而且实现了 SIP、DIAMETER 等规范。本节首先介绍应用中间件的选择，然后介绍基于该中间件进行移动办公业务开发的方法。

4.2.1 应用中间件选择

目前支持以上规范的成熟中间件有很多，其中有 BEA 公司的 WebLogic SIP Server、IBM 公司的 Websphere、Oracle 公司的 Oracle Application Server、Apache 的开源中间件 Tuscany 等。WebLogic 是性能领先并且应用广泛的中间件产品，在电信企业的市场占有率为 80%以上，本课题选择它最为移动办公平台架构中的应用中间件，下面就对其做一个简单介绍。

BEA WebLogic SIP Server 是行业内第一款，也是唯一一款融合 Java EE-SIP-IMS-SOA 应用服务器，拥有强大的功能和标准，其可用性、可靠性、可伸缩性和性能无可比拟。它是 BEA WebLogic Communications Platform 产品系列的 IMS-SIP 应用服务器组件，其 NEP、SI 和 ISV 合作伙伴遍布全球，并且已将 BEA WebLogic SIP Server 作为其新一代产品的标准 IMS-SIP 应用服务器基础。

BEA WebLogic SIP Server 是融合的互联网-IMS 应用容器，符合 SIP Servlet API (JSR 116) 标准，后者是用于开发和执行基于 SIP 的应用的 Java EE 编程模型。为 BEA WebLogic SIP Server 开发的 SIP Servlet 应用还可以利用一些附加 IMS 特定功能，例如通过 Diameter 协议实时访问用户资料。BEA WebLogic SIP Server 3.0 基于 Java EE 1.4 和 Java SE 5，为 JDBC、JMS、JNDI、RMI/IIOP、EJB、Web 服务和 IPv6 等功能提供了支持。BEA WebLogic SIP Server 还支持 BEA WebLogic Server 9.2 中提供的整套 Web 服务和 SOA 功能。

- IP 多媒体子系统(IMS)服务层的基础

随着全球的网络运营商部署 IMS 网络，创建和开发基于 IMS 的新服务变得越来越重要。新的 IMS 服务部署在 IMS 核心网络之上的一层（通常被称为 IMS 服务层）。尽管有一些行业标准为 IMS 核心网络指定了架构，但实施 IMS 服务层却不存在任何标准。

3GPP IMS 参考架构指定了标准接口，IMS 应用服务器需要用于支持会话管理、用户简表和计费等功能。所谓的 IMS 服务控制 (ISC) 就是标准接口，IMS 应用服务器可以通过它与 IMS 核心网络的会话控制功能 Serving CSCF 进行集成。3GPP 还指定了一个称为 Sh 的标准接口，它以 IETF Diameter 协议为基础，定义了 IMS 应用服务器如何与称为归属用户服务器(HSS)的用户资料数据进行交互。BEA WebLogic SIP Server 支持 ISC 和 Sh 接口以及由 3GPP 第 6 版规范强制要求的 SIP Profile。

- 运营商级的可靠性和性能

BEA WebLogic SIP Server 使用 N+1 方法来实现高可用性，其中所有集群成员都完全冗余，可以在另一个集群成员失效时立即用于处理通信。BEA WebLogic SIP Server 集群架构以透明方式提供的应用不但具有高可用性，还具有运营商级别的会话保留功能，从而确保一个集群成员的故障不会影响现有会话。

随着 BEA WebLogic SIP Server 3.0 的推出，BEA 在 SIP servlet 引擎层缓存和 IMS 连接池方面的市场领先性能进一步得到了增强。SIP servlet 引擎层上的缓存允许 SIP 应用更快速地访问会话状态数据，而 IMS 连接池则允许 BEA WebLogic SIP Server 上托管的 IMS 应用更快速地与 IMS 核心网络组件建立连接，例如 Serving-CSCF 和会话边界控制器(SBC)。

BEA WebLogic SIP Server 在提高性能的同时还缩短了反应时间，不但增强了 SIP-IMS 应用的用户体验，还降低了系统管理的 TCO（Total Cost of Ownership)。

■ 无可比拟的高可用性

为解决运营商对高可用性的需求，BEA WebLogic SIP Server 为基于 SIP 的应用提供了无可比拟的高可用性和可靠性。当网络运营商和服务提供商启动融合的互联网-IMS 服务时，实现最终用户满意度、接受度以及收入流增长的一项关键因素就在于这些新服务可靠性和可用性。如果某家运营商的区域数据中心由于未预见到的灾难性故障而脱机，损失的就将是用户和收入。通过在多个区域数据中心之间提供 SIP 会话状态复制，BEA WebLogic SIP Server 可以使客户满意度得到提高。

■ 传统电话支持

大多数融合的互联网-IMS 应用都需要与传统电话网络（通常称为 PSTN，公共交换电话网）建立连接。此类连接的常见使用案例包括点击拨号服务，与基于 Web 的服务进行交互的用户希望通过它与使用 PSTN 电话号码的另一位用户建立 VoIP 会话。这种形式可能需要通过 IP 来传输 PSTN 信号，或者允许 SIP 应用呼叫常规 PSTN 电话号码 (ENUM:E.164 Number Mapping)。

BEA WebLogic SIP Server 为在 SIP (RFC 3824) 中使用 SCTP（RFC 2960：流控制传输协议）和 ENUM 提供支持，从而允许 IMS-SIP 应用在本机访问 SIP 应用中的 PSTN 信号和 ENUM 使用情况。

■ 全面的行业标准交叉支持

BEA WebLogic SIP Server 的核心架构是根据关键 Java、互联网、Web 服务和 IMS 行业标准的全面集合，实施单一的融合应用容器。BEA WebLogic SIP Server 中支持的标准范围广泛、各有不同，为运营商和开发人员创建新型融合互联网-IMS 服务提供了功能丰富的环境。通过提供单一应用容器架构，BEA 使网络运营商能够保持比其它厂商更低的 TCO 结构。

此外，BEA 还在基于 Java 的 SIP 开发人员和厂商社区中发挥着领导作用，率先为新的 SIP Servlet API 版本(JSR 289)制订了规范。

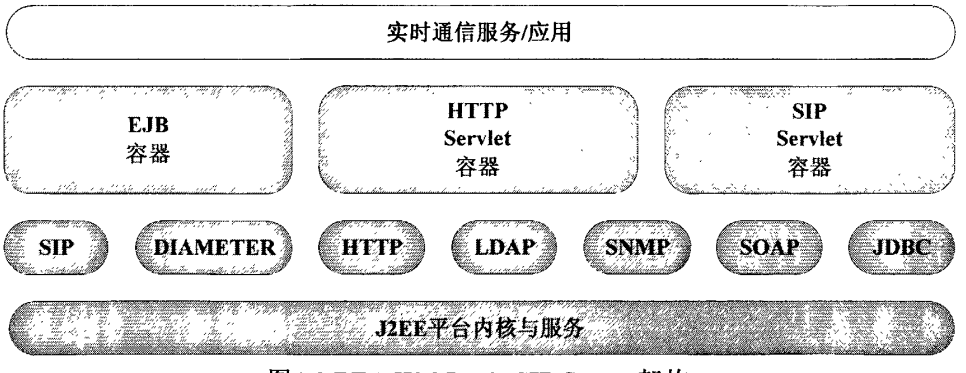

图4-2 BEA WebLogic SIP Server架构

总之，WebLogic SIP Server 的以上特性，对于移动办公平台在中间件层的实现提高了良好的支持，满足移动办公平台架构对于可靠性、可伸缩性、可用性和性能的要求，还可以满足移动办公终端与其他多种办公终端间的协同办公以及支持多媒体视频业务的要求。

4.2.2 业务实现方法

移动办公业务控制的实现采用基于 SIP 消息的方式。SIP 消息的处理由对应的 SIP Servlet 来实现，SIP 消息与 SIP Servlet 的对应关系通过 XML 文件配置完成。

SIP Servlet API 是 JAIN API 的一部分并被标准化为 JSR 116。SIP Servlet API 是一个服务器端接口，它描述了一个 SIP 组件或服务的容器。

Java Servlet 是一种构建 J2EE Web 应用程序主要技术。尽管 Java Servlet 仅仅能够用在应用服务器上开发基于 HTTP 协议的应用，它基本上包括用于开发服务器应用的通用 API。SIP Servlet 被定义为在 Servlet API 基础上加上了与 SIP 相关功能。Servlet API 与 SIP Servlet 的关系如下图所示：

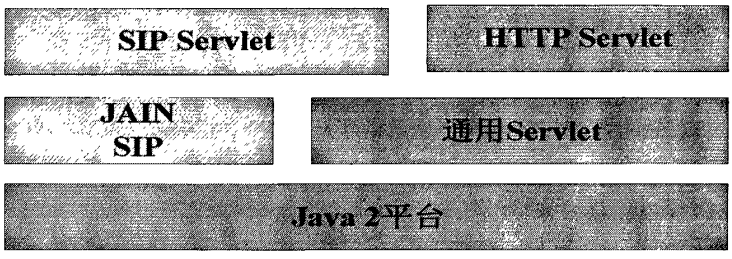

图4-3 Servlet与SIP Servlet的关系

SIP servlet 最核心的概念是包含。SIP 服务是部署或运行在在一个容器或 SIP 应用服务器上的打包 SIP servlet。容器提供了可供应用程序使用的许多服务，比如自动重试、消息调度和排队、分流和归并，以及状态管理。应用程序中只需包含高级的消息处理和业务逻辑。这使 SIP 服务的开发成为一件轻而易举的事情。

SIP Servlet 规范是 HTTP Servlet 规范的扩展。其语法、容器行为，甚至方法名都是相似的。下面我将详细分析该例子。它主要由 3 个部分组成：

- 生命周期方法

  这些方法在启动或关闭 servlet 时被容器调用。

- 消息处理方法

  SIP servlet 与 HTTP servlet 稍有不同。HTTP servlet 处理传入的请求，并发送响应消息。对于 SIP servlet，可以发送和接收请求和响应。下面将说明如何做到这一点。

当收到消息（请求或响应）时，容器将调用下面的方法。容器将按照下面图表的顺序调用这些方法，也可以重写这些方法再根据消息的类型处理消息：

表 4-1 收到 SIP 消息后调用的方法

void service(ServletRequest,ServletResponse) 如果对其进行重写，不要忘记调用 super.service()。 其默认实现调用以下方法之一：	
void doRequest(SipServletRequest) 如果对其进行重写，不要忘记调用 super.doRequest()。 其默认实现调用以下方法之一：	**void doResponse(SipServletResponse)** 如果对其进行重写，不要忘记调用 super.doResponse()。 其默认实现调用以下方法之一：：
下列请求方法之一（自解释）： doAck(SipServletRequest) doBye(SipServletRequest) doCancel(SipServletRequest) doInfo(SipServletRequest) doInvite(SipServletRequest) doMessage(SipServletRequest) doNotify(SipServletRequest) doOptions(SipServletRequest) doPrack(SipServletRequest) doRegister(SipServletRequest) doRequest(SipServletRequest) doResponse(SipServletResponse) doSubscribe(SipServletRequest)	下列响应方法之一： doProvisionalResponse(SipServletResponse)— 对应于 1xx-类响应消息。 doSuccessResponse(SipServletResponse)—对应于 2xx-类响应消息。 doRedirectResponse(SipServletResponse)— 对应于 3xx-类响应消息。 doErrorResponse(SipServletResponse)— 对应于 4xx-、5xx-以及 6xx-类响应消息

下面以 **MESSAGE** 消息为例，说明 SIP Servlet 的对处理消息的方法。

下面是处理即时消息的代码：

```
/** 当 MESSAGE 消息到达时由容器调用 */
protected void doMessage(SipServletRequest request) throws
        ServletException, IOException {
    request.createResponse(SipServletResponse.SC_OK).send();
    String message = request.getContent().toString();
    String from = request.getFrom().toString();
    {业务逻辑处理代码部分...}
}
/** 当收到错误消息后由容器调用 */
protected void doErrorResponse(SipServletResponse response)
        throws ServletException, IOException {
    super.doErrorResponse(response);
    {业务逻辑处理代码部分...}
}
/** 当接收到一个 200 OK 消息后由容器调用 */
protected void doSuccessResponse(SipServletResponse response)
        throws ServletException, IOException {
    super.doSuccessResponse(response);
    //结束创建的应用 Session
    response.getApplicationSession().invalidate();
}
```

第一个方法在收到一个 MESSAGE 消息时被调用。最初以一条 200 OK 消息响应，表明收到了消息。然后它处理服务器命令。最后，它调用一个业务逻辑方法来广播传入的消息。传入的错误响应消息表明上一个请求失败了。成功的响应消息表明上一个 MESSAGE 消息被应用系统正确地接收了。因此不再需要该会话，可以将其删除了。通常，MESSAGE 消息是以无状态的形式发送的，并不保存消息之间的连接信息。(对于 INVITE 消息来说，情况不是这样的，它打开一个有状态的会话直到发送 BYE。)。

对于 HTTP servlet，还必须编写 web.xml 部署描述符。而在 SIP servlet 中，对应的文件是 sip.xml，我们在其中列出 SIP servlet、初始化参数以及映射（哪个 SIP servlet 处理哪些 SIP 消息）。其语法类似于 web.xml，但<servlet-mapping>标签除外。它不会将一个 URL 模式映射到 servlet，而是（基于字段和子字段的内容）描述一个条件，SIP 请求必须满足这个条件才能被映射到 servlet。SIP Servlet 规范第 11 节描述了所有的字段、子字段以及用于该映射的条件。注意，该映射只用于初始请求；同一个会话/对话中的后续请求由处理初始请求的同一 Servlet 处理。下面是用于移动办公应用系统（MOAServer）的 sip.xml，它表达了 SIP Servlet 类是如何与 SIP 消息建立映射关系的。

```
<?xml version="1.0" encoding="UTF-8"?>
<sip-app>
    <servlet>
        <servlet-name>MOAServer</servlet-name>
        <servlet-class>bupt.moaserver.MOAServer</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name> MOAServer</servlet-name>
        <pattern>
            <equal>
                <var>request.method</var>
                <value>MESSAGE</value>
            </equal>
        </pattern>
    </servlet-mapping>
</sip-app>
```

在 sip.xml 配置文件中，首先建立 Servlet 名字与 Servlet 类的对应关系，然后配置 Servlet 名字和 Servlet 消息的对应关系。

现将移动办公平台中利用 SIP Servlet 技术进行应用开发的资源归纳如下：

- 开发支持包：servlet.jar 和 sipservlet.jar
- 配置文件：sip.xml
- 处理 SIP 消息的 Servlet 类

当业务逻辑代码编写完成后，下一步就是对其打包部署。打包部署可以采用命令行方式手工部署，也可以使用 ANT 工具自动部署。当部署成功后，运行 SIP 应用服务器（这里选用了 WebLogic SIP Server）再对移动办公应用系统进行测试。

4.3 客户端实现

移动办公平台客户端分为应用中间件层、组件层和业务应用层。应用中间件层实现了 J2ME 技术架构，组件层通过普通的 JavaBean 组件实现。

本节首先进行应用中间件的选择，然后给出基于该应用中间件实现移动办公业务的方法。

4.3.1 应用中间件选择

移动办公平台客户端的应用中间件应实现 J2ME 技术规范。目前主流的实现 J2ME 技术规范的应用开发工具包有 Sun 公司的 Wireless Tookit 2.5、Nokia 公司的 Prototype SDK 4.0 和 Sony Ericsson 公司的 SDK 2.2。由于最新的 Sun Wireless Tookit 实现了 sip 协议（JSR180），因此选用它作为移动办公平台客户端中应用中间件层的开发工具包。

J2ME Wireless Toolkit 2.5 是 SUN 公司提供的 J2ME 应用开发和运行工具。它的产品发行版本支持运行与面向无线产业的 Java 技术规范(JSR-185)兼容设备上的 Java 应用程序开发。本发行版中包含的完整的仿真器执行与下列产品兼容，并对下列产品提供开发支持：1.连接受限设备配置(CLDC)版本 1.1(JSR-039) 2.移动信息设备配置文件 2.0(JSR-118) 3. J2ME Web Services 版本 1.0(JSR-172) 4.无线通讯 API 版本 1.1(JSR-120) 5.移动媒体 API 版本 1.1(JSR-135)。

Sun Java Wireless Toolkit 是一组用于创建 Java 应用程序的工具，这些应用程序可在符合 Java Technology for the Wireless Industry (JTWI) (JSR 185) 规范和 Mobile Service Architecture (MSA) (JSR 248) 规范的设备上运行，它包含生成工具、实用程序和设备仿真器。

Sun Java Wireless Toolkit 可实现很多通过标准 API 呈现的优异功能。这些 API 通过 Java Community Process(JCP)进行定义，如下所示：

- Mobile Service Architecture (JSR 248)
- Java Technology for the Wireless Industry (JTWI) (JSR 185)
- Connected Limited Device Configuration (CLDC) 1.1 (JSR 139)
- Mobile Information Device Profile (MIDP) 2.0 (JSR 118)
- PDA Optional Packages for the J2ME Platform (JSR 75)
- Java APIs for Bluetooth (JSR 82)
- Mobile Media API (MMAPI) (JSR 135)
- J2ME Web Services Specification (JSR 172)
- Security and Trust Services API for J2ME (JSR 177)
- Location API for J2ME (JSR 179)
- SIP API for J2ME (JSR 180)
- Mobile 3D Graphics API for J2ME (JSR 184)
- Wireless Messaging API (WMA) 2.0 (JSR 205)
- Content Handler API (JSR 211)
- Scalable 2D Vector Graphics API for J2ME (JSR 226)
- Payment API (JSR 229)
- Advanced Multimedia Supplements (JSR 234)

- Mobile Internationalization API (JSR 238)
- Java Binding for the OpenGL(R) ES API (JSR 239)

同时也可以开发针对 CLDC 1.0 和 MIDP 1.0 的应用程序。

总之，基于移动办公平台客户端开发移动办公业务主要用到 Sun Wireless Tookit 中的 SIP API for J2ME (JSR 180)、Connected Limited Device Configuration (CLDC) 1.1 (JSR 139)和Mobile Information Device Profile (MIDP) 2.0 (JSR 118)，其它工具包根据业务需要进行使用。

4.3.2 业务实现方法

移动办公平台客户端与服务器端是C/S架构的方式，客户端采用SIP协议与服务器端进行会话控制信息的交换。

不同的移动办公业务要调用Sun WTK中的相应开发工具包，本节主要介绍使用SIP API for J2ME (JSR 180)完成SIP消息通信的原理和实现方法。

Sun WTK 2.5中SIP的工具包中包含的关键类描述如下所示：

表 4-2 SIP的工具包中包含的关键类

类	说明
Connector	创建各种连接对象的工厂。对于SIP连接，只需使用以"sip:"开头的地址，Connector就可创建SipClientConnection或SipServerConnection对象。
SipClientConnection	此类用于发送不会反复出现的SIP消息，如INVITE和MESSAGE。
SipClientConnectionListener	此接口必须由需要处理SIP响应的类来执行。
SipServerConnectionListener	此接口必须由计划接收SIP请求的类来执行。
SipServerConnection	此类可读取收到的消息。
SipRefreshHelper	该实用类管理反复发出的SIP消息（如REGISTER和SUBSCRIBE）。
SipRefreshListener	实现该接口可处理反复发出的消息的响应。

下面就移动办公平台客户端发送一个SIP请求和接收SIP响应的过程做一个简单说明：

- **发送一个SIP请求**

使用SIP可执行的最简单的操作是发送单个消息，如下时序图所示：

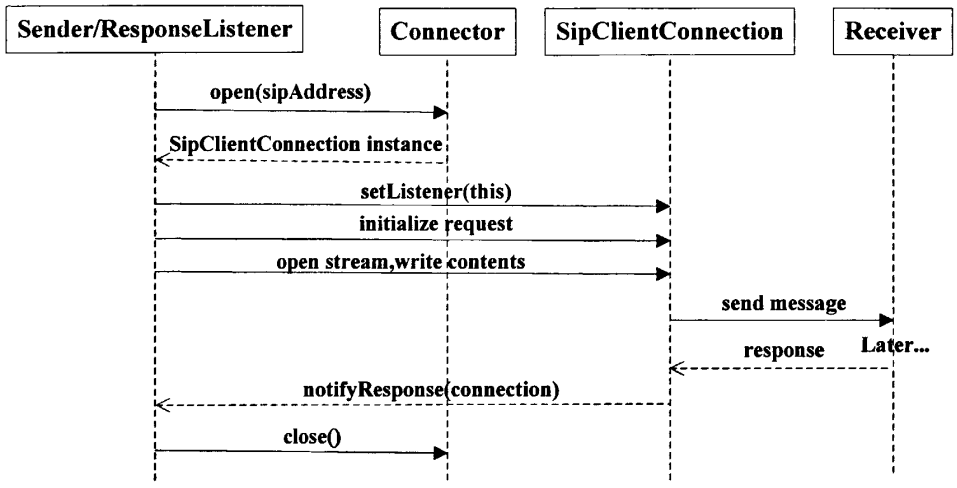

图4-4 发送一个SIP请求

如上图所示，发送消息的过程分为两部分。第一步是准备和发送消息。第二步是处理响应。我们来看一下执行此操作的代码。首先使用SipManager.sendMessage()方法执行第一步：

```
public void sendMessage(final String destination, final String message)
{
  Thread t = new Thread() {
    public void run() {
      SipClientConnection connection = null;
      OutputStream output = null;
      try {
        connection = (SipClientConnection) Connector
            .open(destination);
        connection.setListener(SipManager.this);
        connection.initRequest("MESSAGE", null);
        connection.setHeader("From", registeredAddress);
        connection.setHeader("To", destination);
        connection.setHeader("Content-Type", "text/plain");
        connection.setHeader("Content-Length", String
            .valueOf(message.length()));
        output = connection.openContentOutputStream();
        output.write(message.getBytes());
        output.close();
        output = null;
      } catch (Throwable e) {
        messageListener.notifyMessage("Error sending to "
            + destination + ": " + e.getMessage());
        e.printStackTrace();
        try {
```

```
            if (output != null) {
              output.close();
            }
            if (connection != null) {
              connection.close();
            }
          } catch (IOException e1) {
            e1.printStackTrace();
          }
        }
      }
    };
    t.start();
}
```

发送 SIP 消息时，首先打开一个客户机连接，使用它接收响应，初始化请求类型并设置大量强制的标头。请求所需的大部分 SIP 标头会自动填充默认值。然后打开输出流并写入信息，最后关闭流。此时并没有关闭连接；还需等待响应到达。

■ **接收 SIP 请求**

下图显示了应用程序处理请求的方式：

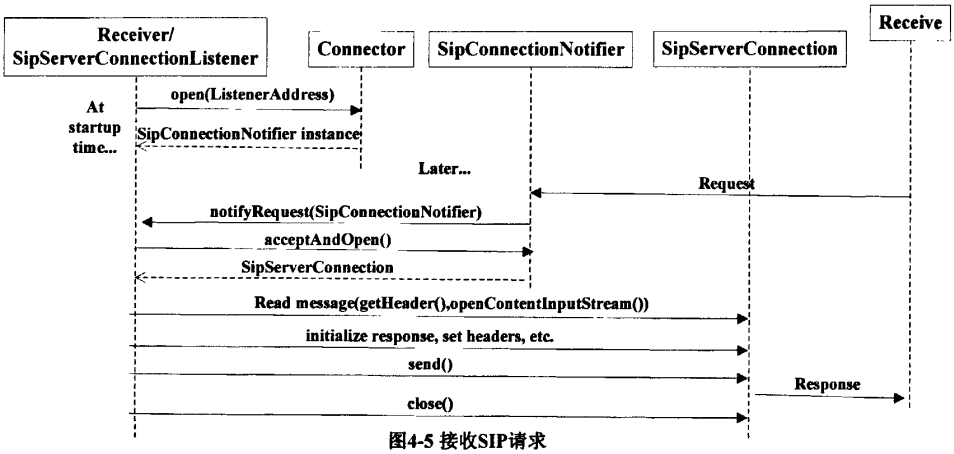

图4-5 接收SIP请求

与发送请求一样，接收请求也分为两步。第一步是在服务器连接中注册监听程序来监听到来的消息。第二步是收到请求到来的通知并发送响应。下面代码将完成第一步：

```
public void reconnect()
{
  Thread t = new Thread() {
    public void run() {
      doClose();
      try {
        sipConnection = (SipConnectionNotifier) Connector
```

```
                .open("sip:" + port);
        } catch (Throwable e) {
            e.printStackTrace();
        }
        try {
            sipConnection.setListener(SipManager.this);
            contactAddress = "sip:" + username + "@"
                + sipConnection.getLocalAddress() + ":"
                + sipConnection.getLocalPort();
        } catch (Throwable e) {
            e.printStackTrace();
        }
    };
    t.start();
}
```

此服务器连接打开之后，当有请求消息到来时，首先读取消息，然后使用SipServerConnection对象发送相应的响应。方式如下：

```
public void notifyRequest(SipConnectionNotifier notifier)
{
    SipServerConnection connection = null;
    InputStream input = null;
    try {
        connection = notifier.acceptAndOpen(); //不应当堵塞
        String size = connection.getHeader("Content-Length");
        int length = Integer.parseInt(size);
        if (length == 0) {
            connection.initResponse(200);
            connection.send();
            return; //不需做其他事情...
        }
        byte buffer[] = new byte[length];
        int readSize;
        input = connection.openContentInputStream();
        readSize = input.read(buffer);
        String from = connection.getHeader("From");
        SipAddress sipAddress = new SipAddress(from);
        from = sipAddress.getDisplayName();
        if (from != null)
            from = from.trim();
        if ((from == null) || (from.equals("")))
            from = sipAddress.getURI();
        String message = "From " + from + ": ";
        message += new String(buffer, 0, readSize);
        messageListener.notifyMessage(message);
```

```
        //所以事情都做完了，回复 OK 即可.
        connection.initResponse(200);
        connection.send();
      } catch (Throwable e) {
        e.printStackTrace();
      } finally {
        try {
          if (input != null)
            input.close();
          if (connection != null)
            connection.close();
        } catch (Throwable e) {
          e.printStackTrace();
        }
      }
}
```

本小节中说明了 SIP 消息在客户端和服务器端消息收发的原理和实现方法，对于开发移动办公平台客户端应用程序是必不可少的一个环节。

4.4 本章小结

本章着重介绍了移动办公平台架构模型中的中间件的选择和以及以及中间件的应用开发原理和方法。在移动办公平台服务器端，选择 WebLogic SIP Server 作为应用开发中间件，在客户端选择 Sun Java Wireless Tookit 作为开发工具包。在服务器端开发方面，介绍了使用 SIP Servlet 技术进行应用开发的原理和方法，在客户端，介绍了 Sun Wireless Tookit 中使用 SIP 开发工具包进行应用开发的原理和方法。

5 第五章 移动办公平台架构测试

本章搭建测试环境验证移动办公平台架构的先进性。由于本课题研究时间有限，本人只实现了一个简单的移动会议系统。验证移动办公平台架构的先进性包括：1.协同性 2.跨平台性 3.可靠性 4.可扩展性 5.可伸缩性 6.可维护性。

5.1 测试环境

测试环境可以配置、部署、运行和测试应用程序，为应用程序迁移到真正的运行环境做好充分准备。

由于实验条件所限，该测试还无法在真实的 3G 移动通信网络中进行，因此测试在一个基于 IPV4 的局域网内进行的。

5.1.1 移动会议系统实验环境

移动会议系统的实验环境中包括会议服务器和会议移动客户端，会议服务器和会议移动客户端都基于移动办公平台来实现。为了验证会议系统的协同性，增加了两个会议桌面客户端。

测试环境示意图如下所示：

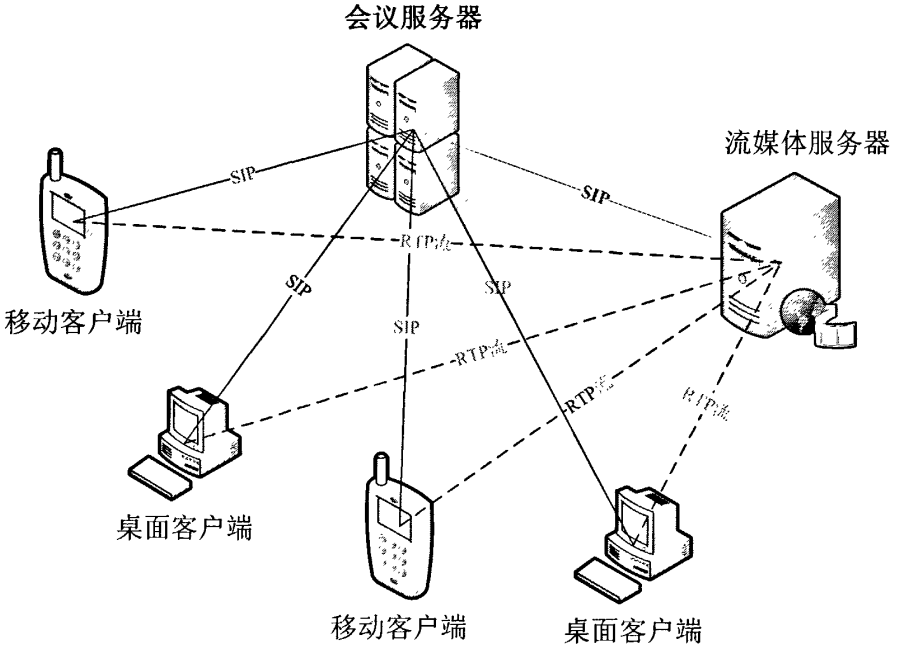

图5-1 移动会议系统实验环境示意图

会议服务器建立在 WebLogic SIP Server 之上，主要完成会议的控制和管理。会议控制采用基于消息的方式，用 SIP Servlet 技术实现。会议管理采用 JSP、Servlet、EJB 等技术。

流媒体服务器主要完成音频、视频数据的混合、压缩等功能。由于研究时间有限，这里仅实现了简单文本数据。

移动客户端基于 J2ME 技术架构来实现，这里采用 Sun 公司 WTK（Java Wireless Tookit）作为开发工具包。它带有手机模拟器，开发的会议移动客户端程序可以在模拟器上运行，也可以使用数据线拷贝到移动终端上进行测试。

桌面客户端基于 J2SE 技术架构来实现，这里采用 JAIN（Java API for Integrated Network）SIP 作为开发工具包。

5.1.2 移动会议系统设计

通过对于移动会议系统的分析，确定系统中包含的角色有系统管理员、会议管理员和参会人员。系统管理员具有管理会议管理员的功能，包括增加、修改和删除会议管理员，同时具有会议管理员的所有功能。会议管理员具有创建会议、终止会议、发布会议公告、创建、修改、删除参加会议用户的功能。参会人员具有参加会议、浏览会议公告的功能。

系统分析是从开发者的角度看待问题。系统分析描述了系统应当做什么，而不关心系统是如何做的，因此系统分析的重点仍旧集中在理解问题，而不是选择何种技术来解决问题上。

面向对象分析方法通常把对象分为 4 中类型：实体、边界、控制、生命周期，其中实体、边界和控制对象符合 MVC（Model View Controller，模型/视图/控制器）设计模式。

实体对象封装了系统的业务数据和业务逻辑。实体对象对应 MVC 设计模式中的 Model 部分，通常在系统描述中以名词形式存在，与数据库中的某个数据表对应。

边界对象用于与系统外的活动者交互。有两种类型的边界对象：用户界面和系统边界。用户界面允许系统与人交互，系统界面允许系统与其它系统交互。

控制对象为其他对象提供工作流服务。控制对象将复杂的一系列到实体对象的请求绑定成共同的工作流，从而便于边界对象访问。从边界对象到控制对象的高层消息被转换为从控制对象到实体对象的一系列消息。作为一种规则，分析模型中每个使用案例只有一个控制对象。

对象生命周期类跟踪实体对象。面向对象系统包括许多对象，这些对象必须被创建、定位或者某些时候被删除。控制或者实体对象可能采用不同的标准定位某个实体对象。作为一般规则，每一个实体类对应有一个生命周期类。

本节主要是对用户登录、增加会议管理员、创建会议几个场景进行描述，再画出相应原型设计界面。

（1）用户登录

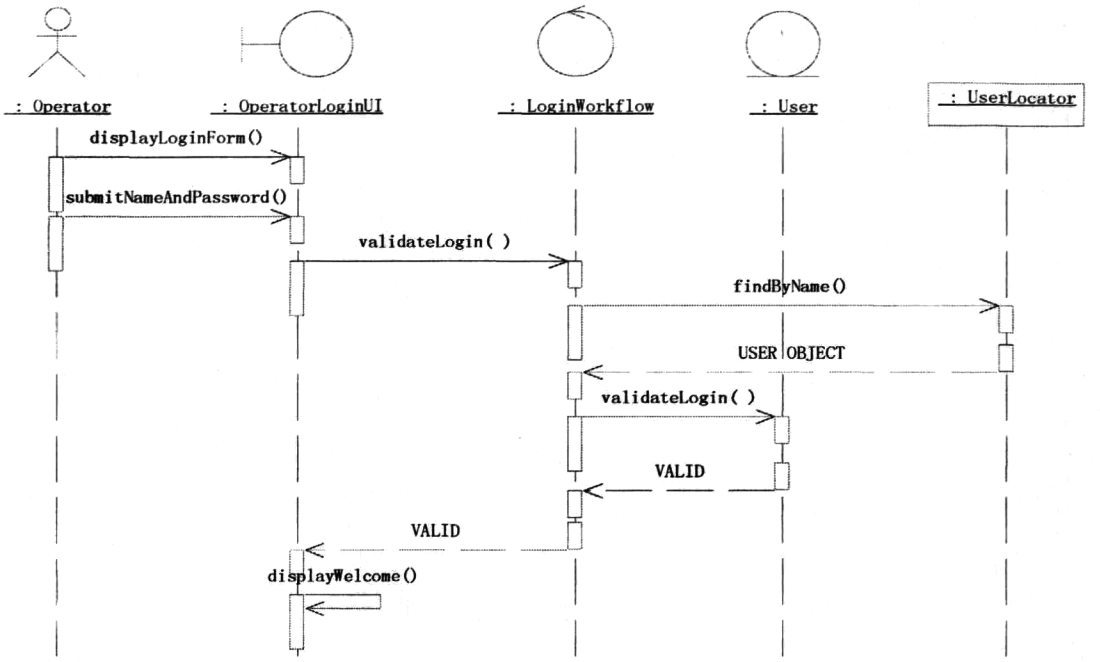

图 5-2 会议管理员登陆系统时序图

上图中，用户（这里是会议管理员 Operator）首先显示边界对象 OperatorLoginUI。当用户输入用户名和密码后提交。控制对象 LoginWorkflow 负责验证输入是否合法，它首先借助生命周期对象 UserLocator 找到该用户对象。返回后的 User 对象实际上存储于 LoginWorkflow 控制对象中，这时 LoginWorkflow 对象再去查找出的 User 对象中校验用户名密码的正确性，如果合法则显示“欢迎登录”，至此会议管理员整个登录过程结束。用户登录界面如下所示：

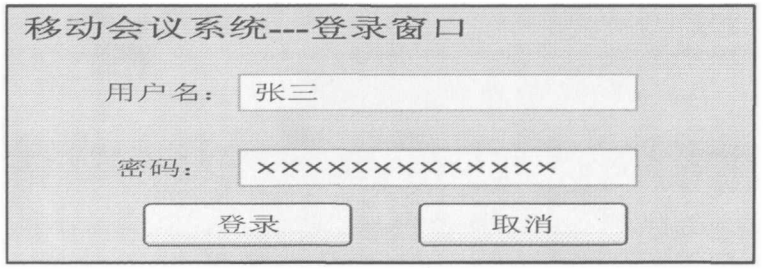

图5-3 系统登录窗口

(2) 添加界面如下所示：

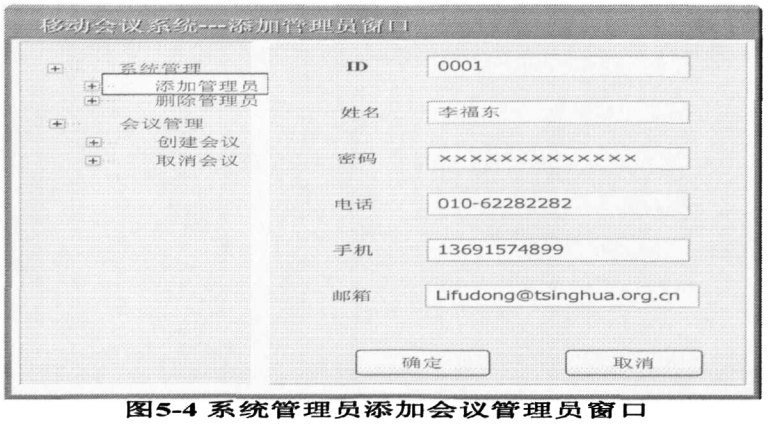

图5-4 系统管理员添加会议管理员窗口

添加会议管理员流程：

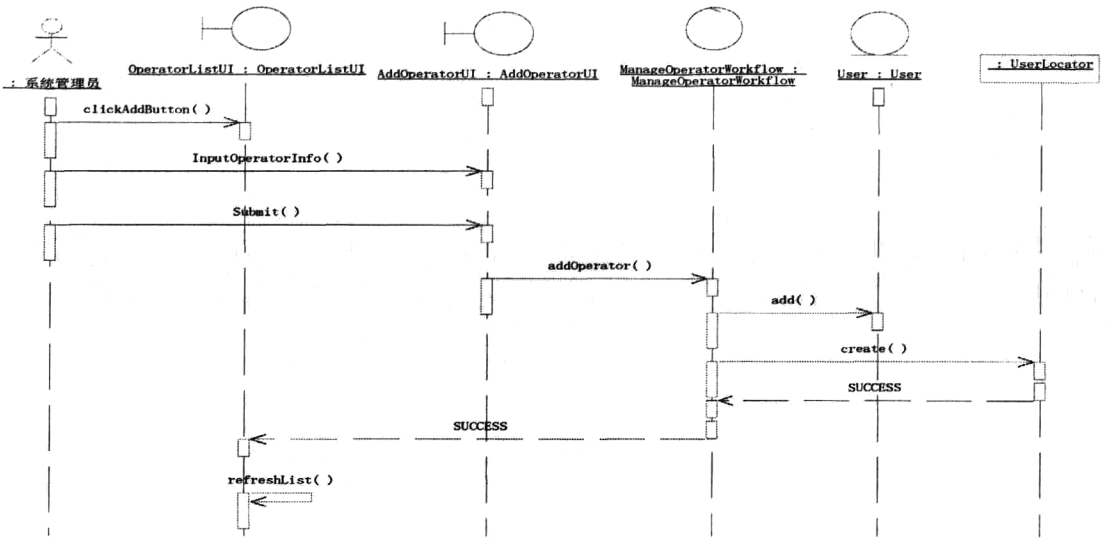

图 5-5 系统管理员添加会议管理员时序图

(3)创建会议流程：

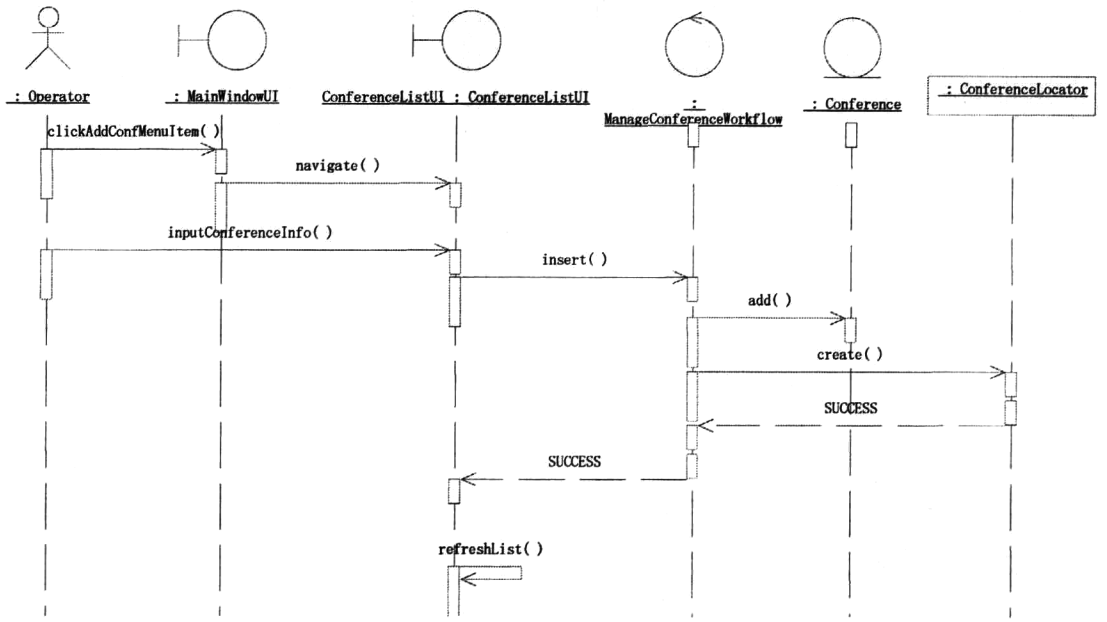

图 5-6 会议管理员创建会议时序图

创建界面如下所示：

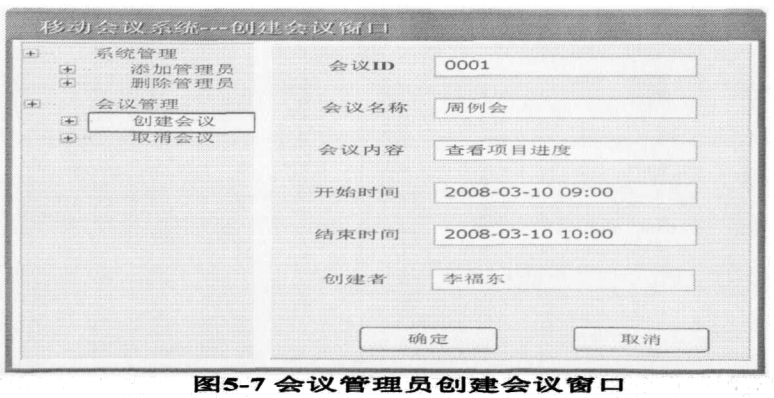

图5-7 会议管理员创建会议窗口

面向对象设计是对对象的详细描述，通过对象之间协作来满足系统的需求。设计详细地描述了解决方案，它指定实例变量、方法参数、返回类型以及技术细节。

在设计阶段应该和实现技术相结合。通过分析将系统从逻辑上分为 3 个主要模块，分别为边界模块（ConferenceUI）、控制模块（ConferenceWorkflow）和域模块（ConferenceDomain）。边界模块中存放和系统边界有关的类，控制模块存放与系统控制有关的类，域模块存放实体类。

在技术选择方面，用到 Servlet、EJB Session Bean 和 EJB Entity Bean。Servlet 的特点是以 Java 代码为中心，JSP 的特点是以用户界面为中心，在这里我们混合使用 JSP 和 Servlet，在界面面设计方面使用 JSP，当用户提交数据后交由 Servlet 处理返回 HTML 结果。

EJB 的会话 Bean 分为有状态和无状态两种。有状态会话 Bean 的优点是可以携带状态信息，缺点是占用系统资源多，无状态会话 Bean 的优点是节省系统资源，执行效率高，缺点是不能携带会话信息。由于会议控制会话 Bean 要存放于每一个办公终端特定的会话信息，因此这里选择有状态会话 Bean。

EJB 的实体 Bean 实现对象关系映射功能，它将实体对象持久化到关系型数据库中。在这里通过 EJBHome 类实现实体对象的创建和查找，相当于分析阶段的生命周期类。

移动会议系统中子系统之间及对实现技术的依赖关系如下图所示：

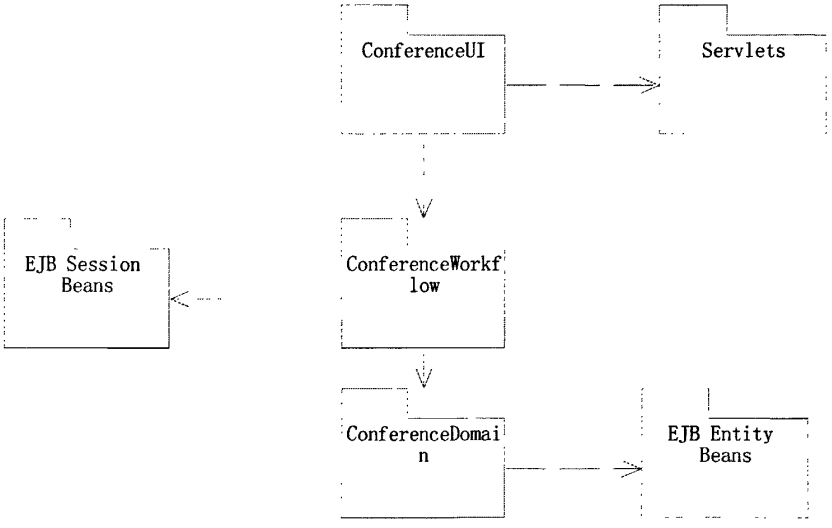

图 5-8 移动会议系统中子系统之间的依赖关系

移动会议系统服务器端的主要类如下所示：

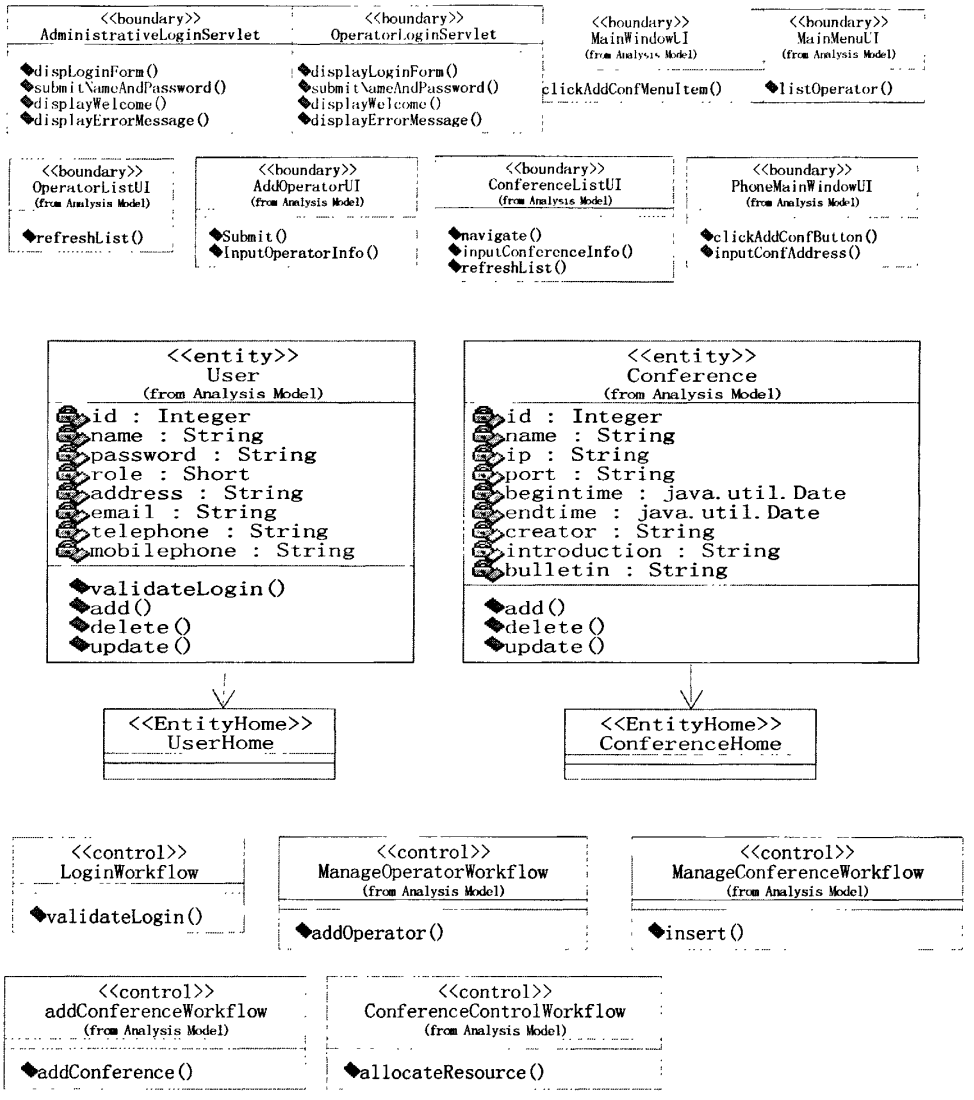

图 5-9 移动会议系统服务器端类图

移动会议系统客户端的类如下所示：

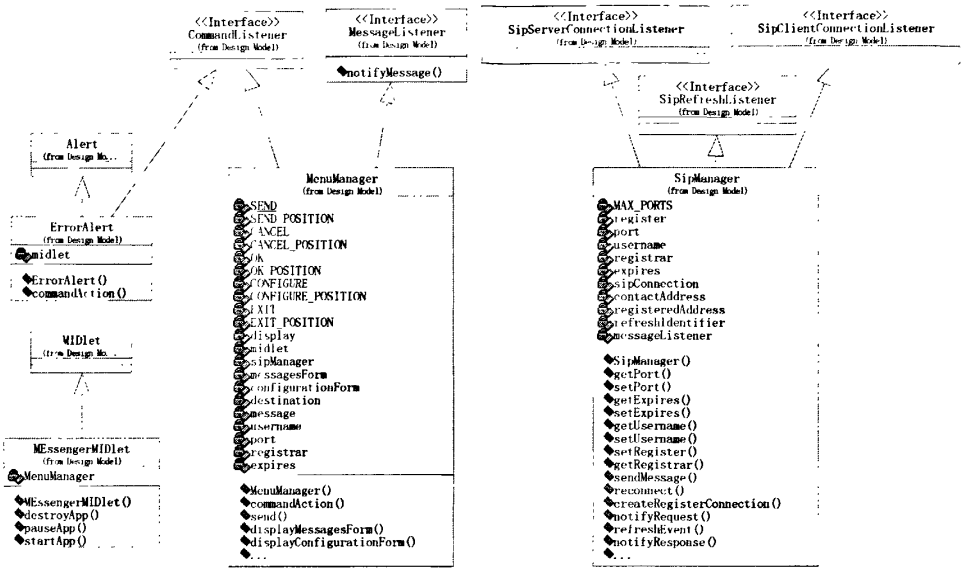

图 5-10 移动会议系统客户端类图

5.1.3 移动会议系统实现

本节中主要讲述移动会议系统的实现方法。移动会议系统包括服务器端和客户端两部分，客户端又分为移动客户端和桌面客户端两种类型。

（1） 服务器端

服务器端主要完成会议管理、会议控制功能。会议管理功能由 JSP、Servlet、EJB 技术实现，会议控制功能由 SIP Servlet 技术实现。移动会议系统的集成开发环境是 Eclipse 3.3，基于的应用中间件是 WebLogic SIP Server 并用到它带有的 weblogic.jar 和 sipservlet.jar 包进行辅助开发。

Eclipse 是 IBM 公司开发的一种集成开发环境。它采用开放的架构体系，可以与各种插件配合构建各种不同的开发和运行环境。目前，在 Eclipse 之上可以方面集成支持 UML、C++、Java 等语言的插件，可以集成如 Tomcat、WebLogic、WebSphere 等中间件插件，也可以集成包括 Oracle、MySQL、SQL Server 等数据库插件，对软件开发从系统分析、设计、实现和测试全过程提供支持。

在本课题中，使用的仅是包含 Eclipse 的缺省 Java 开发运行支持的环境，没有使用 WebLogic SIP Server 插件。Eclipse 开发环境中可以设置 JDK（Java 开发工具包）和 JRE（Java 运行时环境），可以包含开发 SIP 应用和 Web 应用的开发包，在这里分别把 JDK 和 JRE 设置为 JDK 1.6 和 JRE 1.6，把 Web 应用开发包设置为 weblogic.jar 和 sipservlet.jar。

基于 WebLogic SIP Server 开发的应用程序可以引用这些开发包并使用开发包中的相关类。当应用开发完成后，利用 ANT Build 工具对应用进行打包和部署。当将应用成功部署到 WebLogic SIP Server 上后，再对编写的应用程序进行测试。

移动会议系统在开发工具的支持下，首先将编写的源代码文件编译成目标文件，然后将目标文件 build 成 sar 文件，最后由 WebLogic SIP Server 对打包后的文件进行部署，部署成功后应用程序开始运行。

移动会议系统开发涉及的开发资源如下图所示：

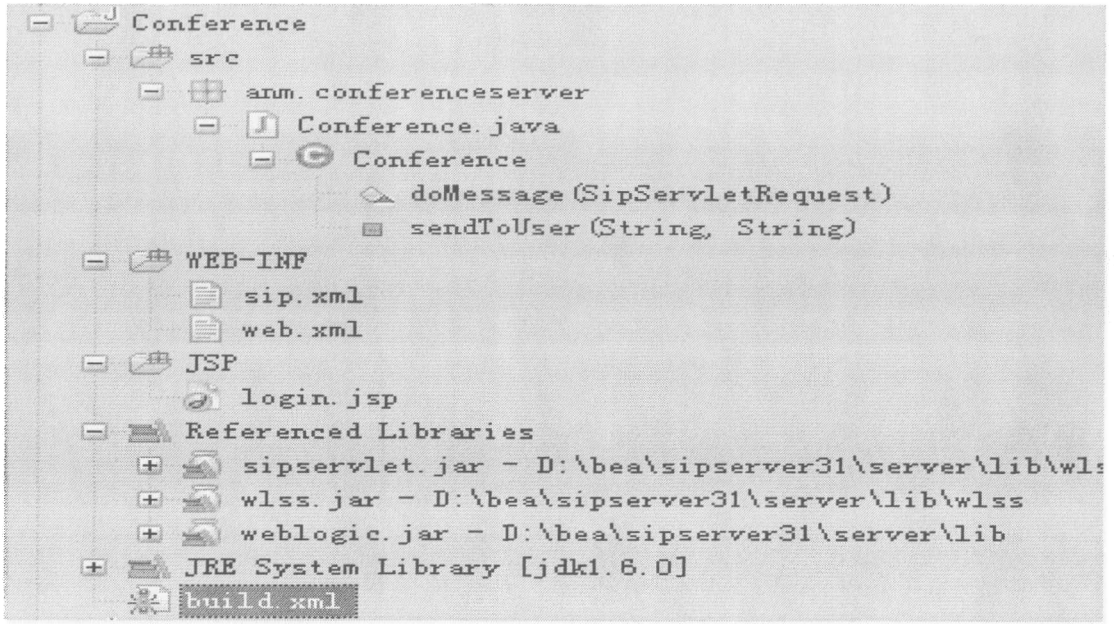

图 5-11 移动会议系统服务器端开发资源

在上面的目录中，在“Conference”根目录下，有“src”、“WEB-INF”、“JSP”3 个目录和 build.xml 文件。“src”目录下存放程序源代码，“WEB-INF”目录下两个配置文件 web.xml 和 sip.xml，“JSP”文件夹下存放所有 jsp 文件。

在移动会议系统服务器端主要难点是通过 SIP Servlet 技术对于会议进行控制，会议控制的关键代码如下所示：

```
/** 会议控制类 */
public class ConferenceServer extends SipServlet {
/** 存储用户列表的关键字. */
public static String THE_LIST="anm.conferenceserver.userList";
/** 检索会议控制服务器地址的关键字. */
public static String THE_NAME="anm.conferenceserver.name";
/** 会议控制服务器的地址. */
public String serverAddress;
/** 当启动服务时，由容器调用. */
public void init() throws ServletException {
    super.init();
    getServletContext().setAttribute(THE_LIST,new ArrayList());
    serverAddress = getServletConfig().getInitParameter(THE_NAME);
}
/** 当关闭此服务的时候，有容器调用. */
public void destroy() {
    try
    {
        sendToAll(serverAddress, "服务器正在关闭 -- 再见!");
    } catch (Throwable e)
    { //当服务器关闭时忽略所有错误.
```

```
            e.printStackTrace();
        }
        super.destroy();
    }
    /** 当MESSAGE消息到达时由容器调用. */
    protected void doMessage(SipServletRequest request) throws
            ServletException, IOException {
        request.createResponse(SipServletResponse.SC_OK).send();
        String message = request.getContent().toString();
        String from = request.getFrom().toString();
        //某个用户请求退出.
        if(message.equalsIgnoreCase("/quit")) {
            sendToUser(from, "再见！ ");
            removeUser(from);
            return;
        }
        //将用户添加到用户列表
        if(!containsUser(from)) {
            sendToUser(from, "欢迎进入多媒体会议室!!!  " + serverAddress +
                    ". 输入'/quit' 退出.");
            addUser(from);
        }
        //查询参加会议的用户
        if(message.equalsIgnoreCase("/who")) {
            String users = "用户列表:\n";
            List list =
                (List)getServletContext().getAttribute(THE_LIST);
            Iterator iter = list.iterator();
            while (iter.hasNext())
            {
                String user = (String) iter.next();
                users += user + "\n";
            }
            sendToUser(from, users);
            removeUser(from);
            return;
        }
        //如果用户悄悄地加入会议，不广播，直接返回
        if(message.equalsIgnoreCase("/join")) {
            return;
        }
        sendToAll(from, message);
    }
    /** 当由于发送了的消息包括超时导致产生错误时，由容器进行如下处理 */
```

```
protected void doErrorResponse(SipServletResponse response)
        throws ServletException, IOException {
    super.doErrorResponse(response);
    //当消息的接收方离开时，从列表中删除它
    String receiver = response.getTo().toString();
    removeUser(receiver);
}
/** 当发送消息后接收的2xx OK响应时，由容器调用 */
protected void doSuccessResponse(SipServletResponse response)
        throws ServletException, IOException {
    super.doSuccessResponse(response);
    //删除创建的应用会话.
    response.getApplicationSession().invalidate();
}
private void sendToAll(String from, String message)
        throws ServletParseException, IOException {
    SipFactory factory = (SipFactory)getServletContext().
        getAttribute("javax.servlet.sip.SipFactory");
    List list = (List)getServletContext().getAttribute(THE_LIST);
    Iterator users = list.iterator();
    while (users.hasNext()) { //Send this message to all on the list.
        String user = (String) users.next();
        SipApplicationSession session =
            factory.createApplicationSession();
        SipServletRequest request = factory.createRequest(session,
                "MESSAGE", serverAddress, user);
        String msg = from + " 已经发送的消息: \n" + message;
        request.setContent(msg.getBytes(), "text/plain");
        request.send();
    }
}
//给用户发送消息
private void sendToUser(String to, String message)
        throws ServletParseException, IOException {
    SipFactory factory = (SipFactory)getServletContext().
        getAttribute("javax.servlet.sip.SipFactory");
    SipApplicationSession session = factory.createApplicationSession();
    SipServletRequest request = factory.createRequest(session,
            "MESSAGE", serverAddress, to);
    request.setContent(message.getBytes(), "text/plain");
    request.send();
}
private boolean containsUser(String from) {
    List list = (List)getServletContext().getAttribute(THE_LIST);
```

```
        return list.contains(from);
    }
    private void addUser(String from) {
        List list = (List)getServletContext().getAttribute(THE_LIST);
        list.add(from);
    }
    private void removeUser(String from) {
        List list = (List)getServletContext().getAttribute(THE_LIST);
        list.remove(from);
    }
}
```

（2） 移动客户端

移动客户端属于移动办公平台架构的客户端部分，基于 J2ME 技术架构实现。我们选择 Eclipse 3.3 作为集成开发环境，用基于 Eclipse 的 J2ME 插件 Eclipse ME 1.7.6 进行配置管理，选择 Sun Java Wireless Tookit 2.5 作为 J2ME 开发工具包。

在移动办公平台服务器端实现部分讲了 Eclispe 3.3，下面再讲一下 Eclipse ME 1.7.6。Eclipse ME 作为 Eclipse 一个插件，致力于帮助开发者开发 J2ME 应用程序。EclipseME 并不为开发者提供无线设备模拟器，而将各手机厂商的实用模拟器紧密连接到 Eclipse 开发环境中，为开发者提供一种无缝统一的集成开发环境。开发移动会议客户端所需的开发资源如图 5-12 所示：

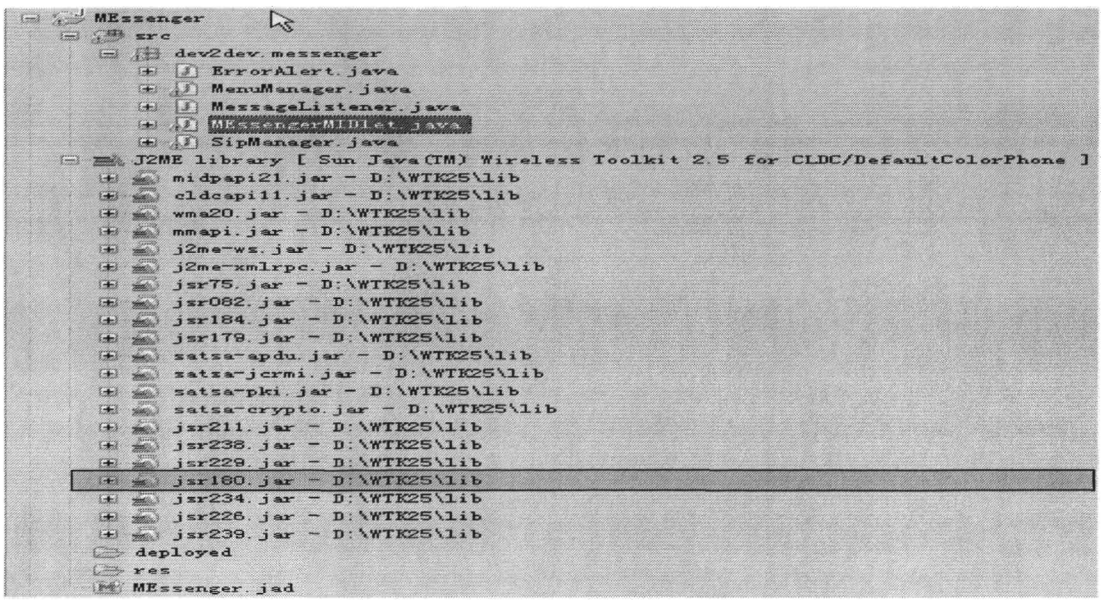

图 5-12 移动会议系统客户器端开发资源

上图中，jsr180.jar 是支持 SIP 开发的辅助工具包。移动客户端的实现关键是如何实现 SIP 消息的发送和接收，由于在移动办公平台客户端实现已经描述，在这里不再重复。

（3） 桌面客户端

为了验证基于移动办公平台的移动会议系统的协同性，通过一个多媒体会议系统的桌面客户端接入移动办公平台。

移动会议系统的桌面客户端开发环境采用 Eclipse 3.3，开发工具包采用 JDK1.6 和 JAIN SIP 1.1。Eclipse 3.3 和 JDK 1.6 已经在前面叙述过了，下面简要说一下 JAIN SIP 1.1。

JAIN APIs 是由 JCP(Java Community Process) 组织推动开发的一套基于 Java APIs 和面向对象技术的标准接口,主要用于在 Java 2 平台上快速开发下一代电信产品及业务。JAIN APIs 包含一系列 API，其中与 SIP 协议有关的 API 有 3 种：JAIN SIP，JAIN SIP Lite 和 SIP Servlets。其中 JAIN SIP 规范由 SUN 和 dynamicsoft 公司(都是 JCP 的成员)合作研发,它完全基于 IETF 的 SIP 规范(RFC2543)制定，所规定的功能包括 API 和与 API 相关的 RI(Reference Implementation)，TCK(Technology Compatibility Kit)。API 提供了 SIP 协议栈底层到 SIP 实体层之间的接口,用于在 SIP Client 和 SIP Server 之间的信息传输。开发桌面客户端用到的相关资源如图 5-13 所示：

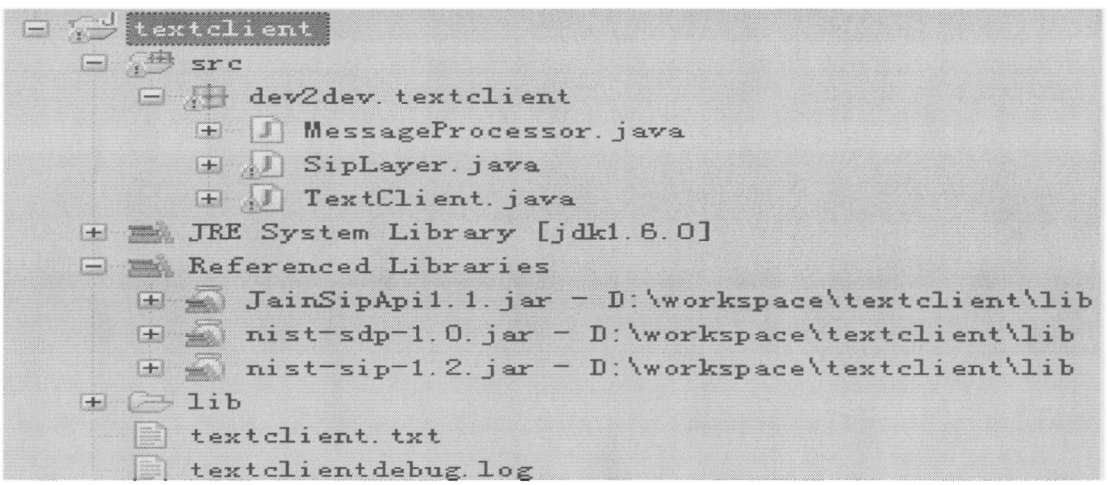

图 5-13 桌面客户器端开发资源

桌面客户端的关键实现是 SIP 消息的发送方法：

```
public void sendMessage(String to, String message) throws
        ParseException, InvalidArgumentException, SipException {

    SipURI from = addressFactory.createSipURI(getUsername(),
            getHost() + ":" + getPort());
    Address fromNameAddress = addressFactory.createAddress(from);
    fromNameAddress.setDisplayName(getUsername());
    //建立SIP头部
    FromHeader fromHeader =
        headerFactory.createFromHeader(fromNameAddress,
                "textclientv1.0");
    //提取用户名和发送地址
    String username = to.substring(to.indexOf(":")+1, to.indexOf("@"));
    String address = to.substring(to.indexOf("@")+1);
    SipURI toAddress =
        addressFactory.createSipURI(username, address);
    Address toNameAddress = addressFactory.createAddress(toAddress);
```

```
        toNameAddress.setDisplayName(username);
        ToHeader toHeader = headerFactory.createToHeader(toNameAddress, null);
        SipURI requestURI = addressFactory.createSipURI(username, address);
        requestURI.setTransportParam("udp");
        ArrayList viaHeaders = new ArrayList();
        ViaHeader viaHeader =
            headerFactory.createViaHeader(
                getHost(),
                getPort(),
                "udp",
                null);
        viaHeaders.add(viaHeader);
        CallIdHeader callIdHeader = sipProvider.getNewCallId();
        CSeqHeader cSeqHeader =
            headerFactory.createCSeqHeader(1, Request.MESSAGE);
        //设置最大跳数，防止回路
        MaxForwardsHeader maxForwards =
            headerFactory.createMaxForwardsHeader(70);
        //创建SIP请求消息头
        Request request =
            messageFactory.createRequest(
                requestURI,
                Request.MESSAGE,
                callIdHeader,
                cSeqHeader,
                fromHeader,
                toHeader,
                viaHeaders,
                maxForwards);
        SipURI contactURI = addressFactory.createSipURI(getUsername(),
                getHost());
        contactURI.setPort(getPort());
        Address contactAddress = addressFactory.createAddress(contactURI);
        contactAddress.setDisplayName(getUsername());
        ContactHeader contactHeader =
            headerFactory.createContactHeader(contactAddress);
        request.addHeader(contactHeader);
        //创建内容类型
        ContentTypeHeader contentTypeHeader =
            headerFactory.createContentTypeHeader("text", "plain");
        request.setContent(message, contentTypeHeader);
        sipProvider.sendRequest(request);
    }
```

5.2 测试实例

本测试的主要目标是测试移动办公平台的协同性。移动办公平台的可扩展性、可伸缩性、可靠性和可维护性等指标主要靠应用中间件体现，在后面会给出测试的方法和测试结果。

5.2.1 跨平台性测试

移动办公平台跨平台性测试的目的主要是验证移动办公业务对于硬件和操作系统平台的无关性，为用户可以选择适合自己的服务器和终端设备。

跨平台性测试的方法在移动办公平台服务器端分别在 RedHat Linux 9 操作系统和 Windows XP 2002 操作系统上运行移动会议系统。

当选择好特定于 RedHat Linux 9 和 Windows XP 2002 操作系统的 Java 虚拟机和相应的 WebLogic SIP Server 后，将移动会议系统部署到应用中间件上去，经过测试，移动办公平台在不同的操作系统下能够正常运行，具有跨平台性。

5.2.2 协同性测试

测试的方法是首先启动会议服务器，然后 2 个移动客户端和 2 个桌面客户端在会议服务器上进行登录注册。不同类型的客户端可以发言，会议系统中其他客户端能收到发言人的发言内容。同一时刻只允许一个人发言，发言顺序按发言的先后决定。

测试的过程分以下几步：第一、运行 WebLogic SIP Server 并部署移动会议系统。第二、移动客户端和桌面客户端首先配置 SIP Server 的 IP 地址和端口号，然后注册到移动会议系统服务器端，其中桌面客户端通过发送“/join”消息完成注册。注册界面如下所示：

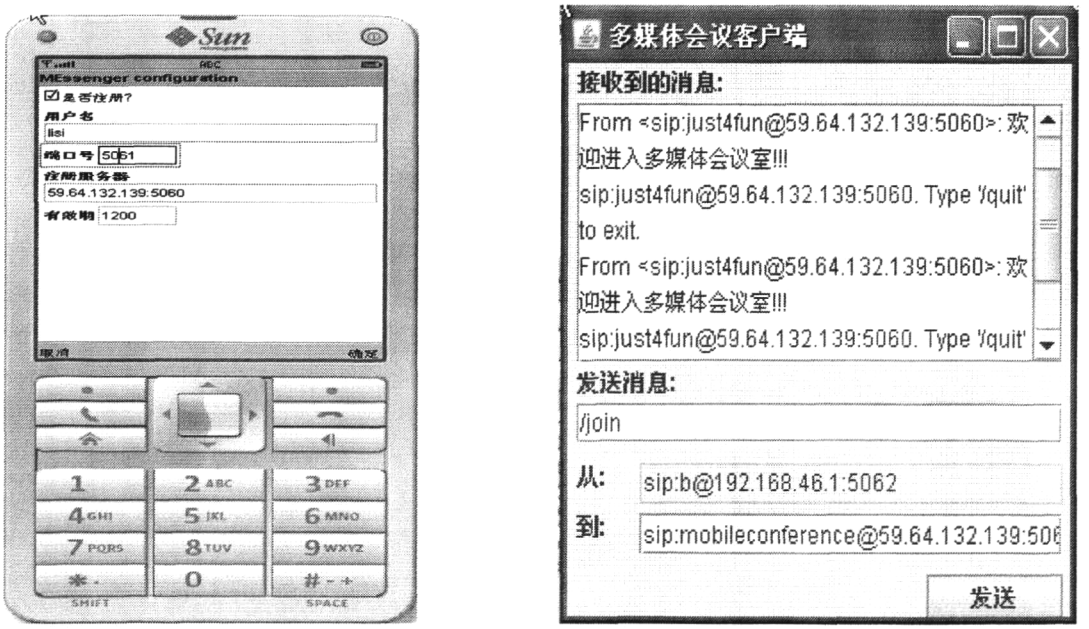

图 5-14 参会人员注册会议

第三、假设移动客户端用户是李四，它的发送目标是张三，他要输入目标地址：sip：zhangsan@59.64.132.139:5060，然后点击发送就会发送到桌面客户端张三，张三收到消息如下图所示：

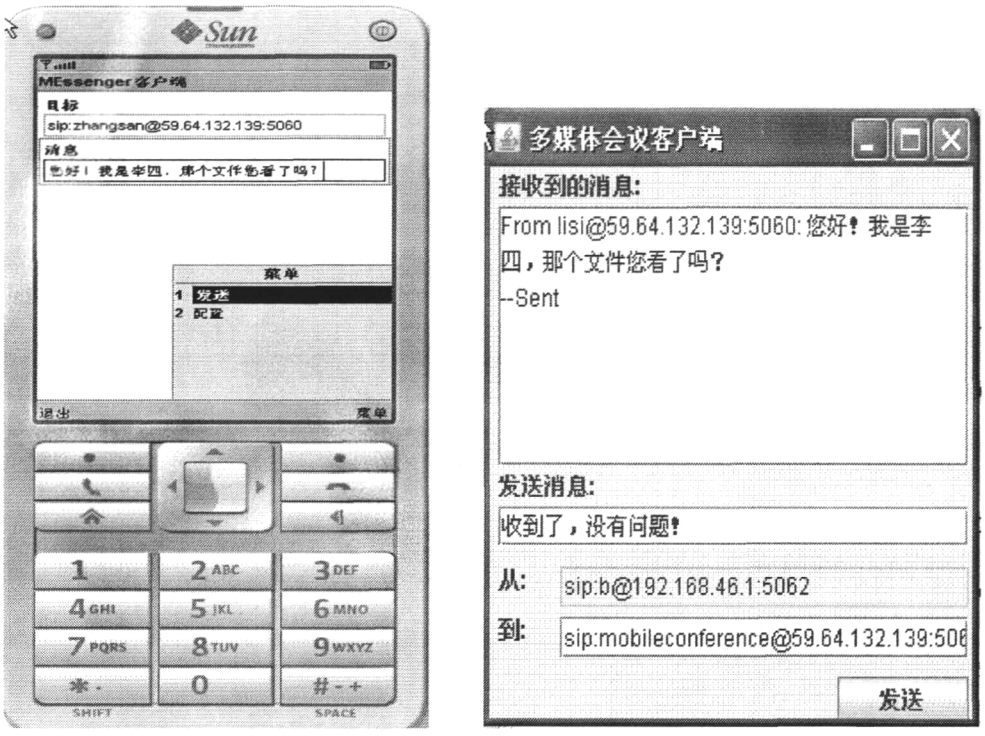

图 5-15 参会人员之间互发消息

5.2.3 可靠性测试

可靠性测试的目的是测试移动办公平台的稳定性，即当支持移动办公平台相关的硬件、应用中间件或者移动办公应用系统发生故障时，系统是否还能够继续为客户提供服务。

移动办公平台的可靠性主要是靠应用中间件来实现的。移动办公平台的具体实现中应用中间件选用 WebLogic SIP Server，它通过集群方式不仅支持 J2EE 架构中 Web 层和业务逻辑层的 Session 状态的复制，而且支持 Web 层中 SIP Servlet 状态的复制。

由于实验条件的限制，我们在一台移动办公应用服务器上启动 2 个 WebLogic SIP Server 实例，由于 WebLogic 的集群配置是在逻辑上实现的，不关心具体部署到哪一台服务器硬件上，因此其集群测试的效果是一样的。

我们的测试方法是将移动会议系统服务器端部署到包含两个 WebLogic SIP Server 实例的集群中，在正常运行是，无论 SIP 移动客户端还是桌面客户端访问集群的代理，代理再将请求转移到其中中的一台主 WebLogic SIP Server 实例中去，这时通过命令行方式把其中这个正常运行的 WebLogic SIP Server 实例停掉，当前的会话会自动转移到集群中另外一个备用 WebLogic SIP Server 实例中去。

测试的结果表明，移动办公平台架构完全符合可靠性的要求。

5.2.4 可伸缩性测试

移动办公平台架构可伸缩性测试的目的是测试能否在随着办公用户增加，系统能否通过增加软硬件资源的方式来阻止系统性能的下降。

本测试的测试方法是增加 15 个移动客户端和 15 个桌面客户端，提供访问系统的客户端个数，客户端每个 1 秒就移动会议系统发送 1 次消息，看是否移动会议系统的响应时间变慢。

在实际测试中，系统在增加客户端请求数目的情况下使得系统响应时间变长，这时通过在另外一台机器上启动一个新的 WebLogic SIP Server 实例加入到 WebLogic 集群中的方法。当配置完成后，系统的响应时间明显减少，而且可以采用热切换的方式，无需暂停现有的系统的运行。这说明移动办公平台架构满足可伸缩性的要求。

5.3 测试分析

通过以上测试表明，移动办公平台架构在协同性、可靠性、可伸缩性和跨平台性方面都能满足设计要求。

5.4 本章小结

本章首先确定使用移动会议系统测试移动办公平台架构的先进性，然后搭建了移动会议系统的测试环境，最后对移动会议系统进行了协同性、可靠性、可伸缩性和跨平台性的测试，证明了移动办公平台的先进性。

6 第六章 总结与展望

6.1 本文工作总结

本文首先分析了移动办公的发展现状以及发展要求，从而产生对于移动办公平台架构研究的必要性。然后根据移动办公平台存在的不足，结合当今先进的移动通信技术 IMS、SIP，架构设计方法 SOA，应用技术架构 J2EE、J2ME 等，给出了移动办公平台架构设计方案。

本文作者提出移动办公平台架构设计方案具有以下特点和优势：第一、支持移动办公终端与其它办公终端协同办公。第二、支持移动多媒体视频办公业务。第三、可以快速生成移动办公新业务。第四、易于与其他应用系统集成互通。第五、跨平台性、良好的可伸缩性和可扩展性。第六、高复用性。

3G 核心网子系统 IMS 可以融合多种类型的网络，从而为移动办公平台架构支持移动办公终端与其它办公终端协同办公提供了网络层支持。移动办公平台采用 SOA 的架构方法，使得移动办公平台与其他应用系统之间呈现松耦合的架构形式，有力地支持移动办公平台与其他应用系统之间的集成和互通。移动办公平台架构中设立应用中间件层，移动办公平台可以选择先进成熟的中间件作为支撑，在这里选择基于 Java 技术的两个技术平台 J2EE 和 J2ME，从而使移动办公平台具有跨平台性，应用中间件应该支持应用级集群，从而使移动办公平台具有可伸缩性和可扩展性。在移动办公平台架构中设置组件层，提供了系统的复用性。

6.2 尚存在的问题

本课题中主要完成了移动办公平台的架构设计方案并通过移动会议系统完成了对移动办公平台架构的测试，证明了它的先进性。然而，还没有完成的测试任务还有：1.移动办公平台架构中关于多媒体视频业务的支持 2.除 3G 网络、因特网之外的其他网络中的办公设备的协同性 3.支持移动办公业务实现的通用组件的设计并实现 4.采用 SOA 方式与其他业务系统集成互通等还没有进行测试验证。

另外，对于移动办公平台中关于 QoS 管理、安全管理和计费管理方面有待于进行深入研究。

6.3 对本课题研究的展望

移动办公特别适合那些经常外出的工作人员，随着社会交往的日益密切、人员之间的流动日益频繁，移动办公将在人们的生活中将会变得越来越重要。

移动办公的发展靠移动通信技术、信息技术、设计方法等的发展来推动，反过来移动办公的发展也会促进人们对于移动通信技术、信息技术、设计方法等的

研究。当前，移动办公业务的发展迫切需要有一个架构良好的支撑平台，包括先进的网络架构和应用技术架构，而对于移动办公平台架构的研究是移动办公平台研究的一个重点，直接影响到移动办公平台其他方面的研究。

通过仔细研究分析，我认为移动办公平台的架构设计研究将会集中在以下几个方面：1.移动办公平台与其它应用系统的集成 2.移动办公平台如何快速生成新业务 3.移动办公平台中的计费管理 4.移动办公平台中的安全管理 5.移动办公平台中的服务质量管理。

相信随着移动办公平台相关部分的深入研究，必将促进移动办公业务的快速发展，进而使移动办公对人们生产和生活产生更加深远的影响。

参考文献

（1）专著中的文献

[1] Gonzalo Camarillo 等 《The 3G IP 多媒体子系统 IMS-融合移动网与因特网》 人民邮电出版社 2006

[2] 程宝平 《IMS 原理与应用》 机械工业出版社 2006

[3] 毕厚杰 李秀川 《IMS 与下一代网络》 人民邮电出版社 2006

[4] 张智江等 《基于 IMS 融合、开放的下一代网络》 人民邮电出版社 2007

[5] 程宝平 梁守清 《IMS 原理与应用》 机械工业出版社 2007

[6] 田辉等 《3GPP 核心网技术》 人民邮电出版社 2007

[7] 梅玉平 《3G 的业务及管理》 人民邮电出版社 2007

[8] 周海华等 《SIP 原理与应用》 机械工业出版社 2006

[9] Eric Pulier 等 《Understanding Enterprise SOA》 Manning Publications 2006

[10] Angela Yochem 等 《J2EE 应用与 BEA WebLogic Server》电子工业出版社 2005

[11] 李绪成等 《Java EE 5 实用教程》 电子工业出版社 2007

[12] 郝玉龙等 《J2ME 移动应用开发》 清华大学出版社 2006

[13] 陈旭东等 《J2ME 应用教程》 清华大学出版社 2007

[14] 周之英 《现代软件工程·中》 科学出版社 2000

[15] 邵维忠，杨芙清 《面向对象的系统分析》 清华大学出版社 1998.12

[16] Ronald J. Norman 《Object Oriented Systems Analysis and Design》 Prentice-Hall 1998

[17] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides 《Design Patterns》 Adison Wesley 2002

[18] Mark Priestley 《Practical Object-Oriented Design with UML》 McGraw-HillCompanies, Inc. 2000.12

[19] Geri Schneider Jason P. Winters 《Applying Use Cases-A Practical Guide (Second Edition)》 2003.8

[20] CT Arlington 《Enterprise Java with UML》 Wiley Computer Publishing 2001

[21] 吴健 郑潮 汪杰 《UML 基础与 Rose 建模案例》 人民邮电出版社 2004.9

（2）期刊中的文献

[22] 张友波，张焕强，孙利民 《基于 SIP 的视频会议系统设计与实现》《计算机工程》 2005.11

[23] 王萍 廖建新等《基于 IMS 的组合类多媒体会议业务的研究和设计》《电信工程技术与标准化》 2006.12

[24] 夏俊，唐文宇 《基于下一代网络的视频会议系统》 《江苏通信技术 》 2005.10

[25] 刘德辉，宋建新 《一种基于 SIP 的多媒体会议服务器设计》 《山东通信技术》 2005.06

[26] 袁满，杨恒涛等 《移动增值服务平台架构及相关技术》 《大庆石油学院学报》 2006.10

[27] 张勇 《移动办公系统实现的方案及实践》 《湖北电力》 2007.02

[28] 苗琳，蒋念玲，李蔚 《NGN 体系下的多媒体会议系统的设计与实现》《中国新通信》 2007.10

（3）学位论文

[29] 赫宇《基于知识管理的综合办公信息管理系统的研究开发》[学位论文]，北京邮电大学，20050318

（4）技术标准

[30] J. Rosenberg, Network Working Group Request for Comments: 3261 June 2002 SIP: Session Initiation Protocol

[31] J.Rosenberg,Network Working Group Cisco Systems Request for Comments: 4353 February 2006 A Framework for Conferencing with the Session Initiation Protocol (SIP)

[32] H. Schulzrinne, Network Working Group Columbia University Request for Comments: 3550 July 2003 RTP: A Transport Protocol for Real-Time Applications

(5) 网络资源

[33] Emmanuel Proulx，SIP 简介，第 2 部分：SIP SERVLET，http://dev2dev.bea.com.cn/techdoc/20060419772.html，2006.04

[34] Emmanuel Proulx，Java ME 的 SIP API 简介，http://dev2dev.bea.com.cn/techdoc/2007/08/java-soa-sip-javame.html，2007.08

致 谢

经过一年多的理论学习与科研实践，终于完成了论文的书写工作。回首这段岁月，充满了困惑与希望，欢乐和痛苦，不禁感慨万千。

在学习与科研项目的研发过程中，学到了很多知识，也提高了学习能力，这些都离不开老师和同学们的支持与帮助。

首先要感谢我的指导老师吴伟明教授的悉心指导，吴老师对于科学技术以及业务发展趋势的认识让我受益匪浅。在她的指导下，我公开发表了学术论文两篇，完成了 2 个项目的设计与开发，可以说今天取得的成绩是和老师的关怀密不可分的。

其次，我要感谢王子正、高斐等同学对于我在科研以及论文书写方面的帮助，他们是那样的知识渊博，勤于学习并富于创新精神，与他们在从事科研项目的工作经历是愉快的，这将使我终生难忘！

最后，向在我生活、学习过程中给予我关心支持的女朋友杨阳和家人表示感谢！

攻读学位期间发表的学术论文目录

1. 吴伟明（教授），李福东，高斐，王子正（署名单位为北京邮电大学） 《面向3G发展的移动办公模式》 《办公自动化》 2007年3月 第9～11页
2. 李福东，吴伟明（教授）（署名单位为北京邮电大学） 《移动办公平台架构设计研究及关键技术》 《第十届办公自动化国际学术研讨会论文集》 2007年总第122期 第163～165页