ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.12, December 2007, pp.2967−2979
DOI: 10.1360/jos182967

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

一个基于服务层叠网的分层服务组合框架*

李 扬+, 怀进鹏, 郭慧鹏, 杜宗霞

(北京航空航天大学 计算机学院,北京 100083)

A Hierarchical Service Composition Framework Based on Service Overlay Networks

LI Yang+, HUAI Jin-Peng, GUO Hui-Peng, DU Zong-Xia

(School of Computer Science and Engineering, BeiHang University, Beijing 100083, China)

+ Corresponding author: Phn: +86-10-82339187, Fax: +86-10-82316796, E-mail: liyang@act.buaa.edu.cn, http://www.act.buaa.edu.cn

Li Y, Huai JP, Guo HP, Du ZX. A hierarchical service composition framework based on service overlay networks. *Journal of Software*, 2007,18(12):2967−2979. http://www.jos.org.cn/1000-9825/18/2967.htm

Abstract: As the amount of Web services over the Internet grows continuously, these services can be interconnected to form a service overlay network (SON). On the basis of SON, building value-added services by service composition is an effective method to satisfy the changeable functional and non-functional QoS (quality of service) requirements of customers. However, the previous research on QoS-aware service composition in SON mainly focuses on the context where services have simple interactions, and it can not support application scenarios with complex business collaboration in electronic commerce. This paper proposes the HOSS (hierachical service composition framework based on Service overlay networks) scheme based on active service overlay network (ASON) which is a kind of programmable SON. HOSS can be used to construct more general-purpose SON through describing the relations among services using business protocols. In HOSS, business protocols instead of interactive messages are adopted to simplify the description of service composition requirements and are mapped to dynamical user perspective of SON to implement service composition on demand.

Key words: service composition; service overlay network; protocol; protocol composition; user-programmable

摘 要: 随着 Internet 上 Web 服务数量的不断增长,这些服务能够互联形成一个应用层的逻辑网络——服务层叠网(service overlay network,简称 SON).基于 SON,通过服务的组合提供增值服务,是满足用户动态、多变的功能及非功能需求的一种有效的方法.但是,已有基于 SON 的 QoS 感知的服务组合研究主要是面向服务间具有简单交互行为的应用领域,难以支持电子商务等具有复杂业务协作特征的应用场景.为此,通过使用业务协议刻画服务间的组合关系,从而构建更具普适性的 SON;建立主动服务层叠网(active service overlay network,简称 ASON)以实现可编程的服务层叠网,支持按需的服务组合;提出了一个基于主动服务层叠网的分层服务组合框架(hierachical service composition framework based on service overlay networks,简称 HOSS),通过将业务协议(而不是消息)作为需求描述的

* Supported by the National High-Tech Research and Development Plan of China under Grant Nos.2007AA010301, 2006AA01Z19A (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2005CB321803 (国家重点基础研究发展计划(973)); the National Natural Science Funds for Distinguished Young Scholar of China under Grant No.60525209 (国家杰出青年基金)

Received 2007-06-11; Accepted 2007-10-16

基本单元,提升软件开发效率,并将服务组合需求映射为 SON 的动态用户视图以实现按需的组合.

关键词: 服务组合;服务层叠网;协议;协议组合;用户可编程
中图法分类号: TP393 文献标识码: A

随着 SOC(service-oriented computing)应用的日益广泛,部署的 Web 服务不断丰富,逐渐能够在底层物理网络之上针对不同的应用需求构建虚拟的应用层逻辑网络——服务层叠网(service overlay network,简称 SON).SON 的构建,意味着 Internet 已经从简单的提供主机间互联,逐步演化成为一个支撑 SOC 的应用基础架构.SON 区别于传统的内容分发层叠网络以及 P2P 的文件共享层叠网络,不仅能够适用于应用级的数据路由,而且能够提供增值的服务.

由于 Web 服务具有的自治和异构性的特点以及面向开放互联网的服务组合环境,如何确保组合服务满足应用的功能性及非功能性需求,一直是 SOC 应用所面临的巨大挑战.基于 SON 的服务组合方法,通过针对不同的应用需求构建可管理的 SON,可以在一个确定的系统边界内,借助 service level agreement(SLA)[1]有效地管理参与组合的各服务组件的非功能(QoS)属性[2];而且通过服务间固有的交互协作关系,如输入和输出的匹配关系建立服务间的虚拟链接[3],能够保证参与协作的服务的功能一致性;最终通过将满足复杂、多变的松耦合应用需求映射为 SON 中路由的方式,便捷、可靠地实现服务的组合.因此,基于 SON 进行服务的组合已日益成为互联网环境下面向服务进行软件开发的一种有效方法.

但是目前,基于 SON 的服务组合研究主要集中于多媒体领域,难以适应电子商务等更加广泛的应用场景,主要表现在:

1) SON 的构建:在传统的 SON 中,为避免服务间的功能失配所导致的服务组合失败,通常基于服务的输入和输出关系进行匹配,从而建立服务节点之间的链接.这种方式比较适合于面向数据、具有简单行为逻辑的场景.但是在复杂的业务应用场景,如电子商务应用中,服务间更多地表现为一种复杂、多回合的交互关系,很难基于这种简单的方式构造 SON,因此限制了在其上进行服务组合的应用领域.

2) 需求规约:现有的服务组合研究中通常采用流模式的组合描述语言,如 business process execution language for Web services (BPEL)和 Web ontology language for services (OWL-S).它们通常采用命令式的描述语言,类似于大型主机的任务控制语言 job control languages (JCLs)[4],需要描述大量的服务组合细节,从而加大了需求描述的复杂度.

3) SON 中的服务组合:在 SON 中进行服务组合的机制通常是将服务组合问题转化为 SON 中的最短路径问题.但是,在服务间具有复杂交互关系的应用场景下,应用需求难以映射为 SON 中的一条简单的路径.另一方面,电子商务应用中,甚至每天都会增加无法预知的业务需求[5],这种需求的高度动态性和不确定性决定了预先建立的 SON 不可能满足未知的大量用户需求,也就无法通过路径的查找实现服务的组合.因此,需要一种在现有 SON 基础上实现按需服务组合的机制.

针对上述问题,我们提出了一种基于 SON 的分层服务组合框架(hierachical service composition framework based on service overlay networks,简称 HOSS):

1) 将业务协议视为松耦合的服务组合关系的自然表达,并基于服务间固有的业务协议关系在现有服务的基础上建立业务域;
2) 在业务域的基础上构建一种可编程的服务层叠网——主动服务层叠网(active service overlay network,简称 ASON),以支持按需的服务组合;
3) 基于 ASON 提出了一种服务组合映射机制,以协议作为基本单元,通过组合协议的方式描述功能需求,然后在主动式和需求间建立用户视图层,根据需求动态生成 ASON 对于用户的视图,以支持需求到 ASON 的平滑映射.

本文第 1 节给出 ASON 的概要介绍.第 2 节详细阐述在 HOSS 框架中如何基于 ASON 进行服务的组合.第 3 节给出框架的实现机制.第 4 节对 HOSS 框架进行分析和评估.第 5 节介绍国内外相关的研究现状.最后对全

文进行总结并对未来的工作进行展望.

1 主动服务层叠网(ASON)

1.1 问题描述

在传统的 SON 中,为了满足用户的不同需要,需要预先建立复杂的网络拓扑,用户任意选定一个路由就可以得到一个组合服务.但是,这样的层叠网很难保证满足用户所有的需求.在如图 1 所示的层叠网中,Plane+Hotel+Car 的组合服务就无法提供.因此,网络中为了能够满足更多的用户需求,需要建立全连通网络.显然,这会增加网络拓扑的复杂度,从而提高系统的开销.

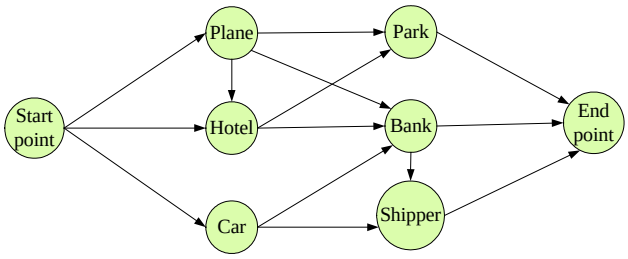


Fig.1 An example of SON

图 1 一个服务层叠网实例

另外,即使是构建一个非常复杂的网络,也不可能保证能够满足所有的需求.这是因为当有复杂的交互行为时,即使是同样的路由,即由相同功能的组件服务所构成的组合服务,也会产生不同的结果.在如图 1 所示的 SON 中,Plane+Bank+Shipper 的组合服务,可以根据不同的组合方式表现出不同的功能和外部交互行为特征.图 2 展示了两个业务协议:Customer 同 Merchant 和 Shipper 服务间进行协作的在线购物协议,Payer 与 Payee 和 Bank 服务间进行协作的信用卡支付协议时以及这两个协议根据不同的需求组合形成的两种不同的组合协议:信用卡支付的预付费购物协议和信用卡支付的后付费购物协议.

对于电子商务等需求变化剧烈的应用场景,这种行为上的不同组合方式往往是无法预知的,因此,组合服务提供者难以预先为所有这样的组合方式进行建模.因此,传统的基于 SON 进行服务组合的方法并不能完全适应具有复杂交互行为,且业务动态变化较为剧烈的应用场景.为此,我们需要构建一种能反映复杂交互行为的 SON,同时该 SON 能够根据用户的不同需求动态地进行调整.

1.2 ASON原理

为了降低网络的拓扑关系复杂度以及提高对需求的动态适应性,我们提出 ASON.除了与传统的服务层叠网具有相同的特性,提供可靠的 QoS 保证以及高效的服务组合以外,ASON 还体现出如下的新特性:

- 基于协议的服务间关系

目前,较为成熟的商业领域通常都会制定大量通用、标准的业务协议(以下简称协议).这些协议定义了业务交互的可发布规范,以规范服务间的协作,避免交互行为的不一致导致的服务组合失败.由于这些协议自然地反映了参与交互的各服务间相对固定的协作关系,因此在 ASON 中,可以将服务组件视为节点,基于这些协议构建 SON 中服务间的链接.由于有着相近的业务目标,遵循同一协议的服务按照其所担当的不同角色组成了一个协作相对紧密的业务域.这种基于协议的业务域能够表达大多数现有的服务层叠网所无法表达的服务间复杂的交互关系.在 ASON 中,一个服务可以同时支持多个协议.

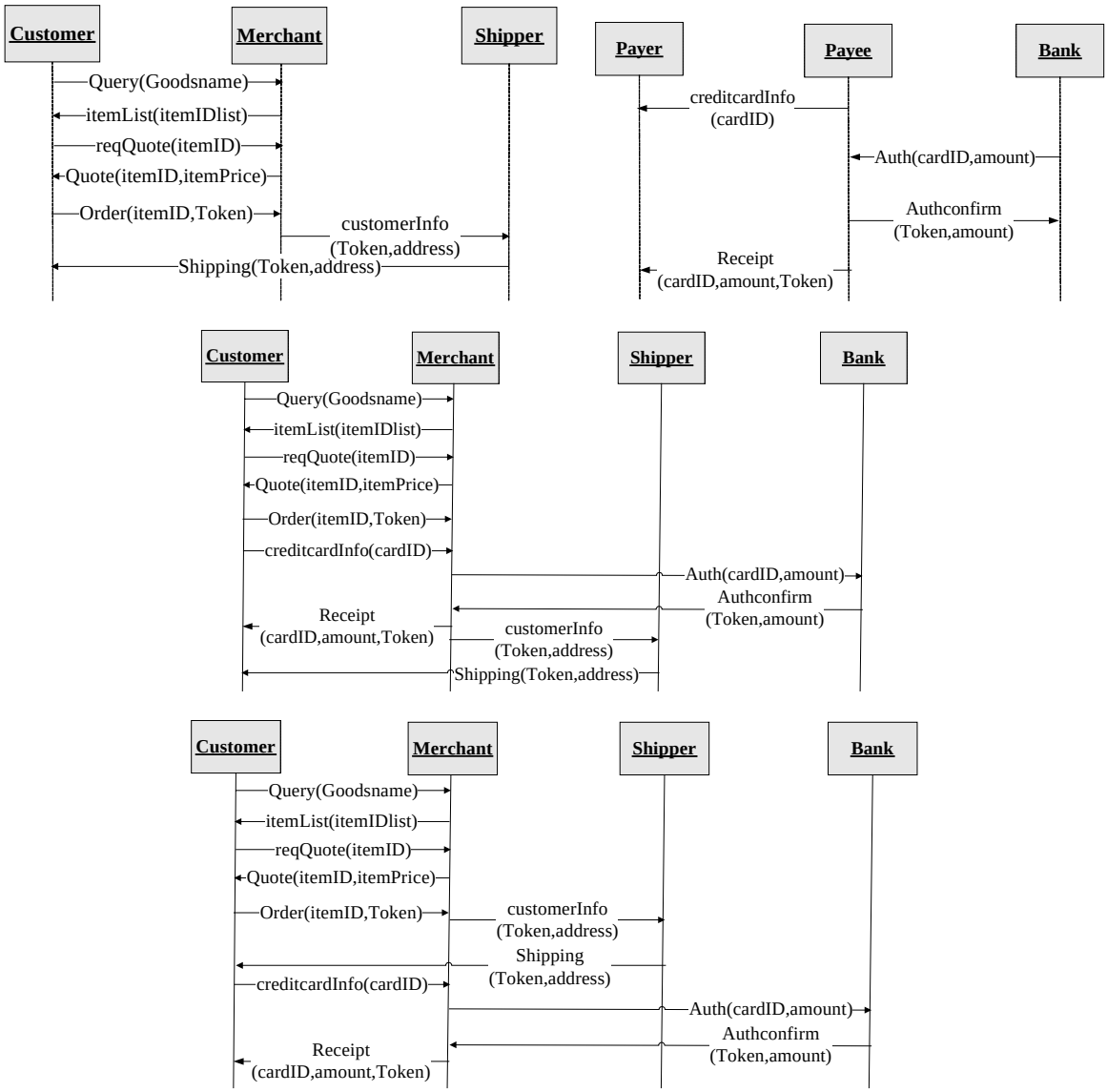


Fig.2 Protocol composition

图 2 协议的组合

- 可编程的服务层叠网

在 ASON 中,我们以协议为基本单元将整个 ASON 分为多个子网——业务域.当用户的需求只涉及单一的协议时,如用户只想完成简单的在线购物,则服务的组合只发生一个业务域内,可以遵照该协议自动地进行服务编排(choreography).但是通常情况下,用户的需求较为复杂,往往需要跨多个业务域的协作.另外,这种跨业务域的用户需求又是极为个性化的,而且难以预测.为此,ASON 区分两类服务节点:普通服务和主动服务(active service).普通服务是仅支持 1 个协议的服务,它的对外交互行为可以由其所支持的协议唯一地确定;主动服务同时支持多个协议,而这些协议的组合方式是由用户在运行态决定的.“主动”是针对用户来说的,即相对于只能被动按照设定的固定路由进行组合的传统服务层叠网,主动服务层叠网可以支持用户的可编程,从而动态改变层叠网的拓扑结构,以支持按需的服务组合.如图 3 所示,通常情况下,组合服务请求者可以分别加入两个业务域进行在线购物或者信用卡交易,但当其试图实现信用卡购物需求却无法在一个域内完成时,则需要动态构建一个

大的虚拟业务域,以支持这种个性化的需求.由于主动服务(merchant 和 payer)可以在两个域中分别担任角色,因此可以根据用户提出的不同协议组合规则将两个业务域连成一个虚拟业务域,并确定自身的对外交互行为,从而实现跨域的业务需求.我们在主动服务的基础上,可以构建一个用户可编程的服务层叠网——主动服务层叠网,从而更加灵活地适应动态变化的业务需求.

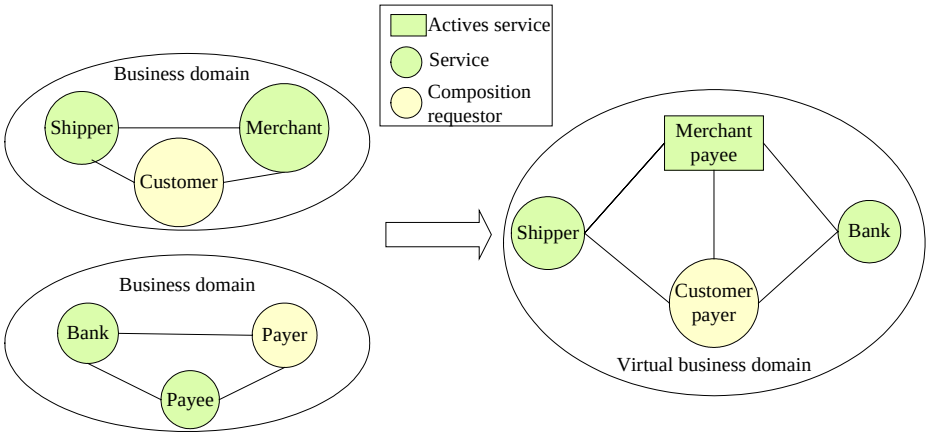


Fig.3 Active service overlay network

图 3 主动服务层叠网

2 基于 ASON 的服务组合

在建立 ASON 以后,关键的问题就是如何基于 ASON 进行服务组合,包括如何对需求进行描述,如何将需求描述映射为 SON 中的一组服务实例,并通过这些服务间的协作实现服务的组合.尤其是当跨多个业务域协作时,由于涉及到主动服务(服务功能并不确定),这些主动服务的可变性如何刻画,如何在可变功能的情况下进行服务选择,而不仅仅只是根据服务的 QoS 属性进行选择.本文采用协议的方式统一地刻画需求和服务组件,因此可以通过针对协议的各种运算来实现需求到服务组合实现的一致、平滑的映射.我们将这种基于协议的服务组合方法称为协议计算[6].本文以下将从分层服务组合框架、基于协议的需求描述以及服务组合映射机制 3 个方面详细阐述协议计算的思想.

2.1 分层服务组合框架

基于 ASON,本文提出分层服务组合框架 HOSS,它是框架为一个 3 层的结构,包含需求描述层、用户视图层(user perspective of SON,简称 PSON)和 ASON,如图 4 所示.其中,ASON 是 HOSS 框架的基础架构,是在现有服务基础上形成的可管理的应用层逻辑网络.在需求描述层,用户采用组合协议的形式描述功能需求,并添加 QoS 需求;PSON 位于需求描述层和 ASON 之间,负责将应用需求映射为 ASON 中的一组服务,从而实现服务的组合.

需求描述层:在一个软件系统中,计算和交互是正交的[7].在 SOC 环境下,可将服务视为负责计算的单元,而服务间的协作交互则独立于服务产生系统所期望的行为[8].我们将协议作为业务合作伙伴之间交互协作的原子单元,通过协议的组合刻画服务组合需求,并独立于具体的服务实例.与传统的基于流模式的服务组合方法不同,协议只强调业务交互的基本特征而不涉及实现细节(如服务调用参数等)[9],这里我们假定 ASON 中的协议都是业务领域内通用的标准协议,所以在需求的描述中不必对协议本身进行大量的刻画,可以直接通过协议的名称进行引用.另外,较为简单、通用的协议可以经过一系列的协议组合生成满足个性化需求的、复杂的服务组合过程,如图 2 所示,信用卡支付的购买协议可以由购买协议和信用卡支付协议组合而成.因此在 HOSS 框架中,我们基于协议声明式地描述组合需求.

PSON:由于 ASON 只是反映了服务间静态的业务域关系,没有根据实际需求给出业务域间的控制流关系,

因此不能满足复杂应用的跨域协作需求.为此,我们在 HOSS 框架的需求描述层和 ASON 之间构建了一个中间层——PSON.在 PSON 中,每个节点表示协议中的一个角色,角色间的链接表示控制流的顺序.当用户需要跨域协作时,动态地在需求描述层和底层 ASON 之间建立虚拟的 PSON,使得基于协议描述的需求可以平滑地映射为 ASON 中服务的组合.

ASON:ASON 为框架中的物理层,负责服务的管理、维护以及组合服务的执行等工作.

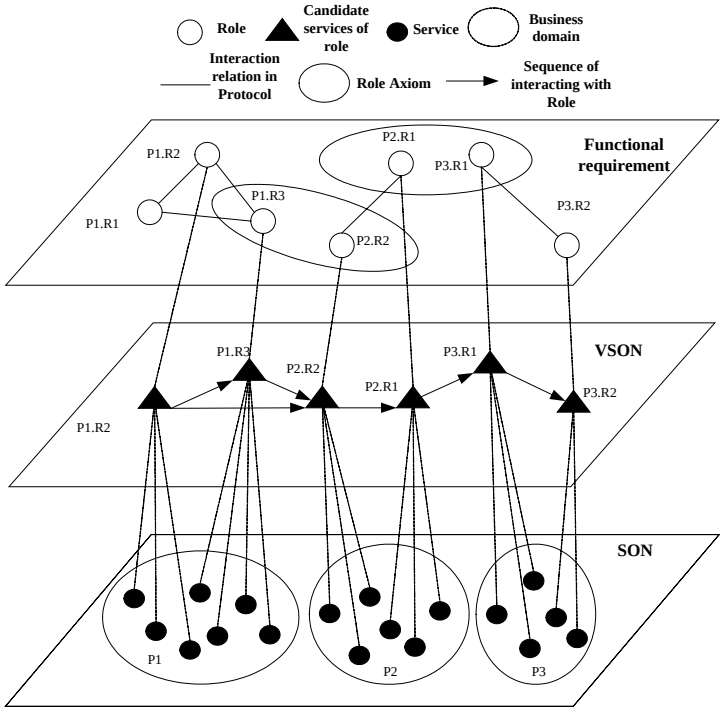


Fig.4 HOSS framework

图 4 HOSS 框架

为论述方便,我们形式化地定义如下基本概念.由于我们假定 ASON 中的协议都是业务领域内通用的标准协议,因此无须考虑具体的交互过程,需要时,可直接通过协议的名称进行引用,所以,我们主要根据协议所包含的角色和角色间是否存在直接交互关系定义协议.

定义 1(协议). 协议 p 是一个二元组 $p=(R_p,Intera)$,其中,R_p 是协议 p 中包含的一组角色,$Intera$ 是 R_p 上的一个谓词,表示对任意的角色 $p.r_1,p.r_2\in R_p$ 我们有 $Intera(p.r_1,p.r_2)=1$,当且仅当角色 $p.r_1$ 和 $p.r_2$ 间存在消息交换.

定义 2(服务). 服务可定义为一个二元组 $s=(PR_s,Q_s)$,其中,集合 PR_s 中的任意元素 $p.r$ 表示服务 s 遵守协议 p,并担任 p 中角色 r;$Q_s=\{Q_s(p.r,p.r')|p.r\in PR_s,p.r'\in R_p,Intera(p.r,p.r')=1$,是服务 s 的非功能属性的描述,其中,$Q_s(p.r,p.r')$表示当服务 s 担任角色 $p.r$ 并与担任角色 $p.r'$的服务进行交互时的 QoS 属性值元组.

集合 PR_s 刻画了服务 s 参与的所有协议及在协议中担任的角色,我们用 $s^{p.r}$ 表示此时服务 s 担任协议 p 中的角色 r.在不引起混淆的情况下,也称 $s^{p.r}$ 是服务,本文中会交替使用服务 s 与 $s^{p.r}$ 的概念.另外,服务的 QoS 属性与交互的角色直接相关,这是因为对于服务间具有多元交互的场景,一个服务的 QoS 权值会随着交互对象的改变而有所不同,而不是只存在单一的 QoS 权值.比如在图 2 中,Merchant 分别与 Customer,Shipper 和 Bank 交互时,应具有不同的价格参数.基于上述服务的描述,我们形式化地给出服务层叠网的定义.

定义 3(主动服务层叠网 ASON). 服务层叠网是一个无向图 $G=\langle V,E\rangle$.其中,V 是层叠网中的节点集,每个节点表示一个服务 $s^{p.r}$.E 是层叠网的边集,满足对任意的边 $e(s_1^{p.r_1},s_2^{p.r_2})\in E$,表示服务 s_1,s_2 分别担当协议 p 的不同角

色 r_1,r_2,且 r_1 与 r_2 之间有交互,我们假设一个服务在一个协议中只能担任一个角色.

从服务层叠网的定义可知,一个服务可以充当服务层叠网中的不同节点,表示该服务遵守不同的协议,可以担当不同协议中的角色.此时,该服务提供的功能是不同的.另外,担任同一协议的每个角色的服务可能有多个.我们称参与协议的 p 这些服务构成了一个协作相对紧密的业务域.

定义 4(业务域 business domain). 协议 p 的一个业务域 G_p 是服务层叠网的一个子网 $G_p=\langle V_p,E_p\rangle$,其中,$E_p\subseteq E$ 是 E 中所有与协议 p 相关的边构成的子集,即 $E_p=\{\ e(s_1^{p.r_1},s_2^{p.r_2})\mid \exists e(s_1^{p.r_1},s_2^{p.r_2})\in E\ \}$;$V_p\subseteq V$ 是 E_p 中所有边的顶点集合,即 ASON 中所有遵守协议 p 的节点子集.

2.2 基于协议的需求描述

HOSS 框架中,用户提出的服务组合需求包含两部分内容:功能需求和非功能需求.通常意义上的 QoS 需求是一个广泛的概念,包含价格、信誉度、执行时间、可靠性、有效性等非功能属性[10],记为 $Q_{require}$.本文不着重讨论 QoS 的需求描述,而重点讨论基于协议的功能需求描述.

协议的组合关系通常无法用顺序、选择、并发等结构化算子刻画.在 HOSS 框架中,我们借鉴 Desai 等人[9]提出的协议组合公理来描述协议的组合.我们将用户 u 提出的功能需求定义为一个三元组$\langle P,R,Axiom\rangle$,其中,P={}是一个协议集合,表示用户 u 参与的协议集合;R={$p_1.r_1,\ldots,p_n.r_n$}中的每一个元素 $p_i.r_i$ 表示用户 u 在协议 p_i 中担任角色 r_i;*Axiom* 是协议组合公理集合.协议组合公理集合包含角色公理集合 *RAxiom*、数据公理集合 *DAxiom* 以及消息顺序公理 *MAxiom*,其中:

- 角色公理集合 *Raxion*:任意角色公理 *raxiom*={ $p_{i_1}.r_{j_1},...,p_{i_l}.r_{j_l}$ }表示角色 $p_{i_1}.r_{j_1},...,p_{i_l}.r_{j_l}$ 必须要由一个服务担任,此时,称角色 $p_{i_k}.r_{j_k}$ $(k=1,...,l)$ 满足角色公理 *raxiom*.比如在图 2 中,Merchant 和 Payee 角色、Customer 和 Payer 角色都需要由同一个服务担当.
- 数据公理集合 *DAxiom*:任意数据公理 *daxion*=(p_1.*Message*1.*item*1,p_2.*Message*2.*item*2),表示将 p_1 协议 *Message*1 消息的数据项 *item*1 的数据转移到 p_2 协议 *Message*3 消息的数据项 *item*2 中,如 *DAxiom*(在线购物协议.购物请求.用户 ID,信用卡支付协议.信用卡信息.用户 ID).
- 消息顺序公理 *MAxiom*:任意消息顺序公理 *maxiom*= (first:p_1.*Message*1,after:p_2.*Message*2),表示 p_1 协议中的消息 *Message*1 在 p_2 协议中的 *Message*2 之前,如 *MAxiom*(first:在线购物协议.请求确认,after:信用卡支付协议.信用卡信息).表示请求确认后发送信用卡信息.

通过上述这种方式不仅可以声明式地描述需求,避免刻画具体的交互细节,而且可以灵活地根据需求改变实际的服务组合过程.如在图 2 中,通过改变两个协议组合后的消息顺序,可以灵活地构造不同的组合协议,即使用同样的在线购物协议和信用卡支付协议,采用不同的消息顺序公理 *MAxiom* 可以构造信用卡支付的预付费购物协议和信用卡支付的后付费购物协议这两种不同的协议.在 HOSS 框架中,我们使用协议组合公理来刻画主动服务的可变性.

2.3 服务组合映射机制

在协议计算方法中,组合需求是以全局方式刻画的,所以无法显性地从组合请求者(用户)的角度得出组合服务的 QoS.比如,在一个协议中可能存在多个不直接与用户 u 交互的角色,这些不直接交互的角色所对应的服务对用户来说并不影响组合服务的 QoS,因此,只需考虑与用户有直接交互的服务.同时,当需求描述中用户 u 参与多个协议时(即$|P|>1$),还需要用户在每个业务域中与之有直接交互的服务集合满足角色公理,而不仅仅是满足 QoS 需求.因此,为了高效地实现服务组合,我们借鉴关系数据库的机制,将服务视为实体、协议视为服务间的基本关系,给出了一组基于协议进行服务组合的计算算子:投影算子(projection operator)、连接算子(join operator)和选择算子(selection operator),其中,投影算子用于生成用户 u 在一个业务域中所有可以与之交互的服务节点集合;连接算子用于当用户 u 属于多个业务域时,根据由投影算子作用于每个业务域的结果生成满足角色公理的服务节点集合;选择算子对投影或连接算子的结果根据 QoS 需求生成服务组合实例.

2.3.1 投影算子

用户 $u^{p.r}$ 可在协议 p 的业务域中找到所有可以与之交互的服务节点集合,记为 $\pi_{u^{p.r}}(G_p)$,我们称 $\pi_{u^{p.r}}$ 为投射算子,即 $\pi_{u^{p.r}}(G_p)=\{t^{p.r'}\in V_p \mid \exists e(u^{p.r},t^{p.r'})\in E_p\}$,称为用户 u 关于角色 $p.r$ 的交互实例集合.

2.3.2 连接算子

假设用户 u 参与协议 $p_1,\ldots,p_n$,分别担任的角色为 $p_1.r_1,\ldots,p_n.r_n$,且 $p_1,\ldots,p_n$ 满足的角色公理集为 $RAxiom$.设 u 关于角色 $p_i.r_i$ 的交互实例集合为 $V_{u^{p_i.r_i}}$,即 $V_{u^{p_i.r_i}}=\pi_{u^{p_i.r_i}}(G(p_i))$.记用户 u 关于协议 $p_1,\ldots,p_n$ 按照角色公理集 $RAxiom$ 生成的组合协议的交互实例集合为 $(\rho_{Raxiom})_{i=1}^{n}(V_{u^{p_i.r_i}})=V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$,满足对 $V_{u^{p_i.r_i}}$ 中任意节点 $s^{p_i.r}$ $(1\le i\le n)$:

1) 若 $p_i.r$ 不满足任何角色公理,则 $s^{p_i.r}\in V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$;

2) 若 $p_i.r$ 满足角色公理 $raxiom(p_{i_1}.r_{j_1},\ldots,p_{i_k}.r_{j_k})$,即 $p_i.r\in\{p_{i_1}.r_{j_1},\ldots,p_{i_k}.r_{j_k}\}$,且 $s^{p_{i_t}.r_{j_t}}\in V_{p_{i_t}.r_{j_t}}$ $(t=1,\ldots,k)$,则有

$$s^{p_{i_1}.r_{j_1}},\ldots,s^{p_{i_k}.r_{i_k}}\in V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}.$$

显然,$V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ 包含 ASON 中所有与用户 u 有交互且满足功能需求的服务节点集合,称 ρ_{Raxiom} 为连接算子.

性质 1. 设 $V^{p_i.r_{i_j}}_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ 为 $V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ 中遵守角色 $p_i.r_{i_j}$ 的服务集合$(i=1,\ldots,n;j=1,\ldots,d_i)$,则 $V^{p_i.r_{i_j}}_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ $(i=1,\ldots,n;j=1,\ldots,d_i)$ 构成了 $V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ 的一个划分,即 $V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}=\bigcup_{i=1}^{n}\left(\bigcup_{j=1}^{d_i}V^{p_i.r_{i_j}}_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}\right)$,且 $V^{p_i.r_{i_j}}_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ $(i=1,\ldots,n;j=1,\ldots,d_i)$两两互不相交.

按照用户 u 提出的功能需求$\langle P,R,RAxiom\rangle$,在 ASON 中依据以上方法生成服务子集 $V^{p_i.r_{i_j}}_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ $(i=1,\ldots,n;j=1,\ldots,d_i)$,我们称之为与用户 u 有交互的每个角色的候选服务集合.

2.3.3 选择算子

设满足功能需求的服务集合为 $V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$,记 $\sigma_{Qrequire}\left(V_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}\right)\in\prod_{i=1}^{n}\prod_{j=1}^{d_i}V^{p_i.r_{i_j}}_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ 为满足用户 QoS 需求 $Q_{require}$ 的服务实例集合,称 $\sigma_{Qrequire}$ 为选择算子,其中,$\prod_{i=1}^{n}\prod_{j=1}^{d_i}V^{p_i.r_{i_j}}_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ 为 $V^{p_i.r_{i_j}}_{\langle u^{p_1.r_1},\ldots,u^{p_n.r_n},RAxiom\rangle}$ $(i=1,\ldots,n;\ j=1,\ldots,d_i)$的笛卡尔积.

本文的前序工作[11]详细介绍了PSON层的构造以及基于PSON在服务层叠网中进行QoS感知的服务选择,因此,本文只是简单地阐述选择算子实现的基本原理.我们在投影和连接操作完成后,可以在需求描述层和ASON之间建立起一个符合应用需求的用户视图PSON,其中的每一个角色都会有多个候选服务组件,我们据此构造一个虚拟的服务组合图.在该图中,每个节点表示一个服务组件,服务组件间链接的权值表示以该链接的目的服务组件的 QoS 值.构造出该服务组合图后,就可以使用最短路径算法得出最优的服务选择.

3 框架实现

3.1 框架结构

我们将 HOSS 框架的实现分为应用层、ASON 管理层和 ASON 层.其中,应用层为面向最终用户和服务提供者的一组工具集;ASON 管理层包含为服务层叠网维护、管理以及进行服务组合所需的一系列功能模块;ASON 层主要包含各种服务资源以及维护这些服务资源,并支持这些服务对外交互所必须的功能模块. ASON 层和 ASON 管理层内部以及层间的消息交互统一通过服务总线进行传输.如图 5 所示.

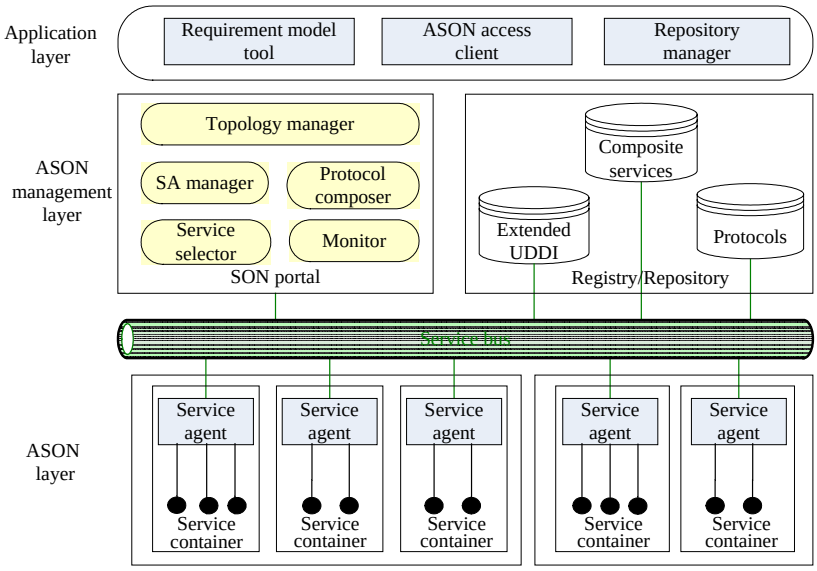


Fig.5 Architecture of HOSS

图 5 HOSS 框架结构图

3.1.1 ASON 层

ASON 层是 ASON 中相对静态和稳定的部分,包含各种服务资源,这些服务资源都由服务容器进行维护和管理.每个服务容器中部署一个服务代理,并且可以包含多个服务组件,使用 Apache Tomcat 以及 Axis 来实现.ASON 层中最重要的功能组件是服务代理,它作为 Web 服务的唯一外部访问点,与 Web 服务一起实现了主动服务的功能,从而使得现有的只能被动调用的 Web 服务转变为可以根据用户需求动态改变自身功能和行为的实体.服务代理的主要功能包含为:

- 基于协议统一服务的接口.即使遵循同样的协议,由不同组织开发的服务也可能有不同的接口,这使得动态的服务组合实现起来非常困难.为了解决这个问题,服务代理将每个用户的接口按照标准协议映射为一个通用的接口,从而将协议与具体服务实现了分离,屏蔽了服务的异构性.
- 动态重配置:服务代理可以在运行时根据用户的配置动态改变与之交互的特定的服务实例以及协议组合公理.基于动态重配置的功能,多个业务域可以按需地组成一个更大的虚拟子网,从而灵活地支持多变的业务需求.

3.1.2 ASON 管理层

ASON 管理层是实现可编程服务层叠网络的关键部分,主要包含 Portal 和基础库两大类功能部件.其中,基础库由服务库、组合服务库和协议库构成.服务库是在 UDDI 基础上扩展而成,支持 UDDI Java API(UDDI4J),主要增加了服务的 QoS 属性以及其所支持的交互协议的内容.协议库是一个协议注册中心,主要存放服务层叠网中所涉及的各种基本的协议,拥有标准的接口并允许协议的发布和查找.组合服务库则保存已经实现的组合服务,当用户的需求可以通过现有组合服务实现时,可以直接使用,而无须重新进行协议组合、服务选择、服务动态重配置等一系列工作,从而提高了组合服务的执行效率.

Portal 作为一个全局的功能主体,接收来自用户的组合请求,然后选择正确的服务并配置相应的服务代理.Portal 的主要功能包括:

- 接收用户的请求:为了保证组合服务的有效性,我们构建了一个具有确定的可管理边界的 ASON.在 ASON 中,每个期望参与服务组合的服务组件必须通过 SLA 承诺自身的 QoS.Portal 是 ASON 中唯一的外部访问点,也就是说,用户需要将需求提交给 Portal 以触发服务的组合过程.
- 协议组合:根据用户的需求,将多个基本的协议组合形成一个全局的组合协议,并生成用户端的对外交

互过程,发送给用户.

- 服务的选择:服务的描述包括相关协议的 QoS 信息,并且在 Portal 中进行注册.因此,当接收到用户的需求时,Portal 进行优化计算以决定选择的服务实例,从而最大限度地满足用户的需求.
- 服务代理管理:在服务组合执行前,需要一个初始化过程.在这个过程中,Portal 负责将选择的服务实例通知参与服务组合的各个服务代理.另外,协议组合公理也被上载到相应的服务代理.
- 监控器:负责服务层叠网内组合服务执行情况的监控,当有异常情况出现时,及时通知用户,并将出错信息发送给层叠网维护模块进行处理.
- 层叠网维护:主要负责基础库的维护.它主要根据监控器所记录的组合服务执行的历史信息,更新服务的 QoS 信息以及最有效地组合服务实例.另外当基本协议发生变化时,负责协议的一致性维护等工作.

3.1.3 应用层

应用层包含一组工具集,能够使服务组合开发者和最终用户基于 ASON 进行服务和协议的发布、组合服务的创建和执行、组合服务需求的建模等工作.这组工具都能够使用 SOAP 消息和 ASON 管理层进行通信.服务组合需求建模工具提供基于协议的组合服务建模,并提供访问协议库的功能.层叠网访问客户端是一个简化的 BPEL 执行引擎,在需求提交给 Portal 后,接收 Portal 发回的最终用户所需的对外交互过程,并自动进行配置.服务和协议注册工具则为服务提供者给出了一个注册、发布可重用服务和协议的可视化工具 HOSS.

3.2 服务组合过程实例

图 6 展示了在 HOSS 框架中,采用图 2 中应用实例进行服务组合时系统中的消息交互过程.图中的箭头表示不同类型的消息流(系统控制流、在线购买和信用卡支付协议),数字表示消息的时序关系.为论述简单起见,用双向的箭头表示一组消息.另外,假定服务及其对应的服务代理位于同样的节点上,从而忽视服务代理与服务之间的消息交互.

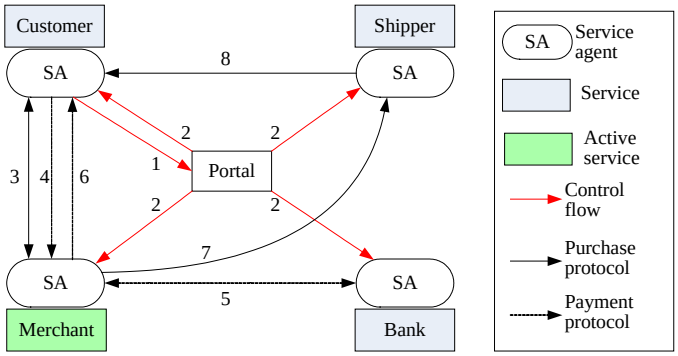


Fig.6 Service composition process in ASON

图 6 主动服务层叠网中的服务组合过程

(1) 用户想要实现服务组合,首先将需求提交给 HOSS 中的唯一访问点——Portal.这个需求将包含参与的协议以及这些协议的组合公理、它在组合协议中扮演的角色以及相应的 QoS 需求.

(2) 当 Portal 收到来自用户的请求时,首先根据角色公理和 QoS 信息进行服务的选择;然后将组合协议按照用户的角色分解成用户的对外交互过程,并将该交互过程发送给用户;最后,通知参与本次服务组合的相关服务实例,并上载相应的数据公理集合 *DAxiom* 以及消息顺序公理 *Maxiom*.

(3) 在本次服务组合实例都已配置完成之后,*Customer* 和 *Merchant* 服务首先通过各自的服务代理进行交互.在这个过程中,将采用在线购买协议,包含多个交互消息,*query*(*Goodsname*),*itemList*(*itemIDlist*),*reqQuote*(*itemID*),*Quote*(*itemID*,*itemPrice*)以及 *Order*(*itemID*,*Token*).

(4) 根据消息顺序公理 *MAxiom*,*Customer* 在收到 *Merchant* 确认(或者收到货物)后,采用信用卡支付协议发送 *creditcardInfo*(*cardID*)消息给 *merchant*,以提供信用卡信息.

(5) *Merchant* 服务发出 *Auth*(*cardID*,*amount*)消息以触发与 *Bank* 服务间的交互,并接收 *Authconfirm*(*Token*, *amount*)消息.

(6) *Merchant* 服务发送 *Receipt*(*cardID*,*amount*,*Token*),以响应 *customer* 的 *creditcardInfo*(*cardID*)消息.

(7) *Merchant* 服务发送 *customerInfo*(*Token*,*address*)消息,以通知 *shipper* 发送货物给 *customer*.

(8) 当 *Customer* 从 *shipper* 服务收到(*token*,*address*)消息时,整个服务组合过程完成.

4 分析与评估

(1) 网络拓扑维护开销

在 SON 中,通常需要预先建立网络的拓扑,从而有效地组织服务间的关系.对于传统的 SON 中来说,假设网络中提供的不同的功能个数为 F,每个功能的候选服务为 N,则为了最大限度地保证根据需求提供新的增值服务的能力以及冗余备份的需要,通常情况下,网络中边的数目=$N\times F\times(F-1)\times N\div 2$.因此,当网络中服务的数量和提供的功能数量增加时,维护整个网络拓扑的开销非常大,尤其是当网络中服务的加入和退出较为频繁时,网络拓扑维护的巨大开销会严重影响系统的性能.在 HOSS 中,一方面由于按照协议将整个网络划分成多个业务域,每个域控制器只负责维护本业务域内的网络拓扑;另一方面,业务域内服务间的关系是基于协议确定的,不用维护非直接相关的服务之间的链接关系,因此,网络维护的开销远小于传统的 SON 网络.

(2) HOSS 中主要实体的运行负载

在 HOSS 框架中,Portal 和服务代理是实现服务组合的最重要的功能实体,因此,它们的运行负载情况直接影响到整个系统的性能.为此,本实验的目的是测量运行过程中这两类实体的工作负载,并与集中式的服务组合方式以及同为分布式执行的 Self-Serv[12]方法进行对比.

为了进行实验对比,我们首先开发了 HOSS 框架的一个原型系统,然后将其部署在一个 PC 集群上.这个集群中的每一台 PC 都采用相同的配置:Pentium IV 2.4Ghz,1GB RAM,并且使用百兆以太网相连.我们根据图 2 中预付款协议的例子,在其中的一台机器部署 Portal,在其他机器上分别部署 Customer,Merchant,Shipper 和 Bank 的服务代理.这里,我们不考虑服务代理和服务间的消息交互.

在交互消息的大小一定,即应用实例相同的情况下,HOSS 框架中实体的运行负载主要体现为对外交互消息的数量以及各实体功能模块的响应时间.我们首先测量在实现图 2 预付款协议所描述的服务组合过程中,不同实现方式下的各实体的对外交互消息数量.由图 7 可知,总体上,HOSS 和 Self-serv 方法下,各服务代理的对外交互数量要略多于集中式方式,但是,Portal 的对外交互数量远远小于集中式方式.这种差别在并发组合服务请求较多的情况下尤其明显.因此,HOSS 和 Self-Serv 方法比集中式的方法具有更好的可扩展性.

第 2 个实验测试实体各功能模块执行时间方面 HOSS 和 Self-serv 方法的差别.对于 Portal,虽然服务选择等模块也会产生一定的系统负载,但不是本文关注的重点,因此考虑主要组合初始化阶段在服务代理重配置方面的代价.服务代理重配置过程包括将组合协议分解成各个服务代理对应的对外交互过程以及上载该交互过程到相应服务代理.对于服务代理,我们主要测试在一次完整的服务组合过程中,对每一个输入消息的响应时间的总和.我们仍然使用图 2 预付款协议的例子,并重复 10 次后取平均数.由图 7 可以看到,虽然 HOSS 和 Self-serv 这两种不同的分布式服务组合方法下各实体的对外交互数量并没有什么不同,但在功能模块执行时间方面,两者具有较大的差别.这是因为,在 Portal 自身的功能处理方面,Self-serv 需要为每一个参与的服务节点生成包含前置条件和后验条件的路由表,对 Portal 的负载很大;而在 HOSS 中,由于服务遵守的协议已经隐含了此类内容,因此只需为服务请求者生成相应的对外交互过程即可.另外,上载路由表也是一项时间耗费较多的工作,在 HOSS 方法中,Portal 不用为每个服务代理上载路由表,只需通知服务代理相应的交互实例,因而有效地降低了系统的负载.在服务代理方面,由于 HOSS 中服务代理除了需要进行对外交互外,还要负责解释协议公理的工作,因此性能上要略差于 Self-serv 方法.总体来看,由于作为全局实体的 Portal 是整个系统的瓶颈,因此在大规模的服务组合应用场景下,HOSS 方法更具可扩展性.

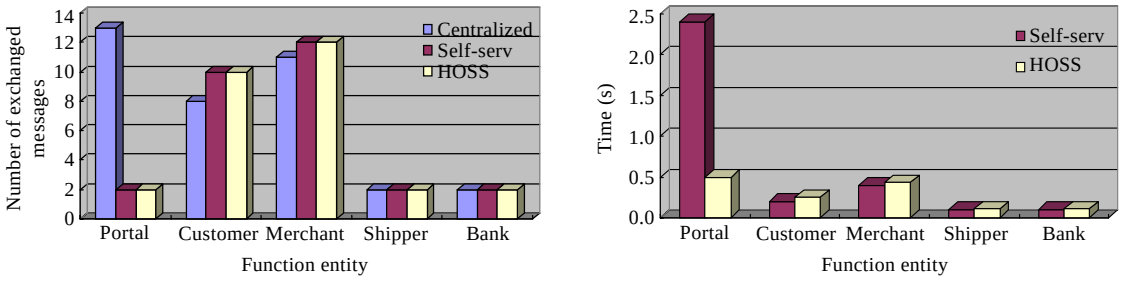


Fig.7 Workload allocation in the execution

图 7 运行负载实验结果示意图

5 相关工作比较

目前,基于层叠网进行 QoS 感知的服务组合研究主要集中在多媒体领域.Illinois State 大学的 Gu[2]提出了 service overlay networks(SON)模型.在该模型下,提出了一个 QoS 保证的可组合服务基础设施 QUEST,可以把用户的组合服务请求映射为 SON 中的服务间路由,并且能够同时对多个 QoS 约束提供端到端的 QoS 保证和负载均衡.Toronto 大学的 Wang[3]提出了一个 SON 网络.该 Overlay 网络分为 3 层:最下面一层是物理网络,中间一层是 overlay graph,最上层是功能需求.中间层的 overlay graph 反映的就是一个 SON 网络,通过服务间的兼容关系,即一个服务的输出匹配另一个服务的输入需求,判断两个服务间是否可以建立链接关系.然后研究节点具有不完全信息的情况下如何将功能需求映射为 overlay graph 中的路由.

与上述层叠网方面的研究相比,本文的特点是:1) 上述研究将服务间的交互看作是简单输入/输出关系,没有考虑电子商务等应用环境中服务间复杂的交互关系,而在本文的 HOSS 框架中,通过引入协议作为功能需求和服务间复杂交互关系刻画的共性基本单元,可以适用于更广泛的应用场景;2) 上述研究的基础是需要预先建立一个能够满足所有需求的 Overlay 网,而本文基于协议构造功能相对独立的业务域,通过这些业务域形成一个松散耦合的 SON,并提供用户可编程功能,实现业务域间的按需协作,可扩展性和灵活性较好.

以协议作为基本计算单元研究服务间的交互可以有效提高服务组合的灵活性.这方面的研究包括: North Carolina State 大学的 Singh[9]等人提出了基于业务协议的业务过程设计与制定方法,协议都公布在一个协议库中,通过协议的精化和组合来实现复杂的业务过程;文献[13]提出通用协议的思想,即给出一个通用的、适合于任何情况的协议,然后根据不同的情况配置成符合特定需求的协议; New South Wales 大学的 Benatallah 等人[14]给出了一组协议代数来分析和管理 Web 服务协议.上述研究初步体现了可以通过协议的复用来灵活地构造复杂业务过程,但是都没有基于协议将需求和服务进行统一的描述和刻画,因此也就很难实现需求到服务组合的平滑和一致性的映射.

当前,大多数的服务组合执行引擎为集中式的,如 eFlow[5]等,但是也有一些分布式服务组合方面的研究.其中,CPM[15],Self-serv[16]和 SpiderNet[2]是其中较为典型的分布式系统.但是,这些研究都是将服务视为一个功能和行为确定的功能实体,在服务组合完成后,组合服务的功能很难根据需求的变化在运行时动态地进行调整.本文提出的 ASON 方法可以支持用户的可编程,从而使得整个架构对需求的动态变化更具适应性.

6 结 论

我们提出了一个基于 SON 的分层服务组合框架——HOSS,用以实现互联网环境下灵活、高效的服务组合.通过引入协议刻画服务间复杂的交互关系,能够构造一个具有更广泛适用场景的 SON;提出了 ASON,通过构建可编程的服务层叠网以实现按需的服务组合;通过基于协议的方式描述服务组合需求,使得需求描述不用涉及具体服务实例的细节,提高了需求描述的抽象层次;引入 PSON 以实现需求到 ASON 的映射,并通过协议的组合实现业务域间的按需协作,提高了原有方法的灵活性和可扩展性.

我们进一步的工作包括总结、归纳出完整的协议代数模型,以准确、全面地刻画组合服务的动态行为与

QoS 能力,从而支持高效的动态组合开发及其评估;深入地研究主动式和可信的服务层叠网以更好地满足跨边界服务组合的需要.

References:

[1] Xiao XP, Ni LM. Internet QoS: A big picture. IEEE Network, 1999,13(2):8−18.

[2] Gu XH, Nahrstedt K. Distributed multimedia service composition with statistical QoS assurances. IEEE Trans. on Multimedia, 2006,8(1):115−124.

[3] Wang M, Li BC, Li ZP. sFlow: Towards resource-efficient and agile service federation in service overlay networks. In: Proc. of the 24th Int'l Conf. on Distributed Computing Systems (ICDCS). Washington: IEEE Computer Society, 2004. 628−635.

[4] Singh MP, Chopra AK, Desai N, Mallya AU. Protocols for processes: Programming in the large for open systems. In: Proc. of the Conf. on Object-Oriented Programming Systems, Languages, and Applications. New York: ACM Press, 2004. 120−123.

[5] Casati F, Ilnicki S, Jin LJ. Adaptive and dynamic service composition in eFlow. HP Labs Technical Report, HPL-200039, Palo Alto: Software Technology Laboratory, 2000. 13−31. http://www.hpl.hp.com/techreports/2000/HPL-2000-39.pdf

[6] Huai JP. Web service and service-oriented protocol computing. Communication of China Computer Federation, 2005,1(4):31−40 (in Chinese with English abstract).

[7] Gelernter D, Carriero N. Coordination languages and their significance. Communications of the ACM, 1992,35(2):96.

[8] Buhler P, Vidal JM, Verhagen H. Adaptive workflow=Web services+agents. In: Proc. of the Int'l Conf. on Web Services Las Vegas: CSREA Press, 2003. 131−137.

[9] Desai N, Mallya AU, Chopra AK, Singh MP. Interaction protocols as design abstractions for business processes. IEEE Trans. on Software Engineering, 2005,31(12):1015−1027.

[10] O'Sullivan J, Edmond D, Ter Hofstede A. What's in a service? Towards accurate description of non-functional service properties. In: Proc. of the Distributed and Parallel Databases. London: Springer-Verlag, 2002. 117−133.

[11] Li Y, H. J, Deng T, Sun HL, Guo HP, Du ZX. QoS-Aware service composition in service overlay networks. In: Proc. of the IEEE Int'l Conf. on Web Services (ICWS). Washington: IEEE Computer Society, 2007. 703−710.

[12] Benatallah B, Dumas M, Sheng QZ. Facilitating the rapid development and scalable orchestration of composite Web services. Distributed and Parallel Databases, 2005,17(1):5−37.

[13] Bartolini C, Preist C, Jennings NR. Architecting for reuse: A software framework for automated negotiation. In: Proc. of the 3rd Int'l Workshop on Agent-Oriented Software Engineering. London: Springer-Verlag, 2002. 87−98.

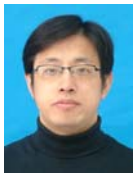
[14] Benatallah B, Casati F, Toumani F. Representing, analysing and managing Web service protocols. Data & Knowledge Engineering, 2005,58(3):327−357.

[15] Chen Q, Hsu M. Inter-Enterprise collaborative business process management. In: Proc. of the 17th Int'l Conf. on Data Engineering (ICDE). Washington: IEEE Computer Society, 2001. 253−260.

[16] Benatallah B, Dumas M, Sheng QZ, Ngu AHH. Declarative composition and peer-to-peer provisioning of dynamic Web services. In: Proc. of the 18th Int'l Conf. on Data Engineering (ICDE). San Jose: IEEE Computer Society, 2002. 297−308.

附中文参考文献:

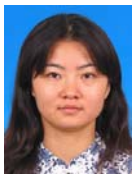
[6] 怀进鹏.Web 服务技术与面向服务的协议计算.中国计算机学会通讯,2005,1(4):31−40.



李扬(1976－),男,山东济南人,博士生,主要研究领域为服务计算,软件设计与生产.

郭慧鹏(1975－),男,博士生,主要研究领域为可信软件设计与生产,面向服务的计算.

怀进鹏(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算机软件与理论,网络安全,网格计算.



杜宗霞(1975－),女,博士,讲师,主要研究领域为面向服务的体系结构,软件设计与生产.