2008年 11月 第 11期
Vol. 29 No. 11 2008

分布式 UDDI互操作模型的研究

吴黎兵[1],崔建群[2],吴产乐[1],杨先娣[1]

[1](武汉大学 计算机学院,湖北 武汉 430072)
[2](华中师范大学 计算机科学系,湖北 武汉 430079)
E-mail: wu@whu.edu.cn

摘 要: 统一描述、发现与集成 UDDI是 Web服务中用来进行服务发布及查询操作的关键技术.目前的 UDDI大多采用集中模式,随着服务数量的不断增加,其查找效率不断降低且对 UDDI服务器的维护和管理也变得十分困难.此外,由于私有 UDDI服务注册库的出现,且私有注册库之间缺乏互通机制,导致出现了新的服务孤岛.针对上述现状,本文提出了一种分布式 UDDI互操作模型,将 UDDI模型分为根 UDDI、超域 UDDI和普通 UDDI三层架构,并对三层模型的维护和应用程序接口进行了较详细的讨论.仿真实验证明该模型可有效提高 UDDI的查询效率并实现了 UDDI注册库之间的信息互通.

关 键 词: UDDI; Web服务;超域 UDDI;分布式互操作模型
中图分类号: TP393 文献标识码: A 文 章 编 号: 1000-1220(2008)11-1990-05

Research on a Distributed Interoperable Model of UDDI

WU Li-bing[1], CUI Jian-qun[2], WU Chan-le[1], YANG Xian-di[1]
[1](*School of Computer, Wuhan University, Wuhan* 430072, *China*)
[2](*Department of Computer Science, Huazhong Normal University, Wuhan* 430079, *China*)

Abstract UDDI (Universal Description, Discovery and Integration) acts a very important role in the web service. It can be used to publish and lookup services. Most of the current UDDI models are centralized so that the performance will decrease if there are too many services to be registered or queried. It is hard to maintain and manage the UDDI servers too. Moreover, the emergence of private UDDI servers and bad interoperability of these servers lead to some new isolated service islands. In this paper, a distributed Interoperable Model of UDDI is proposed. The model divides whole UDDI servers into there types: root server, super domain server and normal server. The simulation results show that the model can get better lookup performance and implement the interpretability of UDDI servers.

Key words UDDI; web service; super domain UDDI; distributed interoperable model

1 前 言

统一描述、发现与集成 UDDI(Universal Description, Discovery and Integration)[1]是 Web Service中用来进行服务发布及查询操作的关键技术[2],它对 Web服务进行统一描述、发现与集成,将 UDDI和 WSDL(Web Services Description Language)[3,4]技术相结合可以实现 Web服务的查找操作.UDDI的主要目标是实现全球统一的服务目录 UBR(Universal Business Registry),以维护公共可见的服务信息.没有 UDDI,Web服务提供者就不能发布 Web服务而用户也无法查找到所需的服务,导致无法对 Web服务进行远程调用.可见 UDDI性能的好坏直接影响到 Web服务的服务质量,因此对 UDDI的研究成为 Web服务的研究热点.

UDDI规范的第一个版本发布于 2000年 9月,最新版本则是 2004年 10月推出的 UDDI v3.0.2[5].UDDI一直希望依靠建立一个标准的注册中心(Registry)来加速网络环境下的电子交易市场氛围下的企业级应用系统的集成.但是在 UDDI新版本中承认全球统一的服务注册库并不能解决 SOA(Service-Oriented Architecture)结构的所有问题[6],因此提出了附属注册中心的概念,用于指代目前广泛存在的私有或半私有服务注册中心.但其附属注册中心的服务信息是对 UBR信息的部分复制,而 UBR仍然记录所有的公共信息,附属注册中心主要作为信息的本地缓存存在,其集中的模式并没有改变.

目前,UBR由 IBM、Microsoft、SAP、NTT四家公司维护,各节点通过相互复制信息来实现数据同步.由于业务服务的多样性、复杂性以及安全性等原因,Web服务数量的增加导致 UBR的维护变得不可操作.事实也表明,目前使用较广泛的不是 UBR,而是遵从 UDDI规范的私有服务注册库.但由于各私有服务注册库间没有相应的互通机制,导致新的服务孤岛的形成.

综合以上分析,目前 UDDI的主要问题包括以下几

收稿日期: 2007-06-26 基金项目:国家"八六三"引导项目(2003AA001032)资助;国家自然科学基金项目(60672051)资助. 作者简介:吴黎兵,男,1972年生,副教授,博士,研究方向为网格计算;崔建群,女,1974年生,讲师,博士研究生,研究方向为高性能计算,应用层组播;吴产乐,男,1945年生,教授,博士生导师,研究方向为分布式计算;杨先娣,女,1974年生,博士研究生,研究方向为分布式信息集成.

点[7,8]:

(1) UDDI采用集中模式,随着服务数量的不断增加,其维护、管理将变得很困难;

(2) UBR的目标是要存储所有的服务信息,这些不区分行业、类别的信息混杂在一起,导致服务查找的效率较低;

(3) 目前广泛使用的私有 UDDI之间缺乏互通机制,导致出现了新的服务孤岛[9,10];

(4) 当前 UDDI是被动的服务目录,如何主动侦测服务的变动,如何及时标记已失效的 Web服务.

针对以上问题,本文提出了一种分布式 UDDI互操作模型,以提高 UDDI的查询效率、可用性以及提供 UDDI注册库之间的信息互通.

2 分布式 UDDI互操作模型

我们所提出的分布式 UDDI互操作模型分为三层: 根 UDDI超域 UDDI和普通 UDDI.

首先在全球设立类似于根 DNS的根 UDDI注册中心群,可由一台 UDDI Server担任,也可以由多台 UDDI Server担任,但一般为提高可靠性,至少由两台 UDDI Server构成.根 UDDI Server之间实行数据实时同步,以便将收敛时间缩短至最小.根 UDDI Server不提供 Web服务的发布和查询功能,以减轻根 UDDI Server的负荷.根 UDDI Server主要记录超域 UDDI Server相关信息,并为新的 UDDI注册中心提供注册和查询超域 UDDI Server的功能.

超域 UDDI Server负责管理在其附近的普通 UDDI注册中心,并为这些普通 UDDI注册中心提供注册和查询服务.另外,超域 UDDI Server还可从根 UDDI Server处获得其它超域 UDDI Server的相关信息,从而能查询更多的服务.

普通 UDDI Server则向用户提供 Web服务的发布和查询功能,在无法查找到合适服务时,它将向超域 UDDI Server发出查询请求.

分布式 UDDI互操作模型如图 1所示.

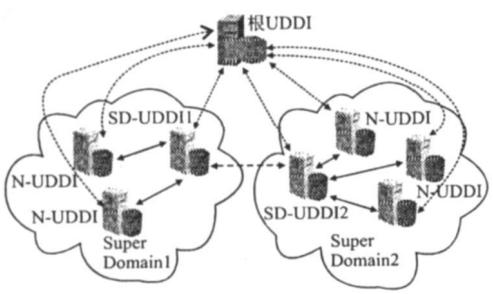

图 1 分布式 UDDI互操作模型

Fig. 1 Distributed UDDI interoperable model

图 1中的 SD-UDDI(Super Domain UDDI)代表超域 UDDI Server,N-UDDI代表普通 UDDI Server.

当企业或科研机构创建了新的 UDDI注册中心(普通 UDDI Server),首先向根 UDDI Server发出注册请求,根 UDDI Server接受请求后,将发送现有的超域 UDDI Server地址表给新的 UDDI注册中心.新的 UDDI注册中心计算到各个 SD_UDDI Server的路径代价,然后向最短路径的 SD_UDDI Server发出邻接(*Adjacency*)请求. SD_UDDI Server收到邻接请求后,向申请的 UDDI注册中心发出 ACK报文通知对方同意建立邻接关系,并同时向根 UDDI Server发送 Inform报文,以便让根 UDDI Server更新该超域所包括的 UDDI Server列表.

3 分布式 UDDI维护机制

3.1 加入 UDDI超域

当新的 UDDI注册中心从根 UDDI Server得到 SD_UDDI Server地址列表后,向每个 SD_UDDI Server发送 100个 ICMP报文,以探测 RTT(Round-Trip Time)往返时延和经过的路由器数目,通过公式(1)计算得出到达每个 SD_UDDI Server的 *average_metric*,并将 *average_metric* 值小于代价阈值 Metric_MAX的项按从小到大排序存入数组 *SD_metric*[]中.新的 UDDI注册中心首先向代价最小的 SD_UDDI Server发送 *Adjacency* 请求,如果请求被接受,则返回成功信息指示已建立邻接关系,并记录 SD_UDDI Server的 IP地址.如果请求超时或不被接受,则向 *SD_metric*[]中的下一个 SD-UDDI Server发送请求.如果与 *SD_metric*[]数组中的所有 UDDI Server都无法建立邻接关系,新的 UDDI注册中心则向根 UDDI Server发送 *Inform* 报文,通告自己新建立一个超域,并担任此超域的 SD_UDDI Server.根 UDDI Server收到 *Inform* 报文后,将该 UDDI Server IP地址添加到 SD_UDDI Server列表,并向此 UDDI Server发出确认通告信息.

$$average\_metric=\frac{\sum_{i=1}^{n}(rtt_i+\cup\times router_i)}{n} \quad (1)$$

公式(1)中的 n 为发出的 ICMP数据包总数,$rtti$ 为第 i 次 ICMP报文的往返时延,$routeri$ 为第 i 次 ICMP报文的经过的路由器数,权重 $\cup$ 是一个常数,其取值在 0.5到 0.9之间.

新的 UDDI注册中心选择加入超域,并与 SD_UDDI Server建立邻接的算法 *uddi_Select_SuperDomain* 伪代码描述如下.

```
Algorithm uddi_Select_SuperDomain(node, root[])
1 Metric_MAX= constant1;
2  For(i= 0; i <ubound(root[]); i++)
3    Udp_send_hello(root[i]);    //send registration request to root
4    msg= receive_msg();              //node receive the response
5    if(msg is valid)
6      break;
7  Endfor
8  If(msg is null)
9    return "error information";
10 else
11   SD_uddi[]= get_SD_UDDI_Server_list(msg);
12 Endif
```

```
13 for(j= 0;j < ubound(SD_uddi[]);j+ + )
14   send 100 ICMP packets to SD_uddi[j] and calculate average metric;
15   SD_uddi[j]= average_metric;
16 Endfor
17 sort the SD_uddi[] < Metric_MAX to SD_metric[] by ascending;
18 for(k= 0;k < ubound(SD_metric[]);k+ + )
19   TCP_send_Adjacency(SD_uddi[k]);
20   msg= receive_msg();
21   if(msg is "Accept")
22     return "successful";
23 Endfor
24 if(k= = ubound(SD_metric[])
25   send inform to root uddi server "I am a new super domain uddi server!";
```

3.2 超域内部 UDDI之间数据互通

分布式 UDDI模型中把加入到超域中的非 SD-UDDI Server称为 N-UDDI(普通 UDDI注册中心).一个超域中的 N-UDDI Server只与本超域 SD-UDDI Server进行数据交流,这样可以大大减少以往 UDDI之间互相复制数据所造成的较大网络负荷.通过在超域中设置 SD-UDDI Server,使得一个超域内部的 UDDI之间的数据交换次数由 $O(n^* n)$次减少为 $O(n)$次.

3.3 服务注册与查询

如果还没有找到合适的 Web服务资源,则由本超域 SD-UDDI向其它超域 SD-UDDI同时发出查询请求,当其它 SD-UDDI的响应信息返回到本超域 SD-UDDI后,SD-UDDI对响应数据进行综合并返回给本地 N-UDDI,再由本地 N-UDDI将查询结果返回给用户.

SD-UDDI缓存 30分钟,本地 N-UDDI缓存 10分钟.

3.4 UDDI异常处理

UDDI Server出现异常时会影响 Web服务的注册和查询,特别是当一个超域中的 SD-UDDI Server出现故障时,会严重影响超域内 UDDI之间的信息交互.因此当根 UDDI Server在一定时间内没有感知到某个 SD-UDDI Server时,会认为该 SD-UDDI可能失效.此时根 UDDI会立即通知该超域中的所有其它 UDDI重新选举 SD-UDDI.超域中的 UDDI之间通过 ICMP数据包互相探测到对方的 average-metric,然后通过公式(2)计算出每个 UDDI到超域内其它 UDDI的综合代价 total-metric,综合代价最小的 UDDI Server将被选择为超域新的 SD-UDDI.

$$total\_metric=\sum_{i=1}^{k} average\_metric_i \quad (2)$$

(其中 k为超域中 UDDI Server个数-1)

N-UDDI Server每隔 180秒向 SD-UDDI Server发送 KeepAlive报文,以便让 SD-UDDI Server感知自己的存在.如果 N-UDDI有更新数据发送给 SD-UDDI,则在发送更新报文时,将 *KeepAlive*定时器清零,换言之,更新报文一方面通知 SD-UDDI注册项的变化,另一方面可替代 *KeepAlive* 报文.如果 SD-UDDI在 6个定时通告周期内没有收到某个 N-UDDI的 *KeepAlive*报文,则将此 N-UDDI标记为暂时失效,并通知超域内其它的 N-UDDI,将从暂时失效 N-UDDI处学习到的 Web服务标记为"不可信".如果 SD-UDDI在三天内依然没有收到 *KeepAlive*报文,则将该 N-UDDI从超域中删除,并通过 Inform报文向根 UDDI报告.

3.5 注册服务的验证

在 UDDI Server中通过 *certifier*进程验证服务提供者宣称的 Web服务及其服务质量是否属实.*certifier*一般选择在轻载时间段通过 SOAP远程调用服务提供者发布的 Web服务,如果失败则将该 Web服务标记为失效,在多次验证失败后,该 Web服务将会从 UDDI注册中心删除.

当一个 N-UDDI失效后,SD-UDDI将从该 N-UDDI学习到的 Web服务全部标记为"不可信",并启动 *certifier*进程对这些 Web服务验证后,重新标记为"可用"或"失效",并将验证结果通告给超域中其它 N-UDDI.

3.6 Web服务分类

在 UDDI分布式注册中心的设计中,另一个重要的目标就是使用分类法以及实现对服务信息自动进行分类的能力.如果不使用分类类别或不具有鉴别 Web服务类别的能力,那么在跨超域 UDDI注册中心中搜寻定位 Web服务将比较浪费时间,这里的类别信息包括众所周知的行业、产品、地理信息等分类方法.随着社会和科技的进步与发展,很多信息的分类法规范会新建和消亡,因此需要有一个灵活多变且又易于实现的模式来实现分类法规范的表示和引用.

UDDI信息模型中的 categoryBag结构就是在 tModel结构的支持下实现了这个功能.其结构规范说明如下.

```
<element name= "categoryBag">
  <type content= "elementOnly">
    <group order= "seq">
      <element ref= "keyedReference" Minoccurs= "0" maxoccurs = "* "/>
    </group>
  </type>
</element>
```

它包含了零个或多个 keyedReference元素的实例,通过一个指向 tModel结构的额外引用来实现分类法模式的可扩展性.

4 UDDI应用程序接口(API)

UDDI的应用程序 API规范分为两个逻辑部分:查询 API和发布 API.程序员可以使用这些 API开发各种类型的工具,以实现与 UDDI注册中心的交互,便于企业技术人员管理 businessEntity或 tModel结构的发布信息.

4.1 查询 API

查询 API包含两类调用,使程序能快速地定位候选商业实体、Web服务及其调用规范,然后在最初调用获得初始信息的基础上,获得进一步相关信息的细节.这类以 find-xxx

命名的 API提供了多种搜索,从而能对注册中心中的数据进行广泛地搜索.另一方面,如果已经知道所需数据的关键字,则可以通过直接调用 get_xxx API得到相应的结构数据(如 businessEntity、businessService、bindingTemplate、tModel等).

4.2 发布 API

发布 API包括四个 save_xxx函数和四个 delete_xxx函数,每个对应于一个 UDDI结构(businessEntity、businessService、bindingTemplate、tModel).一旦得到授权,一个独立的机构可以注册 businessEntity或 tModel信息,也可以修改原先发布的信息.注册和修改功能通过调用 save_xxx函数实现,删除信息功能则通过调用 delete_xxx函数实现.

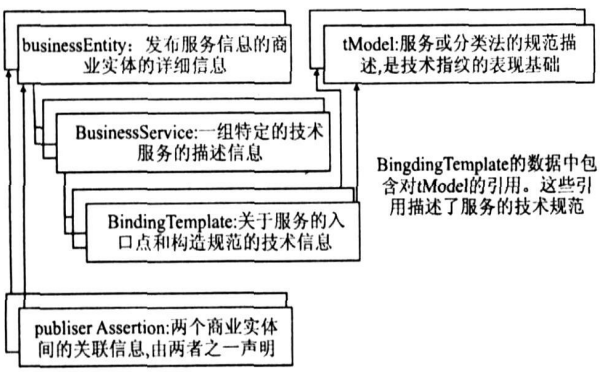

图 2　UDDI结构数据模型之间的关系

Fig. 2　The relationship between data models of UDDI structure

上述 API中 UDDI结构数据模型之间的关系如图 2所示.

5 仿真实验

为了测试上述分布式 UDDI互操作模型的性能,我们设计了系列仿真实验来仿真分布式 UDDI结构以及相关的服务提供者、服务请求者,并在其上通过服务查询请求的响应有效率、响应时间等度量标准来进行评估.首先利用 BRITE[11]拓扑生成工具产生包含 1001个节点的网络拓扑,每两个节点间的网络延迟依据节点间的最短路径算法计算.我们共设计了两种实验场景:

5.1 集中式 UDDI架构

由 3个节点担任 Web服务的注册中心,接收并响应服务请求者的查询请求.我们在节点上一共生成了 6000个 Web服务,并随机注册在 3个 UDDI Server的注册库中.

5.2 分布式 UDDI架构

由 1个节点担任根 UDDI,其余节点形成 5个逻辑 Super Domain,每个超域共包括 200个节点,其中 1个节点为 SD_UDDI Server,4个节点为 N_UDDI Server,其它节点为 Web服务的提供者或 Web服务请求者.我们在节点上一共生成了 6000个 Web服务,并随机注册在 25个 UDDI Server的注册库中.

在仿真过程中,为了避免大量无效请求(请求查询注册库中根本没发布过的 Web服务)影响评估结果,我们将仿真请求限制在已有的服务类别范围内,仿真主要考虑已发布的服务变化对于服务查询请求的影响,服务请求同样需要映射到网络拓扑结构的节点中.

第 1组仿真实验的目的是为了比较二种 UDDI结构响应结果的有效率,也就是响应的有效 Web服务数占总请求数的比率,这主要是考查 UDDI中心对失效 Web服务的感知能力.在文献[12,13]中报告,UBR中每周大约有 16%的 Web服务失效,我们以每天有 5%的 Web服务改变来仿真服务变化情况,其中 2%是由服务提供者主动发出的 Web服务更新,3%是失效的 Web服务并且服务提供者由于链路故障、宕机等原因无法通告 UDDI中心.实验中前 12天每天产生 5%的 Web服务变化,后 3天没有产生 Web服务变化,即没有 Web服务的更新和失效.每天发出 2000个请求来检查从 UDDI Server上找到的第 1个满足条件的 Web服务的可用性,仿真结果如图 3所示.

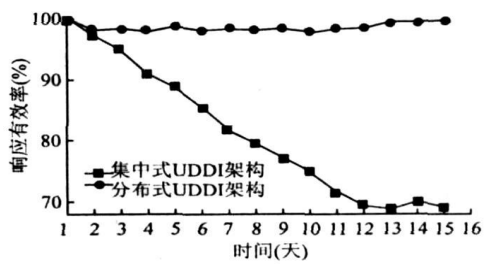

图 3　响应有效率

Fig. 3　Responding efficiency

从图 3可以发现,在集中式 UDDI架构中,在前 12天时间里,查询请求的响应有效率单调下降.这是由于 UDDI Server只能被动接受服务提供者的 Web服务更新,不能够主动探测失效的 Web服务,所以其注册库中将会存在越来越多的无效 Web服务,但 UDDI Server依然将其作为响应结果返回给服务请求者,导致响应有效率下降.在分布式 UDDI架构

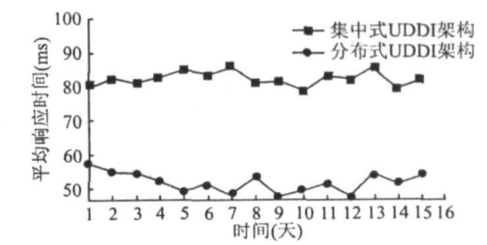

图 4　平均响应时间

Fig. 4　Average responding time

中,随着时间的推移,查询请求的响应有效率维持在较高的水平.这是因为 UDDI Server的 *certifier* 进程对本地注册的 Web服务进行定期检验,所以能够主动探测到失效的 Web服务,但由于 *certifier*进程检测的滞后性、在其它超域中可能已缓存该失效 Web服务并且还未"老化"等因素,UDDI Serv-

er对失效 Web服务的感知具有一定的延时,但这种延时是可以接收的.而在集中式 UDDI架构中,如果增加验证功能,将会进一步加大 UDDI Server的负荷,另外在集中式 UDDI架构中,UDDI Server注册库中的 Web服务数太多,对其进行验证不太现实.

第 2组仿真实验是为了比较两种 UDDI结构的平均响应时间,这里平均响应时间(Average Response Time)是指每天所有请求的响应时间的平均值,每个请求的响应时间为发出请求到收到 UDDI Server的响应信息所经历的时间.实验过程中没有 Web服务更新或失效,每天发出 2000个 Web服务查询,并计算它们的平均响应时间,仿真结果如图 4(见上页)所示.从图 4可以发现,分布式 UDDI架构的平均响应时间小于集中式 UDDI架构,这是由于在分布式 UDDI架构中,相当一部分 Web服务查询请求由本地 N_UDDI和 SD_UDDI响应,本地网络的高带宽、低延时使得响应时间更小.

6 结束语

本文针对现有 UDDI存在的问题,提出了一种分布式 UDDI互操作模型,将 UDDI注册中心分为根 UDDI、超域 UDDI和普通 UDDI三层,仿真实验表明该模型可以提高 UDDI的查询效率并实现了 UDDI注册库之间的信息互通.

作者所在的研究小组正在 Java平台上开发上述分布式 UDDI互操作模型的一个原形系统.整个系统架构以 Tomcat为 Web服务器,Axis担当 SOAP引擎,共同构成 Web服务环境,UDDI模块采用 JUDDI.Tomcat、Axis和 JUDDI均为 Apache组织发布的开源软件.UDDI间信息交互模块由 *uddi_Select_SuperDomain*、*uddi_Inform*、*uddi_KeepAlive*、*uddi_Push_Update*、*uddi_Drag_Update*、*uddi_Select_SDuddi* 等构成.下一步的工作将在实际网络环境中对整个模型的实现算法进行测试和优化,同时在模型的安全性和可靠性方面作进一步的研究.

References:

[1] Universal Description, Discovery and Integration (UDDI 2.0). UDDI technical white paper[EB/OL]. http://www.uddi.org/specification.html, 2001.

[2] Du Zong-xia, Huai Jin-peng. Research and implementation of an active distributed Web service registry[J]. Journal of Software, 2006, 17(3): 454-462.

[3] Chinnici R, Gudgin M, Moreau J J, et al. Web services description language (WSDL) version 1.2 part 1: core language[EB/OL]. Http://www.w3.org/TR/wsdl12, 2003.

[4] Gudgin M, Lewis A Schlimmer J. Web services description language (WSDL) version 1.2 part 2: message patterns[EB/OL]. Http://www.w3.org/TR/wsdl12-patterns, 2003.

[5] Clement L, Hately A, Riegen C V, et al. Universal description discovery & integration (UDDI) 3.0.2[EB/OL]. http://uddi.org/pubs/uddi_v3.htm, 2004.

[6] Endrei M, Ang J, Arsanjani A, et al. Patterns: service-oriented architecture and Web services[EB/OL]. Http://www.redbooks.ibm.com/redbooks/pdfs/sg246303.pdf, 2004.

[7] Cai M, Frank M. RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network[A]. In: Feldman S I, Uretsky M, Najork M, et al, eds. Proc. of the 13th Int'l Conf. on World Wide Web (WWW 2004)[C]. New York: ACM Press, 2004, 650-657.

[8] Shaikh Ali A, Rana O F, Al-Ali R J, et al. UDDIe: an extended registry for Web service[A]. In: Chang C, Murai J, eds. Symp. on Applications and the Internet Workshops (SAINT 2003)[C]. IEEE Computer Society, 2003, 85-89.

[9] Verma K, Sivashanmugam K, Sheth A, et al. METEOR-S WSDI: a scalable P2P infrastructure of registries for semantic publication and discovery of Web services[J]. Journal of Information Technology and Management, 2005, 6(1): 17-39.

[10] Oundhakar S, Verma K, Sivashanmugam K, et al. Discovery of Web services in a multi-ontology and federated registry environment[J]. Int'l Journal of Web Services Research, 2005, 2(3): 1-32.

[11] Medina A, Lakhina A, Matta I, et al. BRITE: an approach to universal topology generation[C]. In: Agrawal DP, ed. Proc. of the 9th Int'l Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS 2001), Cincinnati, 2001, 346-356.

[12] Kim S M, Rosu M C. A survey of public Web services[A]. In: Feldman S I, Uretsky M, Najork M, et al, eds. Proc. of the 13th Int'l Conf. on the World Wide Web (WWW 2004)[C]. New York: ACM Press, 2004, 312-313.

[13] Kim S M. Population of public Web services[EB/OL]. http://nclab.kaist.ac.kr/~smkim/ws_survey/index.html, 2005.

附中文参考文献:

[2] 杜宗霞,怀进鹏.主动分布式 Web服务注册机制研究与实现[J].软件学报,2006, 17(3): 454-462.