软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(4):802−818 [doi: 10.13328/j.cnki.jos.004754]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

移动计算环境下的个人化服务发布和组装*

王明良[1,2], 陈碧欢[1,2], 彭 鑫[1,2], 黄 罡[3], 赵文耘[1,2]

[1](复旦大学 软件学院,上海 201203)

[2](上海市数据科学重点实验室(复旦大学),上海 201203)

[3](北京大学 信息科学技术学院,北京 100871)

通讯作者: 彭鑫, E-mail: pengxin@fudan.edu.cn, http://www.se.fudan.edu.cn

摘 要: 随着移动计算技术的发展,移动设备用户可以服务的方式共享移动设备及其周围设备的计算能力和计算资源.然而,移动设备的私有性、资源受限性以及移动性等特征使得这种基于移动设备提供的个人化服务在发布和组装方面存在着一些特殊问题,例如用户隐私、移动设备资源消耗、网络环境变化引起的失效问题等.针对以上这些问题,提出一种移动计算环境下的个人化服务发布和组装方法.该方法在个人化服务发布过程中引入了服务意愿的概念,并在服务选择过程中综合考虑了服务意愿和服务效用.此外,还提出了资源感知的心跳机制来定期更新服务选择所依赖的设备及服务状态信息,从而解决由于网络环境变化等引起的服务失效问题.基于该方法,提出了一个基于JADE和OSGi的实现框架.通过一个案例分析,验证了所提出的实现框架的有效性以及在移动计算环境下考虑服务意愿的合理性.

关键词: 移动计算;服务发布;服务选择;服务组装;服务提供意愿

中图法分类号: TP311

中文引用格式: 王明良,陈碧欢,彭鑫,黄罡,赵文耘.移动计算环境下的个人化服务发布和组装.软件学报,2015,26(4):802−818. http://www.jos.org.cn/1000-9825/4754.htm

英文引用格式: Wang ML, Chen BH, Peng X, Huang G, Zhao WY. Personal service publishing and composition in mobile computing environment. Ruan Jian Xue Bao/Journal of Software, 2015,26(4):802−818 (in Chinese). http://www.jos.org.cn/1000-9825/4754.htm

Personal Service Publishing and Composition in Mobile Computing Environment

WANG Ming-Liang[1,2], CHEN Bi-Huan[1,2], PENG Xin[1,2], HUANG Gang[3], ZHAO Wen-Yun[1,2]

[1](Software School, Fudan University, Shanghai 201203, China)

[2](Shanghai Key Laboratory of Data Science (Fudan University), Shanghai 201203, China)

[3](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract: With the rapid development of mobile computing techniques, mobile users can share the computing capability and resource of their mobile devices and other surrounding devices as services. However, privately-owned, resource-constrained and portable mobile devices pose challenge to the publishing and composition processes of such personal services. Specifically, such personal services may intrude the privacy of the providers, consume the constrained computing resource, and suffer failures caused by the changing network environment. To address these problems, this paper proposes a new approach for personal service publishing and mixed service composition in mobile computing environment. This approach introduces the concept of service willingness during personal service publishing, and takes into account both service willingness and service utility during service selection. Further, it presents a resource-aware heartbeat mechanism to periodically update the status of devices and services for service selection, which can also help settling service failures caused by the changing network environment. Based on JADE and OSGi, it offers an implementation framework

* 基金项目: 国家自然科学基金(61361120097); 国家高技术研究发展计划(863)(2013AA01A605)
收稿时间: 2014-07-02; 修改时间: 2014-10-14; 定稿时间: 2014-11-14

for the proposed approach. Finally, a case study is provided to show the effectiveness of the proposed implementation framework and the rationales for considering service willingness in mobile computing environment.

Key words: mobile computing; service publishing; service selection; service composition; service provision willingness

随着移动计算相关技术的快速增长,移动设备自身的计算能力和计算资源以及通过无线通信技术可以接入的周围设备的计算能力和计算资源变得越来越丰富.移动设备可以服务的方式发布、共享这些计算能力和计算资源.例如,用户可以通过移动设备将本地应用、自带及环境传感器的能力以服务的形式进行发布和共享.因此,移动设备不再只是无处不在地消费服务,更是可以无处不在地提供服务[1,2].而且,用户可以将其他用户提供的个人化服务与自身设备能力(如本地应用或传感器)、传统的 Web 服务等不同形式的服务混合组装在一起.

然而,移动设备的私有性、资源受限性以及移动性等特征使得移动设备提供的这些个人化服务对发布和组装过程有着特殊要求.首先,移动设备的私有性特征意味着所提供的个人化服务往往涉及个人隐私,这对服务的访问控制提出了额外的要求.其次,虽然移动设备的计算资源得到了快速增长,但这不能改变移动设备资源受限的事实,缺乏对设备资源考虑的服务选择可能在某些情况下影响服务提供者正常使用移动设备.最后,移动设备的移动性特征意味着可能的网络环境变化,从而间接导致发布的服务不能被正确寻址,造成服务失败.

针对以上问题,本文提出了一种移动计算环境下的个人化服务发布和服务组装方法.首先,该方法在个人化服务发布过程中引入了服务意愿的概念,并在服务选择过程中综合考虑了服务意愿和服务效用.其中,服务意愿是个人化服务提供者接收到服务请求后,综合考虑设备资源状况、服务对资源的消耗情况以及隐私策略等因素之后对服务提供的主观愿意程度;而服务效用是各个服务质量的综合度量值.由于服务效用和服务意愿并不能总是同时达到最高值,这种服务选择方法试图在服务请求者的效用和服务提供者的意愿之间达到一种平衡.其次,我们提出了一种资源感知的心跳机制定期地更新服务选择所依赖的设备及服务状态信息,从而确保用于服务选择的状态信息的实时性.同时,这种心跳机制可以解决移动环境下网络动态变化所引起的服务失效问题,而且能够在一定程度上减少电量、流量等资源的消耗.最后,基于本文所提出的方法,我们设计了一个基于 JADE** 和 OSGi[3]技术的实现框架来支持个人化服务发布以及组装.通过一个案例分析,我们验证了本文所提出的实现框架的有效性以及在移动计算环境下考虑服务意愿的合理性.

本文第 1 节详细介绍研究动机以及问题分析.第 2 节介绍本文所提出的移动计算环境下的服务发布和选择方法.第 3 节介绍一个支持个人化服务发布以及组装的实现框架.第 4 节进行一个案例研究,并对本文的相关问题进行讨论.第 5 节对比分析相关工作.第 6 节对全文进行总结并对未来工作进行展望.

1 问题分析

移动设备的计算、存储以及通信等能力的快速增长使得移动设备从一个单一的通信工具转变为一个可以支撑复杂应用、具有整合能力的功能性平台[4−6].同时,借助于越来越丰富的内置传感器***,移动设备具备了传统个人电脑所不具备的感知能力,比如移动设备能够对动态变化的上下文环境进行感知[7,8],从而实现信息世界与物理世界的融合[9].进一步地,通过红外、蓝牙、Wi-Fi 以及近场通信(NFC)等无线通信技术,移动设备可以与周围环境中的传感器进行通信,从而获得更加丰富且精准的上下文环境信息[10],甚至可以操控或者利用周围环境中的设备和资源(例如投影仪等)[6]乃至其他移动设备所提供的设备和资源(例如智能手机上的温度传感器)(如图 1 所示).

目前,借助于社交网络的发展,移动用户可以发布和使用移动设备上的内容资源(例如图片、文字等).相应地,移动用户也可以在移动设备的计算能力和计算资源富足的情况下,以服务的方式发布、共享移动设备的计算能力和计算资源(包括内置的传感器以及所安装的软件应用,可以称为本地服务).同时,移动用户也可以通过

** JAVA Agent DEvelopment Framework, http://jade.tilab.com/

*** Google wants to make you android phone much smarter with accelerometer and other sensors, http://www.unwiredview.com/2009/05/21/google-wants-to-make-your-android-phone-muchsmarter-with-accelerometer-and-other-sensors/

各种通信技术使用并在适当条件下共享周围环境中的设备的计算能力和计算资源(可以称为外围服务).因此,移动设备不再只是无处不在地消费服务,更是无处不在地提供服务[1,2](如图 1 所示),我们把这种基于个人移动设备发布共享的服务称为个人化服务.

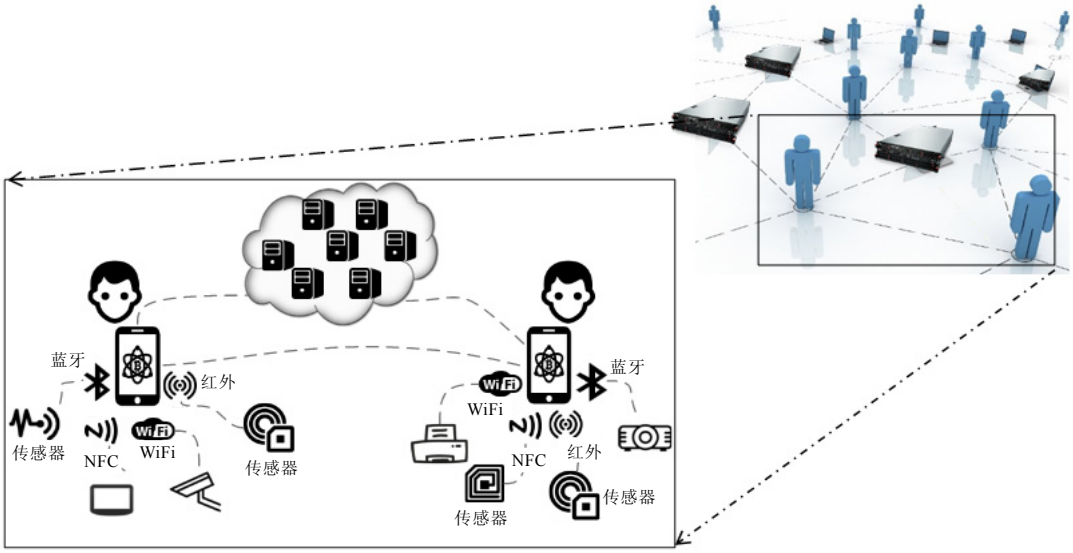


Fig.1 Personal service in mobile computing environment

图1 移动环境的个人化服务

然而,受制于移动设备的资源受限性和移动性特征,移动环境下个人化服务的发布和服务的组装还存在着一些问题.

首先,移动设备的资源受限性使得传统的基于效用的服务选择方法不适用于移动环境下的服务选择.传统的基于效用的服务选择方法是针对部署在计算资源相对充足的服务器上的服务的.对于个人化服务的服务提供可以认为是服务提供者在计算资源存在结余时才有的主动行为.如果仅仅考虑服务效用,那么候选服务中效用最高的那个服务(服务 A)总是被选中,对该服务(服务 A)的频繁调用将占用所在设备大量的 CPU 时间、网络流量以及电量,从而严重影响用户正常使用其设备,这就违背了服务提供者的初衷.兼顾服务意愿和服务效用的服务选择算法(见第 2.3 节)能够同时保证服务效用和服务意愿在较高水平,其中意愿(见第 2.1 节)中的资源意愿(见第 2.1.3 节)部分正是考虑了服务提供设备的资源状况,从而确保服务消费者能够在不影响服务提供者正常使用移动设备的情况的同时获取效用较高的服务.

其次,移动设备的移动性使得移动服务的请求者可能在运行时无法正确寻址到服务提供者[11].移动设备的“移动”同时可能意味着个人化服务提供设备所处网络环境的动态变化,当从一个网络环境切换到另一个网络环境时,其访问地址将随之变化,从而导致其在上一网络环境发布的服务将无法被其他用户正确寻址到.我们所提出的扩展的心跳机制(见第 2.2.1 节)能够确保服务访问不受网络环境变化的影响,但这并不是这种心跳机制的主要任务.相比于传统的心跳机制,资源感知的心跳机制(见第 2.2.2 节)甚至还可以减少不必要的心跳次数,从而在一定程度上减少开销,更适合移动环境.

最后,虽然个人化服务的出现为我们提供了新的服务可能和新的服务可选项,但是如何对这些不同类型的服务进行整合利用,是摆在我们面前的另一个难题.对本地服务的调用方式依赖于具体的本地服务封装技术.对外围服务的调用方式与具体的服务发现协议(比如 Jini,UPnP[12])和通信协议有关.很明显,这些服务的调用方式彼此不同.因此,对这些不同类型的服务进行整合利用变得非常困难.本文提出的基于 JADE 和 OSGi 技术(见第 3.1 节)的服务组装技术(见第 4 节)利用服务驱动(见第 3.3 节)封装具体的访问技术细节,使之对服务调用者透明化,从而方便实现对不同类型服务的整合利用.

2 服务发布和选择方法

本文提出了一种兼顾服务意愿和服务效用的服务选择方法.其中,意愿是服务提供者对服务提供的主观愿意程度,效用是各个服务质量的综合度量值,而本文主要关注于服务意愿.在本节中,我们将首先介绍服务意愿的影响因子以及意愿函数(见第 2.1 节),然后介绍一个资源感知的心跳机制来确保用于服务选择的状况信息的实时性(见第 2.2 节),最后介绍兼顾服务意愿和服务效用的服务选择方法(见第 2.3 节).

2.1 服务提供意愿

2.1.1 意 愿

移动设备通常是私有的,对于设备拥有者而言,其正常使用(娱乐、通信、社交等)才是最重要的.而在移动设备上发布服务是设备拥有者在计算资源出现结余时才有的主动行为.在后文中,提供服务的移动设备称为服务提供设备,服务提供设备的拥有者称为服务提供者.服务提供者接收到服务请求,并综合考虑了设备资源状况、服务对资源的消耗情况以及服务请求者的特征等因素之后,服务提供者并不总是完全愿意提供服务给请求者.以此为出发点,与使用效用来表示服务请求者的主观满意程度类似,本文提出使用服务提供意愿来表征服务提供者的主观愿意程度.

定义 1(意愿/Willingness). 服务提供意愿是服务提供者接受单次服务请求的主观愿意程度,简称为服务意愿.

2.1.2 意愿因子

影响意愿的可量化因素称为意愿因子.我们主要考虑表 1 所示的 5 种意愿因子.进一步分析可知,这 5 种因素可分为 3 类,即设备资源状况、服务特征以及服务请求者特征.可用一句话对这 3 类因素进行概括:在不同资源状况下,为不同请求者提供不同服务的意愿不同.但除了这 3 类客观因素外,意愿还受到来自服务提供者的一些主观因素的影响,比如服务提供者的当前状态等.

Table 1 Willingness factors

表 1 意愿因子

名称	描述	动/静态	类型	方面
Per_s	服务将使用到哪些隐私信息,需要哪些权限等	静态	服务特征	隐私相关
Req	服务请求者是谁以及请求者和“我”的关系信息	动态	请求者特征	隐私相关
R_{left}	移动设备剩余多少资源等,设备当前 CPU 负载情况	动态	资源状况	资源相关
D_R	(针对消耗类资源,比如电量、网络流量)距离下次充电、流量刷新等还有多久	动态	资源状况	资源相关
$R_{s,cost}$	服务会消耗多少资源	静态	服务特征	资源相关

服务特征信息不会因外界因素而变化,所以是在服务发布时就确定的静态信息;每次服务请求的请求者都可能不同,所以请求者是特定于单次服务请求的动态信息;设备资源状况会随着时间和资源的消耗发生变化(D_R 的变化和 R_{left} 变化),所以每次服务请求时的资源状况也会有所不同.受到这些动态意愿因子的影响,服务提供者的意愿也是动态变化的.

进一步分析,服务特征包括服务对资源的消耗量($R_{s,cost}$)和服务的隐私特征(Per_s)两方面特征;根据请求者特征(Req)可以实现对个人隐私的动态控制;设备资源状况则主要是指设备当前的资源剩余状况(D_R,R_{left}).所以这 3 类意愿因子又可归结为两方面,即隐私相关(服务隐私特征 Per_s 和请求者特征 Req)和资源相关(资源消耗量 $R_{s,cost}$ 和资源剩余状况 D_R,R_{left}).

2.1.3 意愿函数

意愿因子 $R_{s,cost}$,R_{left},D_R,Req,Per_s 对服务提供者意愿都有着或大或小的影响.本文提出使用意愿函数来表示意愿和意愿因子之间的关系.函数 f_W 由提供者指定.意愿函数通常如下表示:

$$W = \omega \times f_W\left(R_{s,cost}, Per_s, D_R, R_{left}, Req\right),$$

其中,ω表示服务提供者的主观权重,ω通常只在与个人能力或身份相关的服务中才会起作用,比如法语专家提供的法语翻译服务,修理工提供的空调修理服务,它可以用来反映服务提供者的当前状态,比如当法语专家正在

午休,ω就可能小于 1.

意愿值在区间[0,1)上,0 表示拒绝服务,意愿值越大表示服务提供者的意愿越强.但意愿的强弱似乎没有意义,因为只要意愿值大于 0 就表示服务提供者愿意提供服务,但是选择意愿值较小的服务将极有可能影响服务提供者的正常使用,而且意愿值越小,这种可能性越大.

意愿受到隐私相关和资源相关两方面意愿因子的影响,本文提出分别使用资源意愿函数$f_{W,R}$和隐私意愿函数$f_{W,P}$来分别表示资源方面因子和隐私方面因子与意愿之间的关系,相应地,其计算结果分别称为资源意愿 W_R 和隐私意愿 W_P.

$$W_R = f_{W,P}\left(R_{s,cost}, R_{left}, D_R\right),\ W_P = f_{W,P}\left(Per_s, Req\right).$$

资源意愿函数和隐私意愿函数通过相乘结合.

$$W = \omega \times W_R \times W_P = \omega \times f_{W,R} \times f_{W,P}.$$

基于隐私安全考虑而进行拒绝接受服务请求不难理解.而兼顾资源因素则是因为移动设备是资源受限的,移动服务提供者可能因为资源不足拒绝访问.对于资源充足的设备,服务提供者不会因为资源不足而拒绝提供服务,此时我们认为资源意愿为 1,由以上函数可知,此时,这种兼顾意愿的访问控制退化为单纯的基于隐私的访问控制.

$$W = 1 \times f_{W,P} = f_{W,P}.$$

下文对资源意愿和隐私意愿分别加以讨论.

1.资源意愿

服务提供意愿受不同资源的影响也会不同.如在网络流量充足、但电量不足时,我们会特别在意电量消耗情况而不太关注流量消耗,即电量资源状况对于意愿的影响较大,流量资源状况对意愿的影响较小.基于此,我们引入了单项资源意愿的概念,资源 r 的单项资源意愿——如果只受到资源 r 的影响,服务提供者的意愿.资源 r 足够充足时,服务提供者不会因为资源 r 不足而拒绝提供服务,此时我们可以认为资源 r 的单项资源意愿为 1,即 $W_r = f_{W,r} = 1$,当资源 r 足够贫乏以至于服务提供者不愿意任何服务消耗 r 时,资源 r 的单项资源意愿为 0,即 $W_r = f_{W,r} = 0$.

服务提供者的资源意愿(为了更好地区分,称为总体资源意愿)由不同资源的单项资源意愿共同决定.在此意愿模型中,总体资源意愿是由不同资源的单项资源意愿相乘计算得到的.以服务 s 的总体资源意愿为例:

$$W_{s,R} = f_{W,R}(D_R, R_{left}, R_{s,cost}) = \prod_{r \in R} f_{W,r}(r_{left}, d_r, r_{s,cost}),$$

其中,$r_{left} \in R_{left}, r_{s,cost} \in R_{s,cost}, d_r \in D_R, f_{W,R} \in [0,1], f_{W,r} \in [0,1]$.

$R_{s,cost}$ 是服务 s 消耗的资源及该资源消耗量的集合.$r_{s,cost}$ 是服务 s 对资源 r 的消耗量,$r_{s,cost}$ 由服务开发者测试统计得到,它是服务特征信息的一部分.r_{left},d_r 是设备资源状况信息的一部分,需要在接收到服务请求时获取.

$f_{W,r}(r_{left}, d_r, r_{s,cost})$是单项资源意愿的计算函数,称为单项资源意愿函数.我们更关心(意愿低 $f_{W,r}\downarrow$)那些量少($r_{left}\downarrow$)、消耗快($r_{s,cost}\uparrow$)且长时间不能补充($d_r\uparrow$)的资源的消耗状况,而对那些量大($r_{left}\uparrow$)、消耗慢($r_{s,cost}\downarrow$)且能快速得到补充($d_r\downarrow$)的资源不太在意(意愿高$f_{W,r}\downarrow$).比如对于同样的 50M 网络流量,通常情况下,在月初时(d_r较大)其单项资源意愿将低于月底(d_r较小);同样是 50M 流量,对于一次只消耗 1K 的服务($r_{s,cost}$较小)和一次调用要消耗 2M 流量($r_{s,cost}$ 较大)的两种服务,服务提供者将更愿意提供前者服务.由此可知,资源意愿和资源剩余量 r_{left} 正相关,与 d_r,$r_{s,cost}$ 负相关.

资源 r 足够充足时,W_r=1,代入意愿函数后,总体意愿的计算结果不会有变化.这意味着资源意愿函数适用于传统 SOA 环境——在传统 SOA 环境中服务由固定服务器提供,与移动设备不同,这些服务器的几乎所有资源都是“足够”充足的,此时它们的单项资源意愿都为 1,计算得到的总体资源意愿也为 1,资源意愿为 1 意味着服务提供者不太在意资源的消耗情况,这与事实相符.

资源 r 贫乏时,W_r=0,服务提供者不允许任何外部服务消耗资源 r.当接收到对服务 s 的请求(服务 s 消耗资源 r)时,代入意愿函数可知计算得到的最终意愿值为 0,即服务提供者拒绝提供该服务——因为只要调用该服务

就需要消耗资源 r,而服务提供者不允许任何外部服务消耗资源 r.

由于资源意愿可由单项资源意愿相乘计算得到,服务提供者只需为每一种关注的资源指定单项资源意愿函数.单项资源意愿函数是“全局”的,它适用于该设备发布的所有服务.资源意愿函数具有可扩展性,当开始关注新的资源类型时,只需为这种资源设置意愿函数.

2.隐私意愿

社交网络中隐私控制策略函数通常都是离散的,即用户为每一项隐私资源(比如相册)针对社交圈的每一个人(比如张某)或每一个群体(比如同学)设置访问权限(可以访问/不可以访问等).隐私意愿依赖于服务请求者信息 Req 和服务的隐私特征 Per_s,隐私特征由服务实现者描述,这包括该服务使用到的隐私信息(比如个人通信录信息)和权限(比如读写本地文件权限),$f_{W,P}$ 是服务提供者指定的访问控制函数,称为隐私意愿函数.

隐私意愿的计算过程可描述为服务提供者根据服务隐私特征和服务请求者特征(服务请求者的身份、服务请求者和服务提供者的关系等)确定意愿值.

$$W_P = f_{W,P}(Per_s, Req), f_{W,P} \in [0,1].$$

在社交网络中,$f_{W,P}$ 的值域通常为{0,1},即只有允许和不允许.在本意愿模型中,$f_{W,P}$ 的值域为[0,1],这是因为,我们有时会说:“我不是很愿意为某某提供该服务”,此时意愿值应介于 0 和 1 之间.服务提供者为每一项隐私信息和权限设置意愿函数,可称为单项隐私意愿函数(类同于单项资源意愿函数),对单个服务的隐私意愿是该服务所涉及到的所有隐私信息和权限的意愿的叠加.如下所示:

$$W_P = f_{W,P}(Per_s, Req) = \prod_{p \in Per_s} f_{W,p}(Req).$$

Per_s 是服务 s 使用的所有隐私信息和权限,$f_{W,p}(Req)$是隐私信息或权限 p 的意愿函数.采用这种方案的优点是,只需要针对关心的资源和权限进行设置,而与具体的服务无关.比如,假设小李向小张发出的服务 s 请求需要读取通信录和写文件,且小张允许小李使用个人通信录的意愿是 0.5,允许小李读写文件的意愿是 0.4,那么小张允许小李调用此服务的隐私意愿是 0.2.

2.2 资源状况更新

2.2.1 扩展的心跳机制

心跳机制广泛应用于移动互联网应用.无论是 Android 原生应用,还是 QQ、微博和微信等互联网社交应用,都采用了心跳机制.互联网应用的心跳包除了宣告终端在线外,还有一项重要的任务,就是提供移动终端的即时地址——处于“移动”状态的移动设备,其网络环境也可能是变化的,例如移动服务提供设备从 Wi-Fi 网络 A 切换到另一 Wi-Fi 网络 B,此时其访问地址也将随之变化,从而导致其在上一网络环境发布的服务将无法被其他用户正确寻址到,造成绑定失败.

我们借鉴了心跳机制,并在原有心跳机制的基础上进行了扩展,赋予了“心跳”新的任务——提供设备的实时状态信息.意愿函数依赖的动态意愿因子 R_{left},D_R,Req 需要在服务请求时获取,其中 Req(主要为请求者身份信息以及请求者与提供者之间的关系信息)可由服务中介直接获取——服务中介拥有社交网络中所有参与者以及参与者之间关系的所有信息,D_R 可根据当前时间与用户设定时间点(比如流量在每个月的第 1 天刷新)比较得知,但设备资源状况信息 R_{left} 必须由服务提供终端实时提供.这些信息可以通过心跳机制“周期性地”发送,即心跳包的内容将不只是包含即时地址,还包含资源相关状况信息.

2.2.2 资源感知的心跳机制

心跳频率过低会影响资源状况信息的实时程度,过时的状态信息会影响意愿计算的准确程度,还可能使得服务请求者在较长时间内无法正确寻址到服务提供设备.心跳频率越高,发送资源状况信息的间隔也越短,服务选择过程中服务中介进行计算所依赖的这些信息越接近于设备的实时状态,故而计算得到的意愿也会更接近真实情况.这样看来似乎心跳频率越高越好,但是心跳会带来副作用,每一次的心跳都需要消耗一定的电量和网络流量,所以心跳频率越高,副作用也越大——那么应该在什么时候发送心跳包以及以多大的频率发送心跳包?

通常来说,具体心跳周期是由系统实际需求决定的.在已有的软件系统中,心跳周期往往是固定的,比如旧

版 QQ 的心跳周期为 30s,新版 QQ 为 180s,微信为 300s,Google 原生应用为 1 680s 左右.考虑到移动设备的资源受限特性,我们提出了一种由资源状况决定周期的心跳机制——资源感知的心跳机制.

资源感知的心跳机制在心跳周期的确定机制上有别于传统的心跳机制——其心跳周期不是固定的,而是受到资源状况的影响,资源状况好的时候,心跳周期短,资源状况变坏时,心跳周期变长.资源感知的心跳的合理性在于:(1) 设备资源贫乏时,服务意愿也会变低.相比于其他设备提供的效用相同.但是意愿更高的服务,意愿较低的服务被选中的可能性也会变低.所以该设备被寻址的概率也会低,此时降低地址刷新频率(访问地址刷新是心跳机制的任务之一)对服务绑定成功率的影响不会太大.(2) 设备资源贫乏时,用户使用手机(进行娱乐等非必要功能使用)的频率会降低,服务被调用的概率也降低,所以资源消耗的速度也会降低,资源状况的变化也会变慢.资源状况在较长时间内不会有太大变化,此时高频率地更新资源状况信息(更新资源状况信息是心跳机制的任务之一)没有必要.(3) 设备资源贫乏时,即便有限的心跳开销也会变得紧要,降低心跳频率就非常有必要.

综上可知,资源贫乏时降低心跳频率有条件也有必要.此时心跳会开始放缓,心跳周期变长.而资源充足时,用户意愿已经可以得到保证,这时我们更需要关心的是状态信息的准确程度;又因为资源充足,心跳所带来的开销对设备已经不再那么重要,所以心跳模块会加快心跳包的发送频率以确保服务中介能够获得及时、准确的状态信息,并在网络变化后迅速更新访问地址.

资源感知的心跳机制的具体周期由心跳函数决定,心跳函数以资源状况为输入,输出距离下一次发送心跳包的时间间隔(即下一次心跳包在多长时间后发送).资源状况是动态变化的,所以心跳周期也会动态变化.

2.2.3 心跳驱动的服务管理

资源感知的心跳机制不仅可以告知当前移动设备的“存活”状态,还可以根据心跳频率间接地判断移动设备的资源状况,从而驱动对“资源敏感(resource-sensitive)”服务的自动管理——资源感知的服务管理(resource-driven service management).资源敏感的服务是指那些对资源要求较高、资源消耗量较大的服务.

当移动设备资源状况较差时,依然维护所有发布的服务是不明智的.一方面,该设备提供的所有服务的意愿都会降低,尤其是资源敏感服务,这些服务越来越不太可能被选中.另一方面,维护这些服务本身就需要消耗资源,维护所有的服务会加剧资源紧缺状况.

心跳周期是资源状态的间接指示灯,心跳驱动的服务管理会根据心跳决定是否继续托管某些服务.每一个服务都有一个根据服务自身特征设定的心跳频率阈值,资源越敏感的服务其心跳频率阈值越高,当心跳频率降到阈值以下时,该服务会被取消发布,并进入休眠状态;当心跳恢复到阈值以上时,该服务被重新激活.

2.3 服务选择

局部选择(local selection)和全局最优(global optimization)是传统 Web 服务选择过程中两种常用的服务选择方法.局部选择方法针对给定抽象服务从候选服务集中选择效用最优的单个原子服务;全局最优方法主要用于解决基于 QoS 的服务组合问题——在满足用户给定的全局 QoS 约束前提下,对给定的多个抽象服务,从候选服务集合中选择总体效用最高的服务组合.关于局部选择方法和全局选择方法已经有了大量的相关研究[13−16].

为了更好地与已有的众多服务选择方法兼容,我们不提出全新的服务选择方法,而只是对已有的服务选择方法进行继承式扩展.扩展后的服务选择方法的最大变化是增加了预处理和意愿叠加步骤(如图 2 所示).兼顾服务意愿和服务效用的服务选择方法能在很大程度上与已有局部选择方法正交.

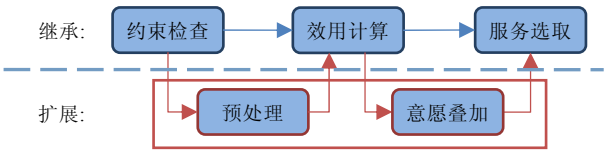


Fig.2 Service selection method with both service willingness and utility taken into account

图 2 兼顾服务意愿和服务效用的服务选择方法

2.3.1 预处理

预处理过程的主要任务是意愿计算和基于意愿的预筛选.

服务中介会为服务候选集中的每一个服务计算意愿值,服务中介已经拥有了进行意愿计算所需要的所有信息(这包括意愿函数,服务特征,服务请求者信息以及设备资源状况).将意愿因子归一化后代入意愿函数计算得到的意愿值 W 会随服务进入下一阶段.

意愿为 0 的服务最后必定不会被选取,这些服务可以直接被移出候选集;当服务候选集合较大时,可适当移除一些意愿大于 0 的服务.从候选集中剔除这些意愿较小的服务的过程便是服务预筛选.预筛选的目的是缩小候选集,避免不必要的操作和计算.

2.3.2 意愿叠加和服务选择

重新设计的服务选择方法能够与局部选择方法兼容.在局部选择方法中,抽象服务的每个候选服务都可以根据服务请求者给定的效用函数计算得到一个效用值 U.基于效用的服务选择方法根据效用值排序并选择值最大的服务,但在兼顾服务意愿和服务效用的服务选择方法中,每一个候选服务在预处理阶段计算得到了一个意愿值 W.意愿叠加阶段(如图 2 所示)将二者进行融合,这一融合可通过相乘进行,可视为在效用 U 的基础上叠加了一个反映服务提供者意愿的权重 W,用 wu 表示最后的计算结果.最后,根据 wu 排序并选取 wu 最大的服务.

$$wu_s = U_s \times W_s.$$

兼顾服务意愿和服务效用的服务选择方法同时也适用于传统 SOA 环境.传统 SOA 环境中的服务提供者通常都是资源充足的固定服务器,由于资源总是充足的,根据资源意愿函数可知,这些设备的服务意愿总是 1.在预处理阶段,由于所有意愿都为 1,所以不会有任何服务被剔除;在意愿叠加阶段,根据意愿叠加算法可知,效用值在叠加意愿前后不会有任何变化.最后的选取阶段和基于效用的服务选择方法效果完全相同.

由表 2 所给示例可知,效用最高的 $S1$,由于意愿太低落选.兼顾了服务提供者意愿和服务消费者效用的服务选择方法了确保意愿和效用值都保持在较高水平.兼顾了意愿和效用的服务选择过程不仅是服务中介代理客户对服务提供者进行选择的过程,也是服务中介代理服务提供者对服务请求进行选择的过程.

Table 2 Example of the new service selection method

表 2 兼顾服务意愿和服务效用的服务选择方法示例

服务	意愿	效用	wu	
$S1$	0.2	0.9	0.18	☒
$S2$	0.7	0.7	0.49	☑
$S3$	0.8	0.4	0.32	☒

3 实现框架

本节实现了一个基于 Android 的移动服务共享框架,为本文所提出的服务提供和选择框架提供了底层的基础设施.

3.1 技术基础

JADE-LEAP[17,18] 和 OSGi[3] 是实现该服务共享框架的核心技术.JADE-LEAP 是 JADE(Java agent development framework)的针对移动平台的一个扩展;OSGi 是一个基于 Java 语言的业务规范,它存在着多种不同的实现,本文使用的是 Apache Felix[****].二者都支持面向服务的体系结构,但又有各自的侧重点.

具体而言,OSGi 支持动态地装载和卸载组件(bundle),因此通过将服务/协议驱动(见第 3.3 节)包装成 Bundle,可以实现服务/协议驱动的动态装载和卸载.此外,OSGi 提供的服务注册和发现功能,使得 OSGi 容器内部的本地服务注册和消费非常便利.然而,OSGi 缺乏分布式支持,而 JADE 所提供的平台间协作的能力可以赋予 OSGi 一定的分布式能力.由于服务共享的本质也是多个主体(服务提供者和服务请求者)之间的协作,而 JADE 正是为简化多主体系统的开发过程而设计的.此外,JADE 提供的目录服务器(directory facilitator,简称 DF)可直

[****] Apache.org. Apache Felix http://felix.apache.org/

接用作服务中介,它可为平台内的主体提供服务注册和服务查找支持.而 JADE 主体结合 FSMBehaviour 还可实现轻量级的流程引擎[19].因此本文选择了 JADE 实现服务共享社交圈.然而,JADE 缺乏对本地服务的管理支持.

可以看到,将 JADE 与 OSGi 结合可以方便地实现服务的发布、查询、管理和使用(如图 3 所示).其中,OSGi 用于实现服务容器,服务和服务桩(stub)“生存”在此容器中.而 JADE 搭建起不同主体之间沟通协作的桥梁,不同设备的服务容器通过 JADE 提供的主体通信通道(agent communication channel,简称 ACC)进行通信.我们基于 ACC 实现了一种简单的远程服务请求机制,从而赋予 OSGi 一定的分布式能力——支持以服务桩的形式注册并使用远程服务和外围服务(这里的远程服务指的是通过服务中介获取的社交圈好友提供的服务).

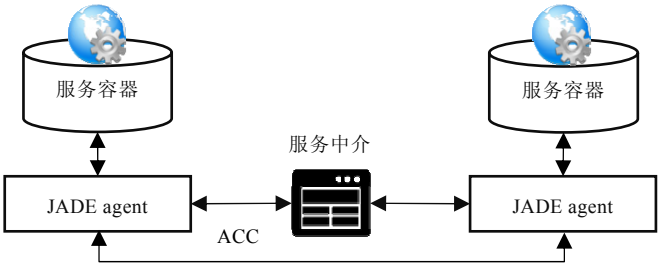


Fig.3 Cooperation relation of JADE and OSGi in the implementation framework

图 3 JADE 和 OSGi 在实现框架中的协作关系

3.2 框架概览

该移动服务共享框架对外表现为一个服务容器,该服务容器对外暴露两个接口——用户管理接口和流程引擎(如图 4 所示).通过用户管理接口,用户可进行发布服务、取消发布服务、对发布的服务进行相关参数和意愿函数设置等操作.上层应用程序制定的业务流程描述为 FSMBehaviour 对象可交由流程引擎执行.

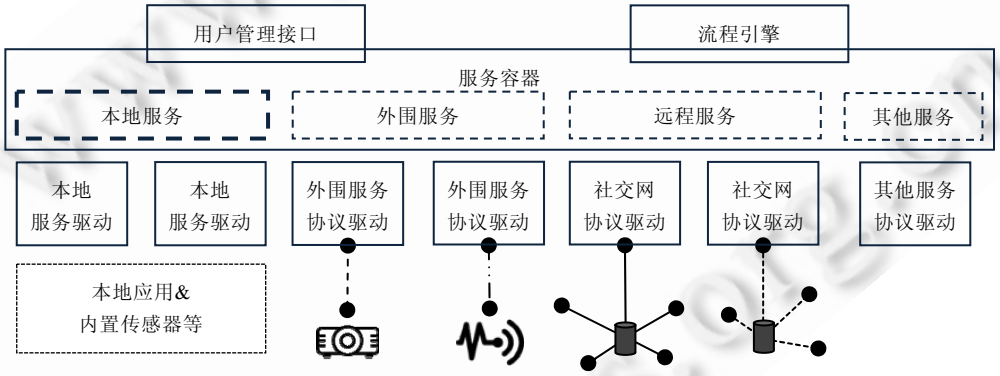


Fig.4 Overview of the framework

图 4 框架概览

在服务容器下层,为服务容器提供服务来源的是驱动模块,驱动模块是服务容器的服务来源.驱动模块包含 3 种类型的驱动——本地服务驱动、外围服务协议驱动和社交网协议驱动,它们与本地服务、外围服务和远程服务 3 种服务类型分别对应.服务驱动的主要作用是将资源转化为服务,协议驱动基于服务发现协议发现服务,并向上对服务容器提供统一的服务接口.

由服务容器对上层应用暴露的接口可知,服务容器的主要任务为服务管理和服务消费.在服务容器内部,与用户管理接口对应的是服务管理模块,其主要任务便是服务管理,它包含一个用以注册本地服务的服务注册表.我们基于 OSGi 提供的服务注册功能实现服务注册表.服务注册表中存在着一块空间有限的区域作为非本地服务的缓存,为区别于服务注册表上层的“一级缓存”,我们称其为“二级缓存”,一级缓存中存有最近被选中的(包括

本地、外围或远程及其他类型的)服务.此外,服务管理模块还实现了对“资源敏感”服务的自动管理,这种自动管理依赖于心跳模块实现的资源感知的心跳机制.该心跳模块包含一个心跳控制子模块以动态地确定心跳周期,它以资源监控模块定期收集的资源状态信息为输入,输出下一个心跳延迟.基于该心跳延迟,心跳模块将心跳包发送到服务中介.心跳包的内容除了资源监控模块收集的资源状况信息外,还包括设备当前的访问地址.支撑服务消费的主要模块是对外暴露的流程引擎接口和接口背后的流程解析模块,流程解析模块解析服务流程并提供直接的依赖管理功能,依赖管理的过程其实就是服务查找、选择和绑定的过程.流程解析过程中的主要角色是依赖管理器(dependency manager),依赖管理器自身依赖于服务注册表.

3.3 服务(协议)驱动与服务接口适配

驱动模块为服务容器提供服务来源,主要包含 3 种类型的驱动,即本地服务驱动、外围服务协议驱动和社交网协议驱动.具体而言,本地服务驱动可基于某些本地应用及内置传感器提供本地可直接访问的“标准”服务,比如内置传感器温度服务驱动可基于手机内置温度传感器提供“标准”温度服务;外围服务协议驱动往往是特定于服务发现协议[20]的,针对某种服务发现协议的驱动可支持动态发现所有可达的基于该协议发布的服务,比如 DPWS 协议驱动可发现基于 DPWS 协议发布的服务;而社交网协议驱动则是为了支持基于不同协议实现的社交网,社交网为服务共享平台提供基本的信息通道,服务的发布、查找和调用都基于该通道.比如在我们的实现过程(基于 JADE)中,该通信通道即上文所提到的 ACC.不同社交网协议驱动对外提供统一的服务发布、查找和调用接口,其内部实现则因具体的通信通道而有所不同.除了前面 3 种驱动外,该平台还支持其他类型的服务,比如 Web 服务,通过安装 Web 服务驱动,可以在需要时从指定的 UDDI 上获得所依赖的服务,后续过程与远程服务类似.

驱动不只作为服务来源,同时也承担着服务接口适配的责任.以远程服务为例,在服务查找过程中,从服务中介返回的只是远程服务的文本描述(包括服务地址,服务域,质量属性等);而要进行服务绑定,则需要将服务描述转化为实现了抽象服务接口的具体类.驱动采用了 Java 动态代理(Java dynamic proxy)技术实现从服务描述到服务接口的自动适配(更多细节见第 3.5 节服务绑定与流程执行).

所有类型的服务驱动都以 OSGi 组件的形式实现,这使得服务共享平台无需重启即可提供新的服务或支持新的协议.所有驱动都由用户手动安装或在用户授权下自动安装.我们基于 Apache Felix OBR(OSGi bundle repository)实现了一个公共驱动库,允许开发者实现并发布新的驱动.

3.4 服务查找与本地缓存

如上文所述,业务流程基于 FSMBehaviour 进行描述,并基于 Java 注解对抽象服务依赖进行描述.业务流程由服务容器提供的流程引擎执行,流程引擎基于 JADE Agent 实现.在流程引擎下层,流程解析模块为其提供直接的依赖管理服务.业务流程进入流程引擎后,流程解析模块基于 Java 注解抽取出尚未完成绑定的抽象服务,然后交由依赖管理器进行服务绑定.服务绑定的第 1 步是服务选择,依赖管理器根据 Java 注解中的依赖描述进行服务选择.从“一级缓存”开始,如果缓存命中,则查找结束;否则,从服务注册表(含“二级缓存”)中查找,如果注册表中“二级缓存”存有该抽象服务对应的具体服务,则从服务注册表(含“二级缓存”)中选择最合适的服务,被选中的服务会被缓存到“一级缓存”中;如果注册表查找失败,启动一个全局查找过程,该查找过程尝试通过社交网协议驱动和外围服务驱动发起服务查找,被选择的服务的描述会被返回并经接口适配转化为服务桩后缓存到“二级缓存”中——全局查找过程也正是“二级缓存”更新的过程,如果全局查找也失败,则整个服务查找过程失败.在全局查找过程中,对于远程服务,服务中介会根据抽象服务依赖描述采用兼顾意愿和效用的服务选择方法从服务目录中选择最合适的服务返回.

需要注意的是,服务缓存(包括一级和二级缓存)的空间以及存储的服务条目的有效期都是相当有限的,在超出给定时间后,缓存条目会被自动移除出服务缓存.这是因为在移动环境下的个人化服务共享网络中,外围及远程服务的服务质量和提供者意愿甚至服务的可用性很难得到“长期”保证,长期保存所有外围服务和远程服务没有太大意义,且需要大量存储空间.但必要的缓存能够在很多时候缩短服务查找时间.

3.5 服务绑定与远程服务请求

通过服务查找过程查找得到的都是可直接调用的“具体服务”,所有这些“具体服务”都由采用工厂模式实现的服务/协议驱动“生产”,不同的是“生产”的本地服务是按需创建的本地服务驱动实例,而外围和远程服务则是协议驱动采用 Java 动态代理技术创建的实现了抽象服务接口的动态代理对象,称为服务桩.在服务缓存中,缓存条目正是这种服务桩.服务绑定的过程是简单的本地服务实例和服务桩实例赋值的过程.

在所有抽象服务都完成绑定之后,业务流程可由 JADE Agent 执行.与本地服务相比,外围和远程服务(服务桩)的调用过程有所不同,这些服务的调用过程依赖于服务桩内部实现的“远程服务请求(remote service request, 简称 RSR)”,远程服务请求基于所支持协议的通信通道.远程服务请求与 Java 远程方法调用非常类似,我们没有直接采用 Java 远程方法调用的原因是:(1) Java 远程方法调用直接基于 Socket 而远程服务请求依赖于驱动所支持协议提供的通信通道;(2) Android 不支持 Java 远程方法调用.

3.6 外围服务实现

外围服务通常由计算能力有限的微型设备提供,它们借助近距离无线通信技术对外发布数据.在实验中,我们采用开源电子原型平台 Arduino*****开发外围服务,Arduino 便捷灵活、轻易上手.图 5 展示了一个接有“蓝牙”模块和温度传感器的 Arduino 开发板.

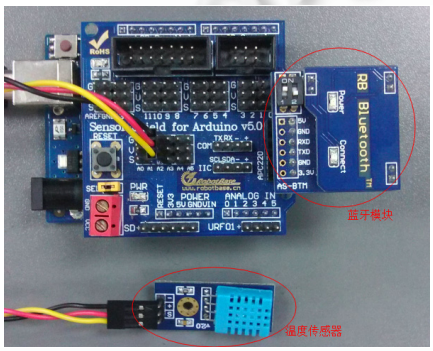


Fig.5 Arduino development board with Bluetooth module and temperature sensor connected

图 5 接有温度传感器和蓝牙模块的 Arduino 开发板

4 案例研究

在本节中,我们通过一个包含多个场景的空调报修服务进行了案例研究,对本文所提出的服务共享框架和组装方案的可行性以及兼顾服务意愿和服务效用的服务选择方法的合理性进行了验证.为简单起见,案例研究中所采用的服务选择方法全部为局部选择方法.

4.1 案例设置

小张开发了一个智能温湿度自动调节应用,该应用能够通过各种传感及控制设备实现个人化的室内温湿度及风速的自动调节,从而达到最佳舒适度.其基本原理为,通过各种传感器获取用户体温及室内温湿度等数据,根据人体舒适度原理推测目标温湿度及风速,并进一步通过空调自身提供的空调调节功能调节温湿度及风速,从而使用户达到最佳舒适状态.该流程总体如下所示:(1) 调用体温检测服务检测当前用户体温;(2) 调用温/湿度服务检测室内温度湿度;(3) 基于用户体温、室内温湿度(假设室内风速为 0),计算舒适度并判断是否需要调节以及如何调节温湿度及空调风速;(4) 调用空调调节服务调节室内温湿度及风速;(5) 定时检测是否达到期望温湿度及风速,并进一步修正调节;(6) 如果调节失败,则检查原因,在必要时将在用户授权下自动调用空调修理服务.

***** Arduino.cc. Arduino-HomePage. http://www.arduino.cc/

某个夏天的午后,小张运动完回到卧室,该应用开始工作.

4.2 步骤分析

在步骤 1 中,该应用请求小张体表温度传感器所提供的体温检测服务,这一步骤不存在服务选择,该服务以外围服务的形式完成绑定.

在步骤 2 中该应用依赖于温/湿度服务.以温度服务为例,由于需要测量的是小张所在室内的温度,所以该抽象服务的依赖描述中限定外围服务和本地服务,而不能是远程服务(缺乏精准的室内定位系统).如上所述,依赖管理器首先从“一级缓存”中尝试查找可用的温度服务,但缓存命中失败;于是从服务注册表(含“二级缓存”)查找,发现存在一个本地温度服务,该服务基于小张手机上的内置温度传感器,并由本地服务驱动封装转换而成.但“二级缓存”并不含有外围温度服务,于是发起一个全局查找过程,由于限定为外围服务和本地服务,所以全局查找只会动员所有的外围服务协议驱动,其中“蓝牙协议驱动”发现了在卧室角落里的一个温度传感器,该传感器通过蓝牙提供温度数据(外围服务),只要成功连接便可获取传感器周期性发送的温度数据及该服务的相关描述,于是该驱动基于该传感器动态创建了一个温度服务(代理)实例——服务桩,该服务桩会被缓存到“二级缓存”中.

由于本地服务对于本机调用总是允许的,所以意愿总是为 1.相比于每次调用所带来的资源消耗,该应用更关注温度服务所提供数据的准确程度.其效用函数为

$$f_U(b_{cost},a)=0.2\cdot norm_b\left(\frac{1}{b_{cost}}\right)+0.8\cdot norm_a\left(\frac{1}{a}\right),$$

其中,b_{cost}表示消耗的电量(battery cost),a 表示准确程度(accuracy),这里,我们假设准确程度可以根据设备型号获得并且不会发生变化.计算发现,蓝牙温度服务的 wu 值较高(见表 4),于是该服务被选取并被缓存在“一级缓存”中.步骤 2 中对于湿度服务的选择和调用与温度服务相同,与之类似,蓝牙湿度服务被选取.由于外围温度服务需要蓝牙通信,而本地温度服务只是简单地调用手机上的温度传感器,因此,外围服务的温度服务电量消耗要高于本地温度服务.但是,手机上的温度传感器容易受到电池温度影响,所以其准确程度比较低.计算得知,外围温度服务的 wu 值更高,故而被选取.

Table 4 Service willingness and utility computing of Step 2

表 4 步骤 2 的效用和意愿计算

提供者	意愿 W	效用计算		$W\cdot U$
	$f_w(\omega=1)$	效用相关服务特征	U	
蓝牙温度服务	1	b_{cost}=2160mJ, a=0.85	0.8	**0.8**
本地温度服务	1	b_{cost}=260mJ, a=0.4	0.2	0.2

步骤 3 需要调用舒适度计算服务计算舒适度并判断是否需要调节以及如何调节.在此步骤中,同样会发起一个全局查找过程,其中通过社交圈服务中介找到了 3 个舒适度计算服务,且这 3 个服务的提供者对 3 种资源的单项资源意愿函数(见表 5)及所有资源的 D_R 都相同(为 d_b=0.7,d_t=0.4).其中,资源意愿函数在用户加入到该服务圈时,由用户手动设置并被同步到服务中介,并在需要时由用户手动更新.这些意愿函数依赖的动态属性及质量参数值,依赖于资源感知的心跳机制动态发送到服务中介,远程服务的意愿和效用计算由服务中介完成.除了这些远程服务,在注册表中发现了一个本地的舒适度计算服务,而通过 Web 服务协议驱动通过 UDDI 还找到了一个舒适度计算 Web 服务,本地服务和 Web 服务的服务提供意愿总是为 1.

对于 3 个候选远程服务,服务 1 提供的舒适度计算更加专业,考虑更全面,数据更精准,所以效用略高,效用计算过程不再赘述,但因该服务的提供设备电量有限(仅剩 550mAh),对该服务的服务提供意愿计算后值为 0.远程服务的效用普遍很高,原因在于大部分耗资源的任务都在服务提供设备上完成,小张的设备只负责发出请求、接收结果;而 Web 服务需要在本地解析 xml.综合考虑意愿和效用之后,最后选择了服务 2(见表 6).可以看到,效用不再是唯一指标,并不总是效用最高的服务被选取.

Table 5 Resource willingness function

表 5 单项资源意愿函数

单一资源	单项资源意愿函数
电量	$f_{W,b}(v_b)=\begin{cases}1, & v_b \geqslant 4.35\\ \dfrac{v_b-0.9}{3.45}, & 0.9<v_b<4.35,\ v_b=\dfrac{b_l}{d_b\cdot b_c}\cdot\dfrac{mJ}{mAh}\\ 0, & \dfrac{b_l}{d_b}\leqslant 600 \,\|\, v_b\leqslant 0.9\end{cases}$
网络流量	$f_{W,t}(v_t)=\begin{cases}1, & v_t \geqslant 20\\ \dfrac{v_t-1}{19}, & 1<v_t<20,\ v_t=\dfrac{t_l}{d_t\cdot t_c}\cdot\dfrac{KB}{MB}\\ 0, & \dfrac{t_l}{d_t}\leqslant 20 \,\|\, v_t\leqslant 1\end{cases}$
内存	$f_{W,m}(v_m)=\begin{cases}\dfrac{m_l-m_c}{m_l}, & m_l>100 \,\&\&\, 10<v_m<150\\ 0, & m_l\leqslant 100 \,\|\, v_m\leqslant 10\end{cases}$

Table 6 Service willingness and utility computing of Step 3

表 6 步骤 3 的效用和意愿计算

服务类型		意愿					效用	$W\cdot U$
		资源状况	服务特征	f_W		W	U	wu
				$f_{W,R}$	$f_{W,P}$(小张)			
本地服务		–	–	f_W=1		1	0.55	0.55
Web 服务		–	–	f_W=1		1	0.60	0.60
远程服务	服务1	b_l=550mAh t_l=48.5MB m_l=50MB	b_c=550mJ t_c=4.0KB m_c=2.0MB	0	1.0	0	0.95	0
	服务2	b_l=1260mAh t_l=32.5MB m_l=486MB	b_c=420mJ t_c=4.5KB m_c=4.0MB	0.87	0.8	0.696	0.90	**0.63**
	服务3	b_l=1750mAh t_l=25.6MB m_l=258MB	b_c=650mJ t_c=4.0KB m_c=3.0MB	0.632	1.0	0.632	0.70	0.44

注:b,t,m 分别代表电量、流量和内存,b_l 表示剩余电量,b_c 表示该服务消耗的电量,以此类推

舒适度计算服务同时也将判断是否需要进行调节以使得用户体感舒适度最佳,以及如何调节,这包括调节应达到的目标湿度、目标温度以及目标风速.调节的过程即步骤 4 所示的调用空调自身提供的空调调节服务.在发起调节之后,该应用会定时检测是否达到期望温、湿度及风速(步骤 5),这一过程的服务选择类同于步骤 1 和步骤 2.但因为一、二级缓存的存在,这些服务能够缓存命中,所以能够在一定程度上缩短服务查找时间.如果检测发现已经达到期望值,则本次调节结束.

但经过 10 分钟的多次检查,发现温、湿度依然没有达到期望值,经检测发现其原因较大可能是空调自身出现了问题.于是该应用在小张授权下自动发起一个空调修理服务请求(步骤 6).修理服务是一个位置相关的服务,而此类服务的选择需要依赖提供两个位置相关的参数:目标位置和范围限定.由于天气炎热,小张希望空调修理师能够尽快上门修理,综合考虑多方面因素,小张将限定范围设置为 5km.同时,该应用会自动调用位置服务以获得当前位置作为目标位置.目标位置和范围限定都会作为该位置相关服务——空调修理服务的依赖描述.小张最希望的是能够尽快把空调修理好,因此修理工离自己的距离成为影响效用的最大因素,其效用函数如下:

$$f_U(dist,price,level)=0.4\times norm_d\left(\frac{1}{dist}\right)+0.3\times norm_p\left(\frac{1}{price}\right)+0.3\times norm_l(level).$$

由于空调修理服务是一项人工服务,所以受到服务提供者主观权重的影响.夏季是空调修理繁忙季节,而修理师 1 的修理水平又高,虽然价格也略高,但仍然预约不断,一上午的繁忙之后,他希望午后能够休息一会儿,所

以主观权重ω相对略低.而修理师 2 因为上午比较空闲,所以主观权重仍然较高.相比而言,最终修理师 2 提供的修理服务被选中(见表 7).

Table 7 Maintenance service selection

表 7 修理服务选择

修理师	意愿 W		效用 U		$W\cdot U$
	f_w(小张)	ω	效用相关服务特征	U	
修理师1	1	0.7	*dist*=2.0km,*price*=100,*level*=0.95	0.70	0.49
修理师2	1	1	*dist*=2.8km,*price*=90,*level*=0.87	0.51	**0.51**
修理师3	1	0.8	*dist*=3.5km,*price*=95,*level*=0.90	0.53	0.42

4.3 案例总结

从服务消费角度来说,服务共享平台使得业务流程设计简化.业务流程只需关注所需要的抽象服务,而无需关心在流程执行过程中所调用的服务的真正来源甚至服务类型,这使得服务之间的耦合程度大为降低,并增加了服务绑定的灵活性,从而一定程度上实现了对不同类型服务的有效整合利用.比如在步骤 3 中,既可以使用本地服务,也可以将 Web 服务或者好友提供的远程服务绑定到抽象的舒适度计算服务上.另一方面,实现了依赖注入的服务容器将服务及服务桩的管理从服务流程中剥离,在减轻服务流程负担的同时,借助服务缓存缩短服务查找时间.比如由于步骤 2 缓存了查找得到的温度服务,步骤 5 的服务查找时间被大大缩短.

从服务提供角度来看,只需在进入服务共享网络时设置好对不同资源的单项意愿函数及隐私意愿函数,便可方便地实现对服务请求的有效控制,比如在步骤 3 中,服务 1 的提供设备电量只剩下 550mAh,如果使用的是传统的单考虑效用的服务选择方法,该服务反而会被选中,即便服务 1 的提供设备电量和内存已经非常紧张,而来自外部的服务调用很有可能会进一步加剧该设备的资源紧张情况,很明显这是不合理的.而如果采用兼顾意愿和效用的服务选择方法,根据预先设定的电量资源意愿函数计算得到意愿为 0,根据计算公式可知其 *wu* 值最后也为 0,即便其服务效用非常高,该服务仍然不会被选中,这就有效地保证了服务提供设备的正常使用.同样地,正确设置隐私意愿函数可以在一定程度上实现对个人隐私的有效保护.综合来看,相比于传统的基于效用的服务选择方法,兼顾意愿和效用的服务选择方法更加合理.另一方面,服务效用及意愿计算包括服务选择(从服务提供者角度来说,称为“请求控制”更为恰当,因为在服务提供设备资源形式严峻或隐私要求较高时,服务中介会由于意愿较低而拦截某些请求)的所有过程都是在资源充分的服务中介(服务器)上完成的,而无需移动设备来承担这些过程的资源消耗,从而有效地降低了移动设备的负载,使得移动设备更加轻捷.

从服务选择过程来看,兼顾意愿和效用的服务选择方法可在很大程度上反映服务提供者的真正意图.继续以步骤 3 为例,资源状况差的移动设备提供服务的意愿也往往更低,相比于另外两个服务,服务 1 的总体资源状况更差,而其计算得到的意愿值也是三者中最低的.

4.4 讨 论

在本文的研究中,我们假设用户之间仅仅凭借社交关系所提供的感情和信任基础而无偿地提供服务.然而,在现实的服务提供中,可能存在着某些奖励和补偿机制,使得服务提供者的意愿将不再只是受到资源状况或者信任关系的影响,而同时会受到利益的影响.在这种情况下,即使在资源状况很差的情况下,移动用户受到利益(奖励、补偿)的驱动,其提供服务的意愿可能仍然很高,比如案例研究中的空调修理服务,受利益驱使,服务提供意愿普遍很高.

此外,在目前的框架实现中,设备资源状况信息是通过心跳机制主动提供的.而另外一种可能的方案是按需获取,即服务中介只有在接收到请求时才去向服务提供者请求其状况信息.这种方案的好处是:相比于主动提供的方式,按需获取能够减少不必要的通信开销.然而,这种按需获取的方案也存在着以下几个问题:其一,服务中介每接收到一个服务请求之后,都需要请求并等待该服务的所有可能提供者返回其状况信息,这是一个耗时的过程,服务消费者往往无法接受.其二,移动设备网络环境的动态性意味着服务中介很可能根本无法寻址到这些服务提供者,也即无法获取到这些设备的状况信息.其三,服务中介无法确认服务提供者的移动设备是否处于“存活”状态还是处于“死亡”,所以也就无法确定是否应“休眠”该设备提供的服务,从而造成较高的服务失效率.

5 相关工作

受制于移动设备的资源有限性,目前关于移动服务提供的研究大多集中于实现适用于移动平台的轻量级服务提供框架,且大多数研究都是基于传统 Web 服务的.已有的移动 Web 服务提供框架,可以根据提供的 Web 服务类型分为两类:基于 SOAP 消息的 Web 服务提供框架[2,21−23]和 Restful Web 服务提供框架[24−26].文献[27]的研究结果表明,轻量级的 Restful Web 服务在性能上更适合移动平台,但更加成熟的基于 SOAP 消息的移动 Web 服务仍然占据大多数.已有的移动 Web 服务提供框架也可以根据网络结构分为 P2P[2,27,28]和 C/S[29]两种结构类型.由于缺乏固定的中心管理节点,移动 P2P 网络环境下的服务发现通常都是通过广播服务请求(request-broadcast)或者广播服务广告(advertisement-broadcast)方式来实现[30],这两种方式都会造成短暂且大量的网络通信且难以保证全局最优,相比于 C/S 结构往往更加耗时耗电.本文所提出的个人化服务发布和共享方法是基于服务中介的集中式架构,所以只适用于 C/S 网络结构;但结合协议驱动(上层应用只关注服务接口,协议驱动可屏蔽网络结构差异),可实现对 P2P 网络结构的支持.

此外,许多研究学者针对传统的 Web 服务提出了许多服务选择方法,主要可以分为 4 类:局部的、全局的、混合局部和全局的以及基于启发式规则的服务选择方法.其中,局部选择方法[15,31,32]会为一个业务流程中的每个服务从候选服务集中选择一个整体效用最优的服务.由于只考虑单个服务,局部选择方法的复杂度通常都是线性的(即与候选服务的数量有关).但是,每个服务的单个最优实现组合在一起后并不总是能够满足全局的服务质量约束,因此局部选择方法通常都不适用于具有全局服务质量约束的服务组合问题.而全局选择方法[15−37]通常会把服务选择问题转化为整数规划问题,即最大化组合效用且满足各个服务质量的全局约束.然而,全局选择方法如果不进行优化,则需要遍历所有可能的候选方案以找到最优方案,因此它的时间复杂度相当高.针对这一问题,文献[38]提出了一种基于启发式规则的服务选择方法,而文献[39,40]提出了将局部选择和全局选择混合的方法,它们都能够以较少的时间开销找到一个近似最优解.但几乎所有这些已有的服务选择方法都主要关注于如何更快地为服务请求者选择一组效用最优或者近似最优的服务.然而,在移动环境下,考虑到提供服务的移动设备通常是资源有限的,如果仅从服务请求者的角度考虑服务效用而不考虑服务提供者的意愿,可能会造成提供服务的设备资源过度消耗,影响服务提供者的正常使用.而本文所提出的服务选择方法增加了对服务提供者意愿的考虑,从而更加适用于移动服务环境.

6 结束语

本文提出了一种移动计算环境下的个人化服务发布和组装方法.首先,我们在个人化服务发布过程中引入了服务意愿的概念,并在服务选择过程中综合考虑了服务意愿和服务效用.这种服务选择方法能够综合考虑受服务提供者的移动设备资源状况、所提供的服务对资源的消耗情况、其隐私策略等因素影响的对单次服务提供的主观愿意程度以及服务请求者对于各个服务质量的综合效用值,从而得到服务意愿和服务效用之间的一种合理平衡.其次,我们提出了一种资源感知的心跳机制来定期更新服务选择所依赖的设备及服务状态信息,从而确保用于服务选择的状态信息的实时性.这种心跳机制能够根据移动设备的资源状况动态地确定心跳周期,即资源状况好时,心跳周期就短;资源状况差时,心跳周期就长.这种心跳机制能够在一定程度上减少不必要的心跳次数,从而在减少电量、网络流量等资源;同时这种心跳机制可以解决移动设备的移动性所带来的服务失效问题.最后,基于本文所提出的方法,我们提出了一个基于 JADE 和 OSGi 技术的实现框架来支持个人化服务发布以及组装.通过一个案例分析,我们还验证了本文所提出的实现框架的有效性以及在移动计算环境下考虑服务意愿的合理性.

References:

[1] Tergujeff R, Haajanen J, Leppanen J, Toivonen S. Mobile SOA: Service orientation on lightweight mobile devices. In: Proc. of the Int'l Conf. on Web Services. IEEE, 2007. 1224−1225. [doi: 10.1109/ICWS.2007.121]

[2] Srirama SN, Jarke M, Prinz W. Mobile Web service provisioning. In: Proc. of the Int'l Conf. on Internet and Web Applications and Services/Advanced Int'l Conf. on Telecommunications. IEEE, 2006. 120−120. [doi: 10.1109/AICT-ICIW.2006.215]

[3] OSGi Alliance. OSGi service platform, core specification, release 4, version 4.1. OSGi Specification, 2007. http://www.osgi.org

[4] Keijzers J, den Ouden E, Lu Y. Usability benchmark study of commercially available smart phones: Cell phone type platform, PDA type platform and PC type platform. In: Proc. of the 10th Int'l Conf. on Human Computer Interaction with Mobile Devices and Services. ACM, 2008. 265−272. [doi: 10.1145/1409240.1409269]

[5] Chang YF, Chen CS. Smart phone—The choice of client platform for mobile commerce. Computer Standards & Interfaces, 2005, 27(4):329−336. [doi: 10.1016/j.csi.2004.10.001]

[6] Nichols J, Myers BA. Controlling home and office appliances with smart phones. Pervasive Computing, 2006,5(3):60−67. [doi: 10.1109/MPRV.2006.48]

[7] Lane ND, Miluzzo E, Lu H, Peebles D, Choudhury T, Campbell AT. A survey of mobile phone sensing. Communications Magazine, 2010,48(9):140−150. [doi: 10.1109/MCOM.2010.5560598]

[8] Gellersen HW, Schmidt A, Beigl M. Multi-Sensor context-awareness in mobile devices and smart artifacts. Mobile Networks and Applications, 2002,7(5):341−351. [doi: 10.1023/A:1016587515822]

[9] Ailisto H, Pohjanheimo L, Väïkkynen P, Strömmer E, Tuomisto T, Korhonen I. Bridging the physical and virtual worlds by local connectivity-based physical selection. Personal and Ubiquitous Computing, 2006,10(6):333−344. [doi: 10.1007/s00779-005-0057-0]

[10] Strommer E, Kaartinen J, Parkka J, Parkka J, Ylisaukko-Oja A, Korhonen I. Application of near field communication for health monitoring in daily life. In: Proc. of the 28th Annual Int'l Conf. of the IEEE, Engineering in Medicine and Biology Society (EMBS). IEEE, 2006. 3246−3249. [doi: 10.1109/IEMBS.2006.260021]

[11] Berger S, McFaddin S, Narayanaswami C, Raghunath M. Web services on mobile devices-implementation and experience. In: Proc. of the 5th IEEE Workshop on Mobile Computing Systems and Applications. IEEE, 2003. 100−109. [doi: 10.1109/MCSA.2003.1240771]

[12] Allard J, Chinta V, Gundala S, Richard III GG. Jini meets UPnP: An architecture for Jini/UPnP interoperability. In: Proc. of the 2003 Symp.on Applications and the Internet. IEEE, 2003.268−275. [doi: 10.1109/SAINT.2003.1183059]

[13] Benatallah B, Dumas M, Sheng QZ, Ngu AHH. Declarative composition and peer-to-peer provisioning of dynamic Web services. In: Proc. of the Int'l Conf. on Data Engineering. IEEE, 2002. 297−308. [doi: 10.1109/ICDE.2002.994738]

[14] Li F, Yang F, Shuang K, Su S. Q-Peer: A Decentralized QoS Registry Architecture for Web Services. Berlin, Heidelberg: Springer-Verlag, 2007. [doi: 10.1007/978-3-540-74974-5_12]

[15] Zeng L, Benatallah B, Ngu AHH, Dumas M, Kalagnanam J, Chang H. QoS-Aware middleware for Web services composition. IEEE Trans. on Software Engineering, 2004,30(5):311−327. [doi: 10.1109/TSE.2004.11]

[16] Ardagna D, Pernici B. Global and local QoS constraints guarantee in Web service selection. In: Proc. of the 2005 IEEE Int'l Conf. on Web Services. IEEE, 2005. [doi: 10.1109/ICWS.2005.66]

[17] Moreno A, Valls A, Viejo A. Using JADE-LEAP implement agents in mobile devices. EXP in Search of Innovation (Special Issue on JADE), 2003.

[18] Caire G, Pieri F. Leap user guide. TILab, 2006. http://jade.cselt.it/doc/tutorials/LEAPUserGuide.pdf

[19] Fortino G, Garro A, Russo W. From modeling to enactment of distributed workflows: An agent-based approach. In: Proc. of the 2006 ACM Symp. on Applied Computing. ACM, 2006. 128−129. [doi: 10.1145/1141277.1141306]

[20] Mian AN, Baldoni R, Beraldi R. A survey of service discovery protocols in multihop mobile ad hoc networks. Pervasive Computing, 2009,8(1):66−74. [doi: 10.1109/MPRV.2009.2]

[21] Zeeb E, Bobek A, Bohn H, Golatowski F. Service-Oriented architectures for embedded systems using devices profile for Web services. In: Proc. of the 21st Int'l Conf. on Advanced Information Networking and Applications Workshops. IEEE, 2007,1:956−963. [doi: 10.1109/AINAW.2007.330]

[22] Asif M, Majumdar S, Dragnea R. Hosting Web services on resource constrained devices. In: Proc. of the IEEE Int'l Conf. on Web Services. IEEE, 2007. 583−590. [doi: 10.1109/ICWS.2007.97]

[23] Pham L, Gehlen G. Realization and performance analysis of a SOAP server for mobile devices. In: Proc. of the 11th European Wireless Conf. 2005-Next Generation Wireless and Mobile Communications and Services (European Wireless). VDE, 2005. 1−7.

[24] Hamad H, Saad M, Abed R. Performance evaluation of RESTful Web services for mobile devices. Int'l Arab Journal of e-Technology, 2010,1(3):72−78.

[25] Aijaz F, Ali SZ, Chaudhary MA, Walke B. Enabling high performance mobile Web services provisioning. In: Proc. of the IEEE 70th Vehicular Technology Conf. Fall (VTC 2009-Fall). IEEE, 2009. 1−6. [doi: 10.1109/VETECF.2009.5378949]

[26] AlShahwan F, Moessner K. Providing soap Web services and restful Web services from mobile hosts. In: Proc. of the 2010 5th Int'l Conf. on Internet and Web Applications and Services (ICIW). IEEE, 2010. 174−179. [doi: 10.1109/ICIW.2010.33]

[27] Mizouni R, Serhani MA, Dssouli R, Benharref A, Taleb I. Performance evaluation of mobile Web services. In: Proc. of the 2011 9th IEEE European Conf. on Web Services (ECOWS). IEEE, 2011. 184−191. [doi: 10.1109/ECOWS.2011.12]

[28] Gehlen G, Pham L. Mobile Web services for peer-to-peer applications. In: Proc. of the 2005 2nd IEEE Consumer Communications and Networking Conf. (CCNC). IEEE, 2005. 427−433. [doi: 10.1109/CCNC.2005.1405210]

[29] Adacal M, Bener AB. Mobile Web services: A new agent-based framework. Internet Computing, 2006,10(3):58−65. [doi: 10.1109/MIC.2006.59]

[30] Helal S, Desai N, Verma V, Lee C. Konark—A service discovery and delivery protocol for ad-hoc networks. In: Proc. of the 2003 IEEE Wireless Communications and Networking (WCNC). IEEE, 2003,3:2107−2113. [doi: 10.1109/WCNC.2003.1200712]

[31] Liu Y, Ngu AH, Zeng LZ. QoS computation and policing in dynamic Web service selection. In: Proc. of the 13th Int'l World Wide Web Conf. on Alternate Track Papers & Posters. ACM, 2004. 66−73. [doi: 10.1145/1013367.1013379]

[32] Menascé DA, Dubey V. Utility-Based QoS brokering in service oriented architectures. In: Proc. of the IEEE Int'l Conf. on Web Services (ICWS). IEEE, 2007. 422−430. [doi: 10.1109/ICWS.2007.186]

[33] Yang K, Henning I, Ou S, Azmoodeh M. Model-Based service discovery for next-generation mobile systems. Communications Magazine, 2006,44(9):122−129. [doi: 10.1109/MCOM.2006.1705988]

[34] Chakraborty D, Joshi A, Finin T, Yesha Y. Service composition for mobile environments. Mobile Networks and Applications, 2005, 10(4):435−451. [doi: 10.1007/s11036-005-1556-y]

[35] Cardellini V, Casalicchio E, Grassi V, Lo Presti F, Mirandola R. Qos-Driven runtime adaptation of service oriented architectures. In: Proc. of the the 7th Joint Meeting of the European Software Engineering Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering. ACM, 2009. 131−140. [doi: 10.1145/1595696.1595718]

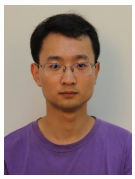
[36] Zeng L, Benatallah B, Dumas M, Kalagnanam J, Sheng QZ. Quality driven Web services composition. In: Proc. of the 12th Int'l Conf. on World Wide Web. ACM, 2003. 411−421. [doi: 10.1145/775152.775211]

[37] Ardagna D, Pernici B. Adaptive service composition in flexible processes. IEEE Trans. on Software Engineering, 2007,33(6): 369−384. [doi: 10.1109/TSE.2007.1011]

[38] Yu T, Zhang Y, Lin KJ. Efficient algorithms for Web services selection with end-to-end QoS constraints. ACM Trans. on the Web (TWEB), 2007,1(1):6. [doi: 10.1145/1232722.1232728]

[39] Alrifai M, Risse T. Combining global optimization with local selection for efficient QoS-aware service composition. In: Proc. of the 18th Int'l Conf. on World Wide Web. ACM, 2009. 881−890. [doi: 10.1145/1526709.1526828]

[40] Sun SX, Zhao J. A decomposition-based approach for service composition with global QoS guarantees. Information Sciences, 2012, 199:138−153. [doi: 10.1016/j.ins.2012.02.061]



王明良(1990－),男,湖南株洲人,硕士生,主要研究领域为软件工程,软件自适应.

陈碧欢(1986－),男,博士生,主要研究领域为自适应系统.

彭鑫(1979－),男,博士,副教授,CCF 高级会员,主要研究领域为软件工程,系统软件.

黄罡(1975－),男,博士,教授,博士生导师,CCF 会员,主要研究领域为系统软件,软件自适应.

赵文耘(1964－),男,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,企业应用集成,电子商务.