第3章 软件体系结构风格的应用与分析

3.1 SMCSP 简介

3.1.1 项目背景

近年来，移动电子商务/电子政务有着巨大的市场前景。随着移动用户的不断增加，引发通过移动设备传输数据业务需求的成倍增加，用户对如何获得更直接、更方便的电子商务/电子政务服务与信息提出了更高的要求。此外，随着移动电子商务突飞猛进的发展，进一步引发了对移动协同商务（政务）的强烈需求，如何构建移动协同商务（政务）平台就成了一个非常具有挑战性的课题。

在过去的一段时间中，传统协同技术被引入到移动计算领域，二者相融合形成了移动协同技术，该技术引起了各大学、科研机构以及如 Microsoft、Oracle、IBM、Nokia、Motorola 等公司的极大关注。移动协同（Mobile Cooperation/Collaboration）的概念起源于 1993 年 Xerox PARC 的 Mark Weiser 提出在办公环境下利用可携带移动设备进行无线通信和交互的可能性。第 1 个应用于工业的 Mobile CSCW（Computing Support Collaboration Work）系统是 1995 年英国的 Lancaster 大学的 MOST 的多媒体协作系统。1998 年英国剑桥大学的两位学者 Luff P. 和 C. Heath 从 CSCW 系统研究者的角度，提出了 CSCW 系统移动性支持的必要性。Microsoft、Oracle、IBM、Nokia、Motorola 等公司或其研究组织主要集中于将传统 CSCW 应用到移动计算环境中的企业级办公应用；CMU、Brunel、Cambridge 等研究机构主要集中于无线网络视频和音频的协作处理、协作数据传输等方面。国内清华大学、上海交通大学等也都处于理论探讨和实验室研制阶段，但均未见实用系统。

这里所介绍的移动电子商务应用系统核心部分是移动协同服务支撑平台

(Supporting Mobile Collaboration Services Platform,SMCSP),该平台是移动通信设备与Internet有机结合所形成的,可以同时方便移动设备服务开发商与移动设备用户的一种电子商务系统,移动通信设备与SMCSP以及SMCSP与移动服务开发商提供的服务Agent之间都是通过TCP/IP协议进行通信。该系统要实现的就是移动设备用户向SMCSP注册后,可以通过网络定制自己需要的服务(已在SMCSP注册过的服务),系统除了完成商务活动外,还包括一些协同任务的处理,比如后面将介绍的应用实例(协同警务)。

移动协同是指利用移动计算技术和传统协同技术使得一组成员为了共同的目标并以群组最高利益为驱动的在一起共同工作的行为。移动协同所要考虑的主要问题包括:支持移动协同性、可动态配置、服务无关性、支持多种网络协议、系统平台无关性以及可扩展性等。

采用多移动Agent架构是解决这些问题的有效途径之一。

移动Agent概念是20世纪90年代初由General Magic公司在推出商业系统时提出的,它是继CORBA、EJB后,新一代的分布处理的关键技术。

移动Agent是一个能在异构网络中自主地从一台主机迁移到另一台主机,并可与其他Agent或资源交互的程序。它实际上是Agent技术与分布式计算技术的"混血儿"。移动Agent技术将服务请求Agent动态地移到服务器端执行,使此Agent较少依赖网络传输这一中间环节,而直接面对要访问的服务器资源,从而避免了大量数据的网络传输,降低了系统对网络带宽的依赖;移动Agent不需要统一的调度,由用户创建的Agent可以异步的在不同节点上运行,待任务完成后再将结果传送给用户;为了完成某项任务,用户可以创建多个Agent,同时在一个或若干个节点上运行,形成并行求解的能力。

多移动Agent系统(Multi Mobile Agent System,MMAS)面临着许多问题,最为关键的一环就是系统中Agent之间的协同。多Agent协同是资源的有界性和时间等约束所需要的,它能使多个Agent协调一致地有效解决问题。Agent之间的协同也是多Agent系统与其他相关研究领域,比如分布式计算、面向对象技术、专家系统相区别的关键概念之一。以自主的智能Agent为中心,使多Agent的知识、愿望、意图、规划、行动协调,以至达到协作,是MMAS的主要目标。

移动Agent技术与协同技术的结合是解决移动协同的有效方式,它结合了两者的优点,发挥了两者的长处。

移动协同服务支撑平台是指利用移动协同技术构建的支撑第三方服务的移动协同应用框架。SMCSP需要解决如下问题:

- 移动协同用户(成员)、任务和动作的知识框架表示。
- 移动资源的发现机制。
- 计算资源的调度和计算迁移机制。
- 移动网络不稳定的处理机制(协作非正常中断的处理机制)。
- 移动状态下监控远程无线或有线系统协作用户状态。
- 多个组(用户)实时协作机制。

- 大流量下移动网络的拥塞与协议优化处理。
- 移动安全机制。
- 移动协同业务共性问题(如图形标绘感知)。
- 移动协同服务支撑平台的开放性、系统平台无关性和应用无关性。
- 移动协同服务支撑平台是动态可配置的。

3.1.2 技术路线

根据 SMCSP 的功能要求和移动 Agent 的技术特点,可以确立 MMAS 分布式组织结构,如图 3-1 所示。

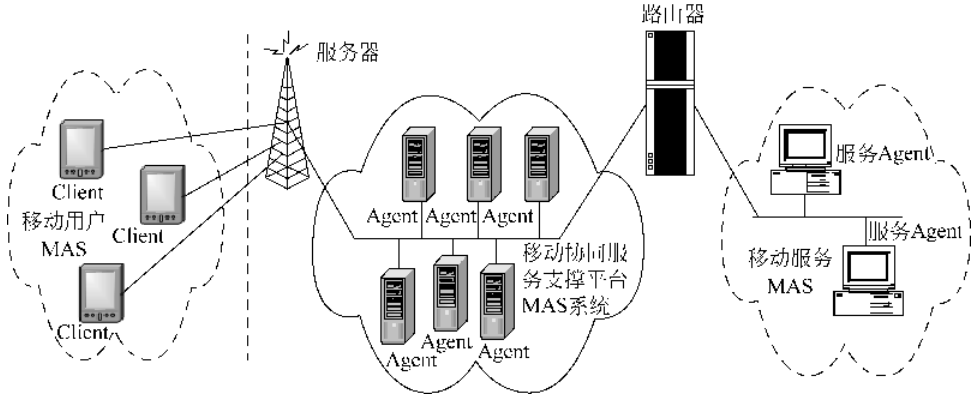

图 3-1 系统分布式的 MAS 组织结构图

系统中存在很多个中介服务机构,这些中介服务机构之间进行协作,成员 Agent 需要协同服务时,可以通过中介服务机构协调或者通过高层中介服务机构与其他中介服务机构进行协调处理,继而完成业务协同。

基于分层的逻辑设计

系统采用基于分层模式与知识库模式、面向对象模式相结合的多组态计算构架。对于不同的应用场合,采用不同的分层次序。每一层解决移动协同支撑平台中的特定问题,使问题局部化。同时利用分层式体系结构良好的复用性、可扩展性,实现整个系统的集成。系统总体上共分 5 层,如图 3-2 所示。

应用层在移动协同服务支撑平台的支撑下通过移动通信网络完成业务协同。移动协同支撑层是平台的核心,完成资源的发现、调度、协同和迁移工作;可信的基础设施为平台提供了有效的安全支撑;优化的 TCP/IP 协议为缓解大流量下的网络拥塞提供了有效的改进机制;系统应用层通过警务协同、娱乐、股票等应用来体现移动协同服务支撑平台的功能和作用。

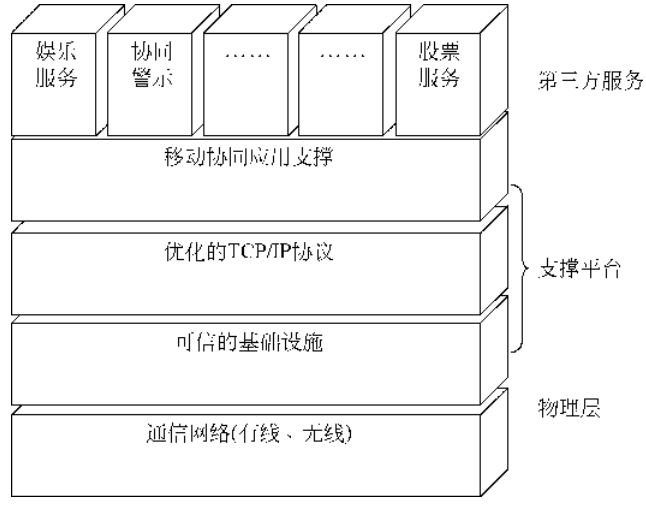

图 3-2　系统总体分层图

3.1.3　功能设计

1. 平台逻辑功能设计

平台在逻辑层面上的功能模块划分如图 3-3 所示。

整个系统包含 3 个部分,核心部分是移动协同服务支撑平台。

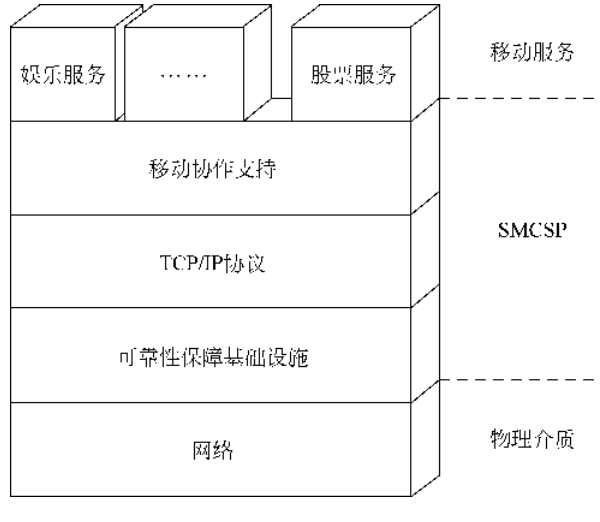

图 3-3　平台逻辑层面功能设计

移动协同服务支撑平台包括以下主要组成部分。

- AMS(Agent 管理器)：负责管理监控整个系统中所有 Agent 的运行状态，具体完成 Agent 运行态的注册、注销、监控、查询、运行状态转换(主要是 Agent 的时间空间转换)。
- DF(目录服务器)：负责管理整个系统中所有 Agent 的功能，具体完成 Agent 功能的注册、注销、查询以及与其他 Agent 系统进行通信。
- 知识管理 Agent(知识管理器)：负责管理整个系统所涉及的源知识以及领域知识的管理和应用，集体完成知识维护、知识描述、知识转换、知识交换以及一定的知识更新，以支持系统的自适应组态功能。
- 平台监测 Agent：负责管理系统中所有 Agent 运行状态维护，用以支持系统的时空转换演算以及代码移动。
- 推理支持 Agent(InferAgent)：在协同工作小组工作过程中给予相应的规则推理支持，扩充推理规则。
- 图形信息管理 Agent(GraphAgent)：负责相关服务所需图片信息的存取。
- 用户消息处理 Agent(UserMessageAgent)：负责接收用户发来的服务请求信息并进行分析，然后调用系统功能 Agent 一起给予解决并返回服务结果。
- 用户信息管理 Agent(UserInfoAgent)：处理移动设备终端用户的注册请求、认证请求，对用户的基本信息进行管理，并维护用户服务列表及在线用户列表。
- 服务消息处理 Agent(ServiceMessageAgent)：负责等待用户消息 Agent 和服务 Agent 发来的请求消息，然后调用服务 Agent 给予响应并返回服务结果。
- 服务信息管理 Agent(ServiceInfoAgent)：处理移动服务商注册及注销、修改服务信息等请求，对服务的基本信息进行管理，并维护平台注册服务列表及平台在线服务列表。

平台客户端的主要组成部分包括：

- 平台客户端程序(User Agent)：负责嵌入在移动节点上完成显示和接受客户响应的功能。
- 与平台通信服务器：负责完成整个应用系统中各个 Agent 之间的通信工作和 AMS、DF 的功能。
- 个性化服务定制模块：具体负责维护平台客户的服务选择列表。
- 身份认证模块：完善应用系统的完整性，提供移动节点的身份认证功能。
- 加密模块：完善应用系统的完整性，提供移动节点和后台服务程序之间的保密信息传递。

2. 平台应用功能设计

平台的应用功能设计如图 3-4 所示。

移动协同服务支撑平台在相关支撑库与子系统的支持下，主要有 3 大类应用功能。

1) 平台管理部分

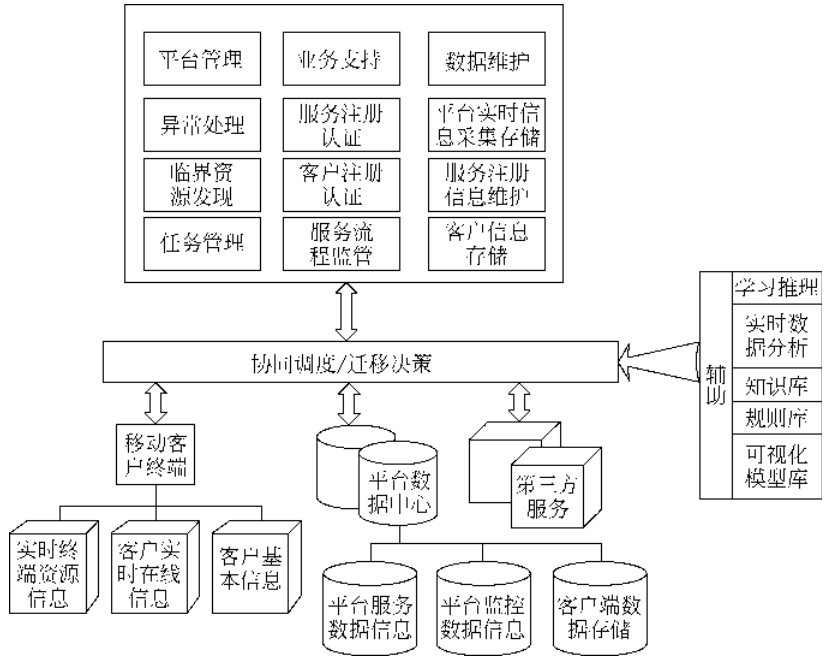

图 3-4　平台应用层面的功能设计

(1) 异常处理：平台在运行过程中，异常事件和不可预知错误的发生会影响整个平台的整个功能。在平台出现异常错误或者局部功能瘫痪的情况下，异常处理 Agent 使整个系统能够继续为用户和服务商服务，并提供异常错误详细报告，在可能的情况下还可以恢复部分功能，保证系统的健壮性。

(2) 临界资源发现：为了很好地支撑各种移动服务包括移动协同服务，平台需要能够很好地发现并支撑第三方服务。当移动用户共同关心一些服务并需要享用服务时，平台可以发现这些服务，优化这些服务的服务质量与速度。

(3) 任务处理：负责平台复杂任务的分析与调度。在知识 Agent、推理 Agent 的支持下，在移动协同各种辅助支撑库包括知识库、规则库、可视化模型库的支持下，平台对复杂的协同任务等的处理变得比较合理，这离不开协同调度与迁移策略的支持。当然，在协同调度与迁移策略的支持下，各相关功能 Agent 在各司其职的同时能够很好地协作，达到信息共享与信息协作，很好地为用户和服务商服务。

2) 业务支持部分

(1) 服务注册认证：平台提供对服务的管理。服务商提供的服务只要满足平台的一些规范和接口标准，就能成为整个 Agent 系统的一部分。服务需要在平台注册后，才能加入整个 Agent 系统，在加入成功后便可以为移动终端用户提供服务，在某些情况下，如需要升级的情况下，平台也要处理服务 Agent 的注销请求及善后处理。由于系统的开放性，系统

还能很好地支持服务 Agent 在出现异常并重新启动后的善后恢复处理。

(2) 客户注册认证：移动终端用户客户端在注册成功并且登录后，就可以享受在平台注册过的、平台支撑的、服务商开发的所有服务(用户有选择服务的权利)。移动终端用户客户端在需要的时候也可以提出注销申请，平台将给予处理。

(3) 服务流程监管：随着服务商的不同，各服务 Agent 的规模与其他特征差别较大。为了很好的支持所有服务商提供的服务，平台提供屏蔽服务 Agent 具体差异的功能，对服务流程进行监管，包括各种服务信息库、资源库的建立等。

3) 数据维护部分

(1) 平台实时信息采集存储：平台在运行过程中，平台运行监控信息流、各类协同任务调度与分析信息流、协同工作信息流、各类服务 Agent 服务信息流、各类移动终端设备用户请求信息流、各种服务 Agent 状态信息流、经验知识数据流、异常情况及处理信息流等主要信息流对平台的策略优化与管理提供必要参考，需要对其中信息进行采集、加工处理、存储，形成平台运行日志文件。

(2) 服务注册信息维护：平台支持能够很好地支撑多种服务，服务商提供的服务各种各样，除了服务平台的一些规范和接口标准，还有一些各自独立的特征。由于这些特色服务的存在，使得服务信息的维护更为繁杂。对于需要协同工作的服务 Agent 来说，要使得服务质量得到充分保障，就必须解决较为复杂服务信息的维护问题。

(3) 客户信息存储：客户信息的来源主要有两个：

① 客户基本信息，包括客户卡号、账号、年龄等，还有服务选择列表。

② 客户实时信息，包括用户在线信息、客户网络连接状态、客户地理位置信息以及需要协同工作的小组用户协同信息等。

总之，平台是为移动设备服务开发商与移动设备用户服务的，它的功能设计就是围绕着它的服务对象来生成的。这些功能的良好执行，离不开相关信息资源库及各类辅助支撑库的支持。

3.2 系统实现

3.2.1 模式选择

上面已经介绍了 SMCSP 的背景知识、不同角度的设计以及关键技术等。下面介绍具体的系统实现。SMCSP 作为一个服务平台，同时面向移动客户以及移动服务供应商。简要地说，移动供应商可以到平台注册服务，移动客户可以使用平台来完成查询与访问已注册的服务。事实上，这样的移动业务可以简单实现，而省略了中介平台。也就意味着移动供应商可以直接向移动客户提供服务，而移动客户可以直接访问供应商来获取需要的服务。实

际上，使用客户端/服务器模式足以实现该商务应用，为什么还要使用客户端/服务器/供应商的模式来取代客户端/服务器模式？下面的讨论中将说明相关原因。

1. 客户端/服务器模式

客户端/服务器模式在信息产业中占据着重要的地位。网络计算已经从基于主机的计算模式发展为客户端/服务器的计算模型。

20 世纪 80 年代之后，集中化结构逐渐被基于个人计算机的微型机网络所取代。个人计算机与工作站的使用改变了联合计算模型。正是基于此，分布式个人计算模型也浮出水面。一方面，大型机的固有缺陷，例如缺乏灵活性，使得它很难适应信息的剧烈增长并且为企业提供完整的解决方案；另一方面，微处理器正快速发展，它强大的处理能力和相对低廉的价格都有效地促进了网络的发展。用户可以根据自己的需要选择工作站、操作系统与应用程序。

客户端/服务器(C/S)软件结构的出现是为了实现不平等资源情况下的资源共享。C/S 结构定义了工作站与服务器之间的连接来实现数据与应用程序到多主机的分布。C/S 结构是软件体系结构中的经典模型，也是教学中的常见范例。从后面第 4 章的介绍中也可以看到，C/S 模型是大多数体系结构描述语言会选择用于阐述语法语义的基础模型。图 3-5 粗略地表示了 C/S 体系结构。其中的连接代表客户提交请求，之后由服务器做出响应并且返回相应的结果。

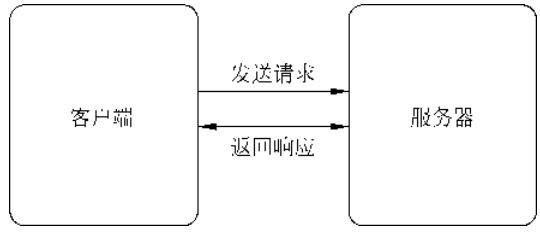

图 3-5　C/S 模型

实际中的 C/S 结构是更为复杂的，如图 3-6 所示。通常，C/S 结构的 3 个主要部分是数据库服务器、客户端应用程序与网络。

服务器管理系统的资源，其主要任务如下：

- 保证数据库的安全性。
- 控制数据库的并发访问。
- 确保数据的一致性。
- 数据备份与修复。

客户端应用程序的主要任务如下：

- 为用户提供图形界面。
- 提交用户的请求至数据库服务器，接收数据库服务的消息和响应。
- 在胖客户端模型下执行数据逻辑处理。

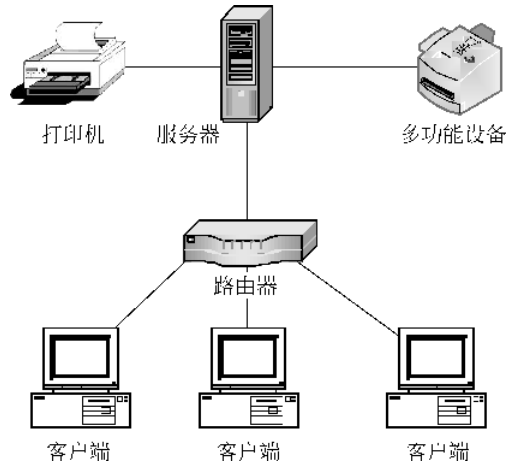

图 3-6　C/S 结构示意图

在胖客户端/瘦服务器模型中，数据传输是通过网络交互机制来实现的来减轻服务器的负担。在这里，客户端执行数据的逻辑处理与分析，该模型的数据流如图 3-7 所示。

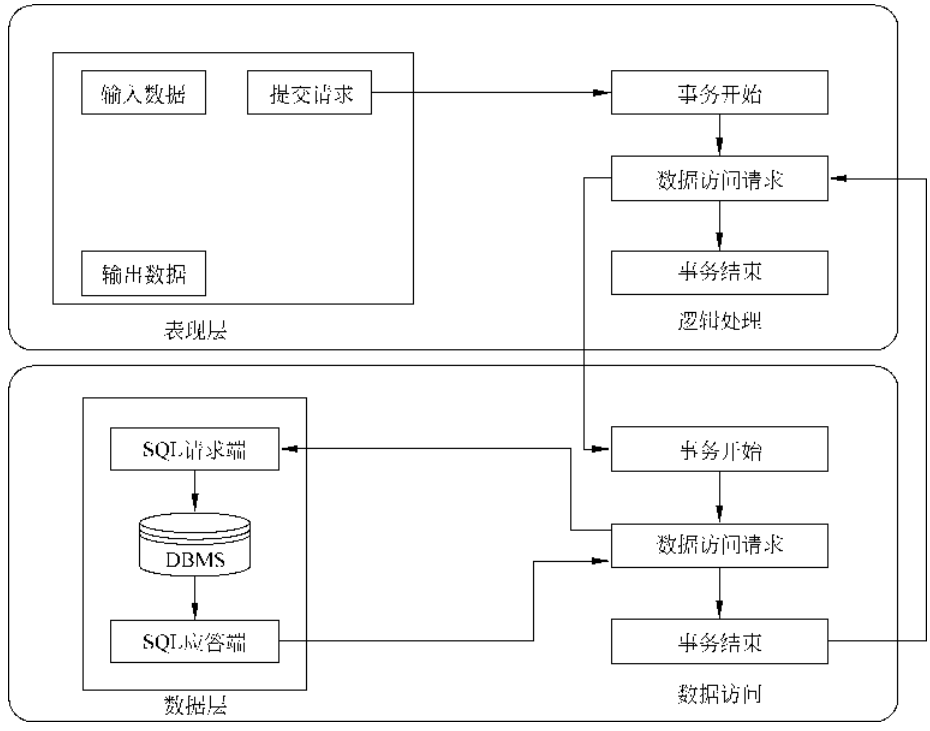

图 3-7　C/S 结构处理流程

C/S结构的优势是显而易见的。其中最重要的一点是功能构建的明确隔离带来的适应性与灵活性。的确，C/S结构拥有强大的数据操作与事务处理的能力，并且C/S模型概念十分简单而直观便于工程师理解并利用。然而，C/S结构也有如下缺点：

（1）开发费用相对较高。C/S结构对客户端硬件与软件配置都提出了更高的要求。持续的软件与硬件升级必然导致系统开销升高。

（2）客户端的程序升级比较复杂。客户端程序设计占软件开发的很大比例。

（3）软件的维护与升级比较困难。客户端必须升级来确保与服务器同步且能够正常通信。

（4）不便于使用新技术，因为不能随意更换软件开发环境。

2. 客户端/平台/供应商模式

除了上面提到的缺点之外，还有一些其他原因可以说明为什么C/S结构不适合本移动应用。

1）移动交互缺陷

与普通网络相比，无线网络更不稳定。因此，供应商与移动客户的直接交互的效率低下并且无法使人满意。客户不能享受到更好的服务质量，并且会对移动服务供应商产生不满，而这仅仅是由于无线网络本身的缺点造成的。这是服务供应商们绝对不愿意看到的局面。采用移动平台作为媒介能有效地缓解移动交互的现存问题。

2）服务管理与服务综合

在移动服务市场中，有众多的移动服务供应商。每个供应商都制定了自己的服务标准与服务接口。如果移动客户直接访问不同移动供应商的服务，那么客户端就需要下载所有服务标准与接口。

如图3-8所示，该客户已经使用了服务商A和B提供的服务，之后又需要使用C的服务，则客户端需要重新下载C指定的接口。否则C提供的服务就不能在客户端上正常运行。那么如果又需要服务商E或者F呢？

即使服务供应商的数量是有限的，但是大量的服务同样造成了混乱。于是上述机制会造成高付出并且是低效率的。在设计概念中，移动平台应向移动客户与供应商提供相关统一的标准与接口。基于该统一标准，服务供应商向平台注册他们提供的服务，然后移动客户可以访问平台完成服务浏览、服务预定、服务访问等。采用这样的方法可以有效地提高服务的综合效率，并且方便广大的移动业务使用者。

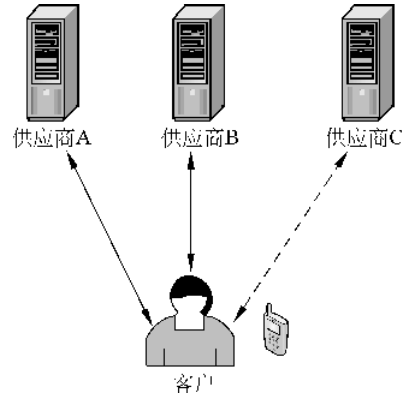

图3-8　供应商与客户

3）移动协同

除了上述两个原因之外，还有一个采用客户端/平台/供应商(CPP)结构的关键理由，即平台实现了移动

协同。移动协同是指利用移动计算技术和传统协同技术使得一组成员为了共同的目标并以群组最高利益为驱动的在一起共同工作的行为，采用 C/S 结构无法实现。前面已经介绍了移动协同的研究发展过程、它在该移动应用中的重大意义，以及数个相关议题。下面介绍移动协同是如何实现的。

粗略的 CPP 模型结构如图 3-9 所示。

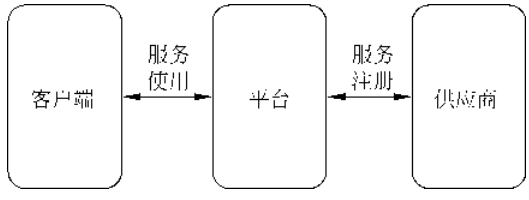

图 3-9　CPP 模型

1）客户端

移动客户端提供与表示层相同的功能，以及用户与应用程序之间的对话功能。除了提供 GUI 之外，它还负责完成客户端与服务平台之间的交互。客户端负责提交客户请求并且接收从服务平台返回的消息、根据用户需求订购平台提供的服务等。

2）平台

平台负责服务整合与服务管理，提供统一的接口与服务标准。平台负责管理服务注册、已注册服务的信息查询、服务订制、服务访问等。平台同时还维护服务供应商与其服务之间的对应关系、客户与其订制服务的对应关系等。

3）供应商

移动供应商开发各种移动服务并依照平台标准将服务发布至服务平台。简单地说，平台与移动客户之间的联系代表了服务的使用，而平台与服务供应商之间的联系代表了服务注册。当然实际的移动客户与供应商的部署关系要复杂得多，图 3-10 显示了一个简单的网络部署情况。

在介绍了该应用系统的不同角色之后，下面讨论交互机制与移动协同的实现，这些是构建移动平台的重要实现机制。

3.2.2　交互机制

SMCSP 系统为交互专门声明了自己的消息类，下面给出详细的 MessageClass 与 MessageQueue 的设计。

(1) MessageClass：消息类，这个类中定义了程序中发送和接收的消息的格式。

① 成员变量具体如表 3-1 所示。