普适计算中动态绑定的研究与实现*)

张向刚　刘锦德

（电子科技大学自动化工程学院　电子科技大学计算机科学与工程学院　成都 610054）

摘　要　普适计算环境中，应用程序需要能够自动发现和使用环境中的可利用服务，以适应普适计算的动态性。本文中，作者使用动态绑定机制来完成服务到应用的自动映射，实现并详细阐述了一个基于 CORBA 的支持动态绑定的中间件原型 CWDB。在 CWDB 中，通过动态绑定层和高性能发现服务的组合来实现普适计算环境中的动态绑定。最后，通过一个位置显示应用来展示 CWDB 的适用性和其性能。

关键词　普适计算，动态绑定，中间件，发现

Research and Implementation of Dyanmic Binding in Pervasive Computing

ZHANG Xiang-Gang　LIU Jin-De

(College of Automation Engineering, College of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054)

Abstract　The pervasive computing is a dynamic environment. The applications in pervasive computing need have an ability to discovery and use the available services automatically. In the paper, the mechanism of dynamic binding is proposed to map the available services to the applications. A CORBA-based middleware for dynamic binding, named CWDB, is implemented and the details are explained. In CWDB, the dynamic binding layer and discovery service are composed to realize the dynamic binding function. At last, an electronic guide application, which shows the user's location, is implemented to verify the dynamic binding function and performance of CWDB.

Keywords　Pervasive computing, Dynamic binding, Middleware, Discovery

计算技术、通信技术、传感技术等新科技的飞速发展，将计算延伸到前所未有的广度和深度，带来了普适计算(pervasive computing)的出现。在这种环境中，自治、异构的应用、资源和设备在不断的更新、加入、离开及移动中，因此，动态性是普适计算环境的一个重要特征。应用需要在不同的环境中获取所需要的资源和信息，以完成其功能。这样的需求在现实中广泛存在，如：旅行者的导游设备需要在旅行中尽可能地获得定位、新闻和气象等服务；现代战场中，即使环境变化或基础设备遭到部分破坏，单兵、战车或飞行器也需要能够相互通信，了解环境各个方面的情况。所有这些情况都需要应用能够动态地发现和使用环境中存在的服务和信息。现在的系统一般使用发现服务来完成在动态环境中的资源和信息发现，但环境的改变将需要应用程序做出相应的改变，或要求应用程序增加其复杂性，以提高适应性。如何根据环境为应用提供更合适的资源，同时保持应用的简单性和稳定性，在本文中我们基于中间件 CORBA，设计和实现了支持动态绑定的中间件原型 CWDB(CORBA with dynamic binding)。CWDB 支持动态环境中资源和信息的自动获取，并保持了应用的简单性和稳定性。

1　CWDB 绑定模型和体系结构

在分布式系统中，对象之间相互作用(交互)的基础是绑定[1,2]。正是由于绑定，才使对象之间的互操作得以实现。绑定是这样一个过程，它按照一定的通信语义联系或互连计算系统中的各个对象。通过这个过程，一个对象(中的一个活动)与其它对象之间建立起了“可在后者的界面处调用其操作”的能力。绑定又是该过程的结果。绑定是一个复杂的过程。图 1 给出了 CWDB 绑定模型的抽象视图，其中绑定生成器根据 CLIENT 的服务需求、SERVER 的服务描述，以及环境信息，综合产生或动态调整 CLIENT 和 SERVER 之间的绑定。图 2 给出了 CWDB 的层次结构，其中，中间件部分采用支持无线功能的 wireless CORBA，以此来支持异构环境下的互操作，同时提供移动支持；发现服务实现动态环境中服务的发现过程，动态绑定层包含服务需求信息，代表应用程序动态地、自动地和发现服务交互，为应用程序提供环境中可用服务资源的信息。

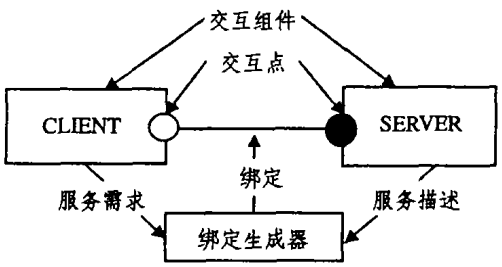

图 1　绑定和组件的抽象视图

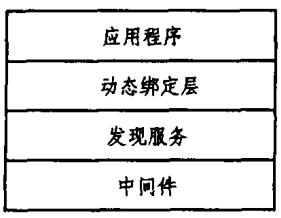

图 2　CWDB 层次结构

*)本课题得到总装备部军事预研项目(41315010103)资助。张向刚　讲师，主要研究领域为：中间件、普适计算。刘锦德　教授　博士导师，主要研究领域为：开放系统与中间件技术。

2 动态绑定层

动态绑定层对上层可以认为是应用程序的代理，对下层可以认为是整个环境的代理。主要功能是在动态的环境中，屏蔽环境的变化，自动为应用提供可利用服务的信息，不需要应用程序的干预和改变。应用只需向它提出服务请求，之后不需要了解下层的映射和环境的变化。它向应用提供统一的访问接口和所需的服务信息。

动态绑定层的主要组件包括接口模块、控制和仲裁、需求管理、绑定管理、发现服务接口以及 XML 引擎的支持。如图 3 所示。

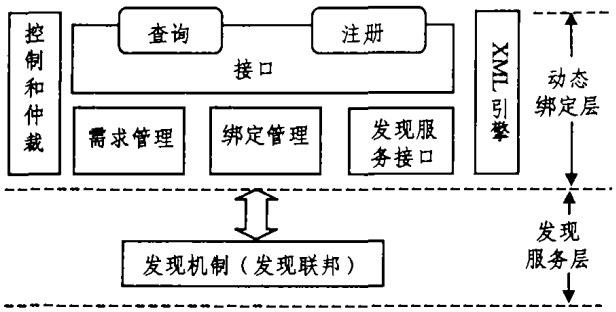

图 3 动态绑定层的体系结构

2.1 控制和仲裁模块

控制和仲裁模块协调动态绑定层中各个模块的功能，并提供一定的算法对返回的服务进行比较、评优和选择。应用程序通过注册接口将需求信息传送到需求管理模块，绑定管理模块将这些需求传送给发现机制（常为一发现联邦），发现服务根据需求进行查询，向动态绑定层返回满足要求的服务引用可能不只一个，这时需要根据控制和仲裁模块中的算法模块进行选择。在我们的实现中采用先到者优先。

2.2 接口模块

应用程序通过注册接口向动态绑定层注册需求信息，利用查询接口获取满足该需求的服务引用。其 IDL 接口定义如下所示。

```
//IDL
Interface DynLayerInterface {
    Int SubscribeRequirement (In string registerEvent;
                              Out int registerNum );
    Int DelSubscribe( in int registerNum);
    Int GetRequirementService (In int RegisterNum;
                               Out string RequirementServiceIOR );
}
```

操作 SubscribeRequirement：注册需求信息；参数 registerEvent：服务需求信息；参数 RegisterNum：注册号；操作 DelSubscribe：取消注册；参数 RegisterNum：注册号；操作 GetRequirementService：获取服务引用；参数 RegisterNum：注册号；参数 RequirementServiceIOR：服务对象引用。

2.3 绑定管理模块和需求管理模块

绑定管理模块 BM(Binding Management)动态、透明地确定(或调整)应用程序与所需要服务之间的对应关系。主要涉及的数据结构包括一张 Map Table。其中包含了应用服务需求与服务之间的对应关系。每个表项包括服务需求描述 ID 和满足该服务需求的服务引用。每个服务需求描述 ID 对应一张服务需求描述表 Req Table。绑定管理模块与发现联邦联合动态查询满足需求的服务，并动态更新 Map table。应用需要使用服务时，只需要查询 Map table，即可获得服务引用，而不用关心下层环境的变化。动态绑定层和发现服务层自动地完成了服务需求到相应服务的动态绑定。

需求管理模块通过一组 Req Table 完成应用需求的管理，通过一组 Service Table 记录选中服务的公告信息。

2.4 其它模块

发现服务接口模块监听公共地址和端口，了解可用发现服务的地址及端口，在自身维护一张可利用发现服务表。XML 引擎进行消息的 XML 格式封装和解封装。

3 发现服务—perTrader

在普适计算这样的环境下，对于给定的任务，用户如何定位合适的服务是其关键。服务发现技术能够帮助用户在网络中寻找所需的服务，检测服务可用性的变化，从而维护服务的一致性视图。因此，很多研究者都将发现服务定义为普适计算的关键技术[6,7]。较为典型的发现技术有：微软公司的 Universal Plug and Play(UpnP)[3]，Sun 公司的 Jini[4]和 Service Location Protocol (SLP)[5]等。

上述各种技术都具有各自的长处，但每种技术都存在各自的不足。总结起来发现服务的组织和发布方式可分为分散型、集中型两种模式。分散型模式的优点是简单、健壮性好，易于在计算能力较弱的设备上实现，但其缺点是受范围限制较大。集中型模式的发现范围可以很广，其缺点是健壮性较差，且随着范围和请求的增加效率会降低。普适计算环境需要在较大范围内进行服务发现，客户和服务数目众多，分散型模式显然无法适应，而集中型模式虽然也能工作但其效率将是非常低的，因此必须寻找新的解决方案。一种有效途径就是在集中型服务发现协议的基础上建立各个服务目录之间的协作机制及相应的基础设施，从而拓展服务发现的范围。

3.1 基于交易服务的方案

CORBA Trader Service(TS)交易服务是标准的 CORBA 服务[8]。CORBA 对象使用交易器定位满足特定约束的对象。但标准交易服务相对于上述的发现协议不具有设备和服务的自发发现和配置、低(或零)人为管理的要求、对动态的网络缺乏自适应性、保证注册对象的可利用性等特征。但交易服务器提供了丰富的表达对象特性的形式，交易器联邦具有任意的拓扑结构，能够根据应用需求提供良好的可扩展性，以及 CORBA 平台的安全机制和高效性等促使我们选用交易服务作为发现服务的基础，并针对普适计算环境增加了通知机制、租约机制，以及询问路由机制，形成了适合普适计算环境的发现服务——perTrader。

3.2 PerTrader 体系结构

perTrader 的体系结构如图 4 所示。其中白色部分为 OMG CORBA 规范定义的标准接口，灰色部分为 perTrader 增加或增强的接口，XML 引擎向系统的其他部分提供调用接口，进行封装和解封装。其中灰色部分将在下面进行详细说明。

3.2.1 Link 在 perTrader 联邦管理中，为提高联邦整体的查询效率，perTrader 之间按照最小生成树建立连接，同时 link 采用双向连接。

交易器联邦中，询问在交易器之间的路由策略，对提高交易器联邦的整体效率至关重要。交易器之间的路由不同于网络路由在于：交易器路由中转发的是带有需求语义信息的询问，而网络路由转发的是带有 IP 的数据报文。询问的语义以及交易器上的注册信息在不断的变化。因而，不同环境的不同时刻，询问的目的地都有所不同或有多个。因此，交易器应该

利用这些动态信息来有效地提高路由效率。

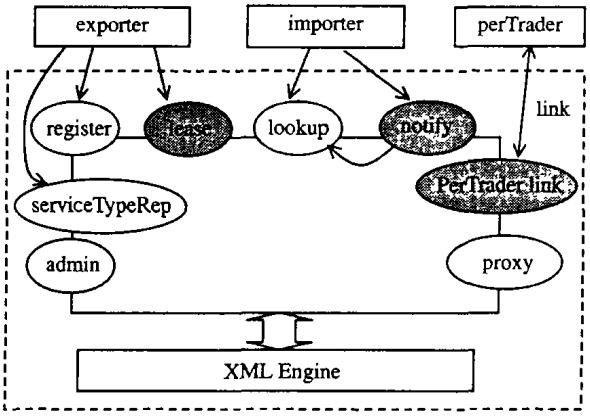

图4 PerTrader 体系结构

为了反映某个 perTrader 能够提供的 service type 及其 service offer 的情况以及它们的变化，有效使用 perTrader 中的动态信息，提高查询效率。我们在系统中增加了如下的数据结构。

·linkTable：包含这个 perTrader 所 link 向的所有 perTrader 的引用信息；

·callback table：包含指向这个 perTrader 的所有 perTrader 的地址；

·service table：包含 service type 及其对应的 SF；每个 perTrader 为自己构造一个 service table，包含该 perTrader 本地能够查找的 service type 及其 SF；linkTable 的每一项拥有一个 service table，包含从这条链所指向的 perTrader 能够查找的 service type 及其 SF。

当一个 perTrader 宣称针对某种 service type 拥有 N 个 service offers，这意味着通过这个 perTrader 能够本地或异地查询到 N 个 service offers。但不能仅此来决定查询的目的地。如：假设存在两个交易器：TA 和 TB。TA 拥有某种 service type 的 service offer 500个，和100个只有1跳的 service offers。TB 拥有5个 service offer，和1000个100跳的 service offers，显然，首先对 TA 进行查找，有利于节约查找时间。因此，询问路径应该尽量首先询问距离近且 service offer 多的 perTrader。这样能够避免先向距离远且命中率低的 perTrader 发出询问。

perTrader 联邦中采用基于存储因子 SF(storage factor)的路由策略，SF 与 service type 相对应。在某个 perTrader 中，某种 service type 的存储因子的确定与这个 perTrader 能够查询到的该 service type 的 service offer 数量有关，同时也与提供这些 service offer 的 perTrader 和这个 perTrader 的距离有关。

不论什么时候，当某种 service type 的 service offer 发生变化的时候，例如：新 service offer 的注册或旧 service offer 的撤消，将会引起 perTrader 根据下面计算公式对相应 service type 的 SF 的重新计算。

$$SF = SF + (changedOfferNum \times e^{-\lambda \times n})$$

n 表示两个 perTrader 之间的跳数，两个交易器之间为1跳，λ为常数因子，通过 λ 来调整距离在整个 SF 中的影响力，ChangedOfferNum 表示该 service type 的 service offer 改变的数量，如：100表示增加100个。本地计算 $n=0$，更新该 perTrader 自己的 service table。之后，该 perTrader 根据 callback table 将这个变化反映到源 perTrader（指向自己的 perTrader）中，同时引起源 perTrader 按照上式更新相应 link 对应的 service table，计算中 n 递增1。之后，源 perTrader 又根据自己的 callback table 将这个变化扩散出去（除去向自己发送变化的 perTrader），每扩散一次 n 递增1。因此某个 perTrader 上，当服务增加或减少的时候，这个信息被传送到所有能够直接或间接访问这个 perTrader 的所有 perTrader 上，以修改所有 perTrader 上的相关信息描述。

当一个 perTrader 得到服务需求的时候，首先查询自己本地存储，如果成功，返回相应的 IOR。如果不成功，根据 linkTable 所对应的 service table。将首先向 SF 最大的 perTrader 询问，然后向次大的 perTrader 询问，以此类推。

3.2.2 通知机制 通知(notify)是一个异步接口，动态绑定层通过通知接口向 perTrader 注册服务请求信息、回调对象，以及相关的参数信息（如：返回 offer 的优先顺序、策略、返回 offer 的属性集等）。当满足要求（或更好地满足要求）的服务出现时，交易器通过回调对象以 push 方式将服务信息推送给动态绑定层，以此来反映环境的动态变化。

```
//IDL
void notify {
        in ServiceTypeName type,
        in string constr,
        in callback_object callback_obj,
        in Preference pref,
        in PolicySeq policies,
        in string desired_props,
        }
```

其中："type"参数表示请求的服务类型；"constr"是输入方给出的服务要求；"pref"参数被用于对符合条件的服务供应进行排序，最大限度地满足输入方的要求；"policies"参数允许输入方指明如何进行服务搜索；"desired_props"参数定义描述被返回服务供应的属性集。

3.2.3 租约机制 在普适计算环境中设备的移动、传感的失败和环境的改变是不可避免的，因此，必须跟踪服务的存在。发现服务通过租约(Lease)机制跟踪环境中所有存在的服务，所有的服务需要周期性地向发现服务发布其"心跳"(heartbeat)或"租约"(lease)。"心跳"或"租约"至少包含：

ServiceId：唯一的标示 ID。

ValidDuration：租约的持续时间（n 个固定时间段）。

发现服务周期性地检查注册服务的租约，每次检查对 ValidDuration 减1，同时删除 ValidDuration 为零的注册。

新的服务加入环境时，通过广播或事先了解的发现服务的地址，向发现服务注册其服务。发现服务向其返回一个唯一的服务标示 ID，在以后的时间内，服务周期性地向发现服务传送服务标示 ID，以表示其可用性。

其接口的 IDL 定义如下所示。

```
//IDL
Interface Lease
{
    Bool UpdataLease(string ServiceId,integer ValidDuration )
}
```

ServiceId：唯一的标示 ID。
ValidDuration：租约的持续时间（n 个固定时间段）。

4 动态绑定过程

应用向动态绑定层注册需求信息。动态绑定层将服务需求复制给可利用的 perTrader。perTrader 根据服务需求进行查询，并将满足需要的服务引用返回给动态绑定层，动态绑定层进行仲裁后，将相应的 IOR 填入 Map table。

（下转第20页）

如图中所示，本地代理的管理代理在收到本地代理发出的 JLA send 资源请求消息后，将此消息转发给其对应的5个 AMR(为说明方便，我们在这里假定是5个)。每个 AMR 在收到资源请求消息后，再转发给其他区域的多代理 JMA1，然后由 JMA1转发给 JLA。JLA 在收到请求后，进行相应的 search 操作，得到相应的结果，此时 back JMA 事件触发。管理代理 JMA2在得到所有相应的 JLA 的返回结果后，执行 back AMR 事件，将结果返回给 AMR2，再返回给 JMA3和发送资源请求的 JLA。当所有 AMR 都返回结果时，系统流程结束。

结束语 下一步我们将对原型系统进行实现。对系统的功能做出更详细的设计，对各个代理的功能和代理之间的接口进行更进一步的设计实现；在充分考虑到该系统与 Globus 所提供的各逻辑功能的结合的同时，将人工智能和知识学习的方法嵌入到代理当中。该系统力求实现对网格任务的更加灵活高效地调度，并具有良好的可扩展性和可移植性。

参 考 文 献

1 Foster I. The anatomy of the Grid: Enabling scalable virtual organizations [J]. Concurrency and Computation: Practice and Experience. 2001,13

2 Bruneo D,et al. Communication Paradigms for Mobile Grid Users [C]. In:Proc. of the 3rd IEEE/ACM Intl. Symposium on Cluster Computing and the Grid. 2003 IEEE

3 Li Chunlin,Li Layuan. Agent framework to support the computational grid [J]. The Journal of Systems and Software. 2004,70: 177～187

4 Wooldridge M J, Jennings N R. Intelligent agents: Theory and practice [J]. The Knowledge Engineering Review, 1995, 10(2): 115～152

5 Wijngaards N J E,Overeinder B J,van Steen M,Brazier F M T. Supporting Internet-scale multi-agent systems [J]. Data and Knowledge Engineering,2002

6 Manola F, Thompson C. Characterizing the agent grid. http://www. objs. com/agility/techreports/990623-characterizing-the-agentgrid. html, June 1999

7 Rana O F, Walker D W. The Agent Grid: Agent-based resource integration in PSEs [A]. In: Proc. of the 16th IMACS World Congress on Scientific Computing, Applied Mathematics and Simulation. Lausanne, Switzerland, Aug. 2000

8 Yushun F, Junwei C. Multi-Agent Systems: Theory, Method and Applications [M]. German: Springer Press, 2002, 5

9 van der Aalst W, van Hee K M. Workflow Management: Models, Methods, and Systems [M]. 2002

(上接第13页)

当应用有新的需求向动态绑定层注册时，将激发又一次的需求请求复制和查找过程。当 perTrader 中注册的 service type 或 service offer 发生变化时(通过 storage factor 了解)，同样也激活又一次的查询过程，并将相应的结果返回给动态绑定层，动态绑定层依然是经过仲裁后，填写 Map table。

应用需要获得服务引用时，只需要查询 Map table，而不用关心下层环境的变化。动态绑定层自动完成需求到相应服务的动态更新映射。

如图5所示，环境中可用的服务发生了变化，原来的服务①不能使用了，同时出现了新的服务②，动态绑定层和发现联邦层结合实现需求和提供之间的动态绑定，应用始终能获得可利用的服务。

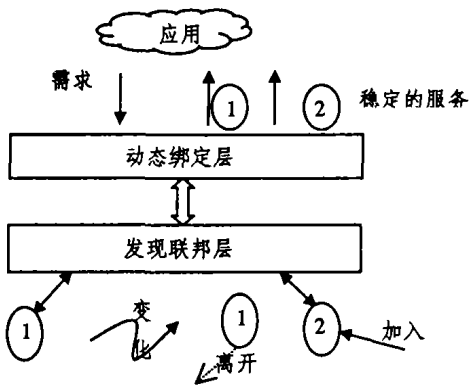

图5 服务的动态绑定

5 试验验证

为验证 CWDB 对动态绑定功能的支持，我们在 PDA 上实现了一个位置显示应用，PDA 具有 GPS 定位功能，整个系统包含室内部分和室外部分，室内部分装配了室内定位系统 Active bat。当 PDA 在室外的时候，位置显示应用使用 GPS 向用户显示位置信息，当 PDA 进入室内的时候，位置显示应用自动使用 Active bat 获取位置信息(室内状态下，GPS 失效)。从而验证了 CWDB 能够在动态环境中实现动态绑定，并保持了应用程序的稳定性和简单化，同时也验证了发现服务的高性能。

小结 普适计算环境是一个动态的环境，如何向应用程序屏蔽环境的动态性?这个问题传统的中间件没有解决。文中我们通过在中间件的基础上引入动态绑定机制来完成这个功能，并开发了相应的原型 CWDB，通过实验验证了 CWDB 对动态绑定的支持，及其高性能。

参 考 文 献

1 ISO/IEC and ITU. Reference Model of ODP Part 1 Overview. May 1995

2 Otway D. The ANSA Binding Model. ANSA Phase 3 Project Report APM. 1392. 01, Jan. 1995

3 Universal Plug, Play Forum. Universal Plug and Play Web Site. 2000. http://www. upnp. org/

4 Edwards W K. Core JINI. Upper Saddle River, NJ: Prentice Hall, 1999

5 Guttman E, Perkins C, Veizades J, Day M. Service Location Protocol. Version 2. RFC 2608, June 1999

6 Nigel D, Mitchell K, Cheverst K, Blair G. Developing a context-sensitive tour guide. In the 1st Workshop on Human Computer Interaction for Mobile Devices, Glasgow, Scotland. May, 1998

7 Gray P, Salber D. Modeling and Using Sensed Information in the Design of Interactive Applications. 8th IFIP Intl. Conf. EHCI 2001, Toronto, Canada, 2001

8 Object Management Group. CORBAservices: Common Object Services Specification. Object Management Group 1999