基于构件的中间件体系结构集成形式化研究

王 琼[1] 杜承烈[1] 蔡小斌[2] 李 刚[1]
(西北工业大学计算机学院 西安 710072)[1] (中航一集团科技委 北京 100012)[2]

摘 要 航空航天等领域大型复杂系统的构建离不开中间件技术的支持。研究了中间件体系结构设计的关键技术和集成机制,介绍了基于TDπ演算的软件体系结构描述方法,提出了一个带有性能约束的构件接口模型,并研究了基于该模型的构件与系统间的集成兼容性。该研究在某虚拟试验实时软总线的研究中得到了应用,起到了比较好的效果。

关键词 中间件,体系结构,集成,兼容性,TDπ演算,软总线
中图法分类号 TP303 **文献标识码** A

Research of Integration of Middleware Architecture

WANG Qiong[1] DU Cheng-lie[1] CAI Xiao-bin[2] LI Gang[1]
(Computer College of Northwestern Polytechnical University, Xi'an 710072, China)[1]
(Committee of Science and Technology, China Aviation Industry Corporation I, Beijing 100012, China)[2]

Abstract Building large-scale complex system in military field needs the support of middleware technologies. The crucial technique and the integration mechanism of middleware architecture were researched. This paper introduced software architecture formal description method based on TDπ-calculus and proposed a performance bound component interface model. Then we studied the integration compatibility between component and system based on this model. This study has already used in a virtual test real-time soft-bus research and has well effect.

Keywords Middleware, Architecture, Integration, Compatibility, TDπ-calculus, Soft-bus

航空航天等领域大型复杂系统的建立离不开中间件技术的支持。而现有中间件缺乏面向领域的扩展能力,不足以满足分布式应用中不断出现的新需求,其根本原因在于其开放性和灵活性不足。因此,新一代中间件体系结构应当具有工程开放性和对所依赖环境的自我调整能力[2],能够更好地支持进一步集成与扩展。已往软件体系结构的描述大多采用传统的图框描述和自然语言相结合的方法,这些方法简单易懂,但是不够严谨,容易造成歧义;形式化方法是一种用于规范设计和验证计算机系统的基于数学的方法,它能在体系结构设计阶段帮助发现其它方法不容易发现的系统描述的不一致、不明确或不完整,其分析工具在设计阶段就可以分析待建系统的集成与扩展的兼容性;因此,形式化方法是提高软件系统,特别是 safety-critical 系统的安全性与可靠性的重要手段,也是描述新一代中间件体系结构的不可缺少的方法。

集成与扩展是现代大型复杂系统的一个至关重要的需求。本文以国防某基础科研项目——航空航天领域某分布虚拟试验关键技术的研究为背景,研究了分布式系统中中间件体系结构的设计及其集成机制,分析了如何用形式化方法描述构件及其接口以支持软件体系结构的进一步集成与扩展。本文第1节研究了中间件体系结构及其集成机制;第2节介绍了一种适合于描述分布式软件体系结构的形式化方法——π演算,并分析了为什么选用基于TDπ演算的体系结构描述语言(TDπ-ADL)来描述基于构件的软件体系结构;在第1、2节的理论基础上,第3节提出了一个具有性能约束的构件接口模型并研究了基于该模型的集成兼容性;第4节以航空航天某虚拟试验实时软总线的研究为背景,深入研究并实现了基于网关的实时软总线体系结构的网络异构集成方法,同时用TDπ-ADL对网关进行了端口和行为描述;最后为结束语。

1 中间件体系结构及其集成机制

1.1 中间件体系结构的设计

尽管中间件技术发展到现在已比较成功,而且相应的一些标准也比较完善,但目前的中间件技术还不能很好地满足群件、多媒体、实时传播及移动计算等应用需求[4,10]。面对这些新的挑战,中间件需要一种新的解决方案,即构造一个新的中间件平台,它应当具有以下特性[1]:

- 可配置的,用于满足给定应用领域的需求;
- 动态可重配置的,使得中间件平台能够对环境的改变做出反应;
- 可扩展的,以满足改变平台设计的需要。

到稿日期:2009-09-23 返修日期:2009-12-19 本文受航空科学基金(04I53066,2007ZD53043),"十一五"武器装备预先研究项目(102010101)资助。

王 琼(1973—),女,博士生,主要研究方向为分布测控与仿真,E-mail:wq78026@gmail.com;杜承烈(1970—),男,教授,博士生导师,主要研究方向为分布测控与仿真;蔡小斌(1957—),男,教授,博士生导师,主要研究方向为航空试验发展战略;李 刚(1981—),男,博士生,主要研究方向为分布测控与仿真。

为此，在中间件体系结构设计中要用到下面几个关键技术[3,5,9]：

1. 反身映射技术(Reflective Techniques)　该技术使得在不用公开无关的执行细节或放弃可移植性特征的情况下就可以实现开放语言的执行。它允许一个程序访问、推断和改变该程序本身，对程序本身的访问是通过一个元对象协议(meta-object protocol，MOP)来实现的；该元对象协议定义了在元层次上所有可行的服务，如元层次上可行的操作，包括在方法调用前后改变消息传送和插入语义等。反身映射为实现工程开放性提供了一个很好的方法，如反身映射能用于检测中间件的内部行为，通过插入一些行为方法去监视中间件的其他行为。反身映射同样可用于调整中间件的内部行为变化，如通过更换消息传送来适应无线连接的最优化操作，在移动计算中插入一个过滤器对象来降低一个通讯流的带宽要求等。

2. 面向对象的选择(The Choice of Object-Orientation)　在开放性分布式的处理中，面向对象计算模型有着一定的优势。这个对象模型的主要特征是：1)一个对象可以有多个接口；2)支持流和信号处理接口；3)在可兼容的接口之间创建开放性绑定(其结果是一个绑定对象的生成)。该模型是一个很成熟的模型，能提供质量服务，其具体细节参考文献[6]。

3. 每对象(或每接口)的元空间(Per object(Per Interface) Meta-Space)　设计中应包含每对象(或每接口)的元空间，这在异构环境中尤其重要，因为在异构环境中，对象包括许多关于反身映射的容器。这种解决方案能对中间件平台中的服务进行很好的控制。由于变化的空间有限，维持完整性的问题就能被减少到最小程度。

4. 过程方法(Procedural Approach)　为了具有反身映射性质，采用了过程方法，如元层次公开了系统执行的实际过程。这种方法同其它已经公布的方法相比，有着许多优点，例如，过程方法非常简单(尤其是能支持映射过程上已经公布的接口，反之则不然)。

5. 构造元空间(Structure Meta-Space)　根据许多关系紧密但有一定区别的元空间模型去构造一个元空间。这个方法首先由 AL-1/D(一种分布式应用映射程序设计语言)的设计者们提出；它的优点是，通过维护不同系统，简化了由元空间提供的接口。元空间主要有：组装元空间(表示对象图中组装对象(比如绑定)的配置问题)、封装元空间(处理对象的接口属性)、环境元空间(表示交互接口执行中包括的处理)。

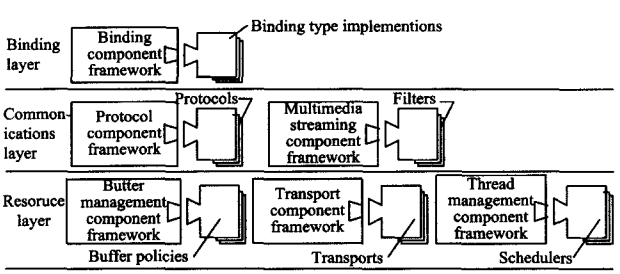

图 1　基于构件的中间件体系结构

图 1[1,3]所示的基于反身映射和构件的中间件体系结构是基于 OpenCOM 的构件框架(component framework，CF)集的扩展。它考虑了一个基础的通信中间件平台(例如没有处理事务支持)，描述的 CF 集是最小需求(该需求受到 GOPI 平台[3]的影响)，这些 CF 按照 3 个层次来组织，其中构件只是针对它们的下一层定义了接口/CF。注意，根据 OpenCOM 架构，构件只是当它们实际需要时才被载入，不用时被卸载。

1.2　集成机制

中间件平台必须兼容应用和所依赖环境带来的广泛的需求，必须能够吸取这些需求在设计时和系统运行时的改变及对已有技术的集成。不幸的是，当前的中间件大都不能解决这些问题。有些研究者曾经引入可重配置，但是它们是典型的初步 ad-hoc，通常只是在一系列可选项中进行选择。因此，需要一个更加系统和更加规范的解决方案。在中间件体系结构设计中，通常采用以下机制来支持进一步的集成[1-5]：

1)构件框架(CF)。CF 是管理一个构件集交互的规则和契约的集合。CF 将一个特定领域的问题处理为一个构件，比如将通信协议作为一个构件，于是一个构件系统中的多个 CF 需要集成。CF 的首要作用是通过限制扩展构件的设计空间提供内置的结构特性和常量。此外，CF 简化了构件的开发和装配，实现了轻量级构件和增加了系统的可理解性与可维护性。

CF 由基础构件和合成构件组成，每一个构件都由与之相联系的工厂对象支持(可以通过简单地引入一个新的工厂对象来扩展 CF)。它基本上能体现以上所述的抽象体系结构，其构件能用于反身映射体系结构的所有层次中。

2)构件框架(CF)元接口。CF 通过元接口相联系。通过元接口，基于 CF 的子系统的执行可以在一定控制和规则下敞开。元接口通常被 CFR 执行，它保留关于当前配置的信息并用该信息实现监测和调整。

由于元接口被设计成 CF 的一个完整部分，它们可以容易地表达特定领域的知识，可以实现理想的一致性和完整性；因此元接口所提供的灵活性的等级要与有效性、一致性、集成性、可理解性相平衡。通过应用特定的元接口，可以为每个不同领域实现不同的交易。

3)统一的构件模型。在上面的体系结构中，应用和中间件构件、CF 全部采用相同的构件模型。这有利于增加互操作性、灵活性和节约开发成本，因为开发人员只需熟悉单一的编程模型。另外，它还消除了应用和中间件之间的差异，解决了应用开发和系统开发之间的传统障碍，必要时前者可以使用后者的开发人员。

2　中间件体系结构形式化描述方法研究

2.1　π 演算简介

π 演算[6]是 Robin Milner 提出的以进程间的移动通信为研究重点的并发理论，是对 CCS(Calculus of Communication System) 的发展。其基本计算实体为名字和进程，进程之间的通信是通过传递名字来完成的。由于 π 演算不但可以传递 CCS 中的变量和值，还可以传递通道名，并且将这几种实体都统称为名字而不再作区分，使得 π 演算具有了建立新通道的能力，因此 π 演算可以用来描述结构不断变化的并发系统。π 演算进程的语法定义如下：

$$P::=0\mid\pi.P\mid P_1+P_2\mid P_1\parallel P_2\mid(\nu a)P\mid !\ P$$

演算进程的定义描述了 π 演算进程之间存在的各种交互模式，关于 π 演算及其交互模式的详细内容请参阅文献[6]。近年来，不少学者又对 π 演算进行了时间和分布式方面的扩

展，出现了 Timed-π 演算、Distributed-π 演算以及 TDπ-演算。

2.2 基于 TDπ-ADL 的软件体系结构描述

在众多的 ADL 中，Wright，Darwin，π-ADL[7] 和 D-ADL[8]能够描述系统的并发行为，它们关注系统的结构视图和计算行为视图，但是前两者忽略了系统的动态行为，只能描述系统的静态行为。而具备对动态行为的表示能力是动态 ADL 的显著标识。基于高阶多型 π 演算的 π-ADL 和 D-ADL 虽然能够很好地描述体系结构的并发行为和动态行为，但是没有考虑系统的时间特性。事实上，大规模复杂系统尤其是军工领域的虚拟试验系统大多是实时分布式系统，系统资源分别处于不同的位置，而且资源的提供也不可能是无限期的，因此，实时性、并发性和分布性是分布式实时中间件所必须面对的问题。

TDπ-ADL(详见本人即将完成的博士论文：分布虚拟试验支撑环境的研究及应用-基于构件的软件体系结构形式化描述及其集成研究)综合了以上 ADL 的优点和实时分布式系统软件的需求，遵循 ACME，Wright 等给出的已被广泛认同的体系结构描述框架，提供专门的标记符号，围绕体系结构实体如构件、连接件、系统配置等进行体系结构建模；TDπ-ADL 将 π-ADL 和 D-ADL 作为语言结构主体，在其基础上引入类型系统和 timer 对资源的位置和具有实时特征的行为、端口等进行扩展描述，满足了实时分布式系统的实时性、并发性和分布性表达需求。用 TDπ-ADL 描述基于构件的软件体系结构的方法如图 2[7]所示。

```
Abstract syntax of architectural abstractions
syntax of architectural abstractions
architecturalAbstraction ::=archtype name is abstraction
archtype::=architecture | component | connector
abstraction::= abstraction ( name0 : ValueType0, …,
                    namen : ValueTypen ) {
                    type0 . … . typen .
                    value0 . … . valuen .
                    port0 . … . portn .
                    behaviour is { behaviour }
                  } assuming { property }
type ::= type name is ValueType
value ::= value name is expression
port ::= port name is restriction {
            connection0 . … . connectionn
          } assuming { protocol is property }
restriction ::= free | restricted
connection ::= connection name is mode ( ValueType )
mode ::= in | out | inout
```

图 2　TDπ-ADL 的软件体系结构描述语法

3　中间件体系结构集成的兼容性研究

目前的中间层软件大多采用基于构件的体系结构，基于构件的中间件体系结构的集成主要是通过对构件的集成来实现的。构件的一个显著特点是具有封装性，而接口是构件与外部进行信息交换的唯一场所，因此构件外部化其接口作为集成点。一个构件能否被一个系统兼容，很大程度上取决于它的接口设计。实时构件的接口不仅涉及到信息的交互，还涉及到实时等性能方面的控制。本文提出一个带有性能约束的面向体系结构的构件接口模型，它的描述如下：

```
Component Interface Specification::=
INTERFACE interface_name
{
[require function_port_section]+ //1 个或多个功能需求端口
[provide function_port_section]+ //1 个或多个功能提供端口
[Performance_control_port_section]* //0 个或多个性能控制端口
[Interfact_level_protocol] //1 个构件接口级的交互协议定义
}
```

相应地，一个构件能否被系统集成、集成后系统能否仍然保持一致性，即能否被兼容的问题，可以通过静态分析构件的端口类型和系统的运行环境来判定。在给出下面的命题之前，先定义几个端口的符号。

定义 1　r_c 为构件的功能需求端口，p_c 为构件的功能提供端口，p_{con} 为构件契约端口，r_s 为系统对于该构件的功能需求端口，p_s 为系统对于该构件的功能提供端口，E_{sp} 为系统提供的运行环境，E_{cr} 为构件需要的运行环境，I_{sr} 为系统要求构件实现的性能控制指标，I_{cp} 为构件可以提供的性能控制指标范围，若端口为 p，则相应的端口类型为 $T_p(p)$。

命题 1(服务端口一致性)　如果 $T_p(p_c)\geqslant T_p(r_s)$，$T_p(r_c)\leqslant T_p(p_s)$，则服务端口能保持一致性。

由假设条件知，构件所需求的服务都可以从系统中得到，系统对于构件所需求的服务都可以从构件中得到，需求端口类型为提供端口类型的子集；构件对于系统来说，相当于一个完整进程中的一段，对输入信息进行处理后再输出到系统，因而系统集成该构件后仍能保持一致性。

命题 2(性能控制端口一致性)　如果 $I_{sr}\in I_{cp}$，$E_{cr}<E_{sp}$，则性能控制端口能保持一致性。

当假设条件成立时，显然结论是成立的。

4　某虚拟试验实时软总线的集成研究

在航空航天某虚拟试验的关键技术平台中，软总线是将试验服务和试验应用连接起来的数据纽带，通过软总线，可以将各个试验应用产生的数据在网络上进行实时传输；在航空航天领域的虚拟试验中，软总线大多采用美国的 HLA/RTI 来实现，但是，RTI 目前只是运行在以太网上，还没有针对 SBS 广播内存网和 VMIC 反射内存网等实时网络的实现。而为了提高虚拟试验系统的实时性，需要对该软总线进行集成，使之可以在上述的实时网络上运行。集成后的软总线称为实时软总线，其拓扑结构如图 3 所示。

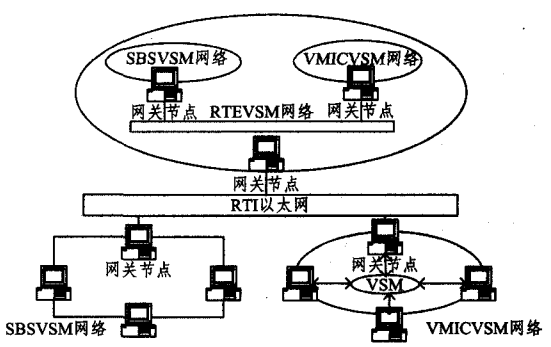

图 3　虚拟试验网络拓扑结构图

从图 3 可以看出，要解决不同网络之间的信息交换，最主要的是不同网络之间的协议转换。总线就是一个中间件，增

加的网关相当于一个构件，它们之间的信息交换都要通过网关接口；网关接口包含两个端口：连接 RTI 网络的 RTI. port 和连接 VMIC 网络的 VMIC. port，其功能是将来自 RTI. port 和 VMIC. port 端口的数据分别转换成 VMIC 和 RTI 协议格式并实现两端网络的同步。下面用 TDπ-ADL 来描述 VMIC 与 RTI 协议的转换网关的端口及其行为。

1）网关的端口描述

```
component GateWay is abstraction() {
      type Data. RTI is Any. type Data. VMIC is Any. type.
      port RTI. port is{connection Data. RTI is in(Data. RTI). }
      port VMIC. port is {connection Data. VMIC is in (Data. VMIC). }
      port Perfor-con. port is {connection Data. RTI is in (Data. RTI). }
    assuming {
      protocol is {( via Data. RTI receive any. true * . via Data. VMIC send any. via Data. VMIC receive any. true * . via Data. RTIC send any. if the interval of data from Data. VMIC is longer than some time-value than via Perfor-con. port send a message) * }
  }.
}
```

2）网关的行为描述

```
component GateWay is abstraction() {
  ...
behaviour is {
      protocoltransform is function (datasource): Data {unobservable}
if datasource::RTI. port then{
  via RTI. port receive Data. RTI.
      adjust data. sendinterval.
  via VMIC. port send protocoltransfor(Data. RTI). }
      if datasource::VMIC. port then{
via VMIC. port receive Data. VMIC.
adjust data. sendinterval.
via RTI. port send protocoltransfor(Data. VMIC). }
      behaviour()
}
}
```

上述端口协议中的最后一条用于实现实时性能控制，即当 VMIC 网络出现故障或其它原因导致接收数据超时时，网关将向 RTI 发出消息以启动备用网络。为了系统的安全性，网关端口必须满足第 3 节中的命题 1 和命题 2。

结束语 中间件是支持分布式应用的一个重要的体系结构构件，其作用是掩盖了分布和异构问题从而向应用开发者体现出一个统一的编程模型。为了支持可重用和实现工程开放性，新一代中间件体系结构大多是基于构件和采用反身映射技术的，这使得用户能够根据需要方便地中间件进行裁剪和扩展，提高了应用系统的灵活性。形式化方法能帮助发现其它方法不容易发现的系统描述的不一致、不明确或不完整，有助于增加软件开发人员对系统的理解；因此形式化方法是提高软件系统，特别是安全攸关系统的安全性与可靠性的重要手段。

本文分析了 TDπ-ADL 的优点和基于 TDπ-ADL 的软件体系结构的描述方法，提出了一个带有性能约束的构件接口模型，并深入研究了基于该模型的集成兼容性。研究结果在以航空航天某虚拟试验实时软总线的研究中得到了应用，该集成方法提高了系统的实时性和稳定性，起到了比较好的效果。

接下来的工作是，进一步细化构件接口模型，研究基于接口的构件集成与扩展机制并用基于 π 演算的模拟工具进行模拟。

参 考 文 献

[1] Blair G S, et al. Reflective Middleware[J]. The Design and Implementation of Open ORB 2, 2001, 2(6). http://www.computer.org/portal/site/dsonline/

[2] Blair G S, Coulson G, Robin P, et al. An architecture for next generation middleware[C]// Davies N, Raymond K, Seitz J, eds. Proceedings of the Middleware'98. Springer-Verlag, 1998: 191-206

[3] Parlavantzas N, Coulson G, Clarke M, et al. Towards a reflective component-based middleware architecture[C]// Cazzola W, eds. On-Line Proceedings of ECOOP 2000 Workshop on Reflection and Metalevel Architectures. 2000. http://citesernj. nec. com/331827. html

[4] 胡志远，顾君忠. 中间件体系结构研究[J]. 小型微型计算机系统，2003，8：1466-1469

[5] Eliassen F, Goebel V, et al. Next Generation Middleware: Requirements, Architecture, and Prototypes[C] // Proceedings of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems. December 1999: 60-66

[6] Milner R. Communicating and Mobile Systems: The Pi-Calculus [D]. Cambridge University Press, 1999

[7] Oquendo F. π-ADL: An architecture description language based on the higher-order typed pi-calculus for specifying dynamic and mobile software architectures[J]. ACM Software Engineering Notes, 2004, 29(4): 1-14

[8] 李长云，李赣生，何频捷. 一种形式化的动态体系结构描述语言[J]. 软件学报，2006，6：1349-1359

(上接第 94 页)

参 考 文 献

[1] Rana K. A new approach of admission control for TCP flows [C]// 2nd IEEE International Conference on Information and Communication Technologies: from Theory to Applications (ICTTA '06). Damascus, April 2006, 2: 3262-3268

[2] Jiang Y, Emstad P J, et al. Measurement-based admission control for a flow-aware network [C]// Next Generation Internet Networks. Rome, April 2005: 318-325

[3] 邱恭安，张顺颐，王攀. 基于动态流感知的接纳控制策略研究[J]. 南京邮电大学学报，2008，28(3)：11-16

[4] 邱恭安，张顺颐. 决策域流感知接纳控制[J]. 应用科学学报，2008，26(1)：55-60

[5] Leonard B, Mimoza D, et al. A CAC scheme for multimedia applications based on fuzzy logic [C]// 19th International Conference on Advanced Information Networking and Applications. Taipei, March 2005, 1: 473-478

[6] Oueslati S, Roberts J. A new direction for quality of service: flow-aware networking [Z]. Next Generation Internet Networks, Roma, Italy, April 2005: 226-232

[7] 陆传赉. 排队论[M]. 北京：北京邮电大学出版社，2003：168-176