Web Service 核心协议与实施研究

Web Service core protocols and implementation

杨德仁　薛 梅　顾君忠　（上海 华东师范大学 计算机应用研究所　200062）

摘要：本文首先介绍了 Web Services、中间件和 SOA。着重分析了 Web Services 的主要协议 WSDL 和 SOAP，及其典型实施 Axis，最后指出了 Web Services 的新规范 WS – Addressing 及其带来的挑战。

关键词：Web 服务 面向服务的体系结构 协议 Axis WS – Addressing

1　引言

Web Services（简称 WS）是一种新型分布式计算模式，它是解决日益增长的互操作、企业应用集成（EAI）、B2B 等需求的良好方案，因而享有业界的广泛支持。

WS 是语言和平台独立的基础设施，用于在互联网上松散耦合的、相互可操作的应用之间的通信：所谓语言和平台独立指规范和实施相分离，松散耦合指基于消息的同步或异步交互，相互可操作指基于标准的，在互联网上指没有中央控制机制。实际上 WS 是通过网络可以访问的程序，它通过在 HTTP/HTTPS 上接收/传输 SOAP 消息而向外界呈现功能。

2　WS 及其相关概念

2.1　WS 与中间件

WS 技术中立，实际上也是一种中间件，属于 App2App 或 Middleware2Middleware 中间件。其特点有：通信机制是消息传递、可用多种语言实现、应用之间的连接是偶尔连接。WS 的支撑标准如 HTTP 和 XML 可以满足企业内与企业之间应用穿越防火墙和各种中间件等需求。

2.2　面向服务的体系结构、服务与 WS

WS 是实现服务之间的连接技术。这种服务的结合就构成了面向服务的体系结构即 SOA。SOA 是一个按需连接资源的系统，其中资源是网络中的可按标准化方式的独立服务。与传统系统体系结构相比，SOA 提供了更加灵活的松散耦合机制。

SOA[1] 本质上是服务的集合，这些服务必然相互通信，通信要涉及到简单的数据传递或多个协调的活动，这就需要考虑连接服务的方法。SOA 是一种 IT 体系结构，把组件、业务功能、事务和过程呈现为定义良好的服务，旨在允许服务复用和应用与服务的快速集成。因此，可以把 SOA 理解为 WS 的高层视图。SOA 的关键组件有服务、消息、动态发现和 WS。

服务：在 SOA 中，服务是用 WS 连接起来的，因此服务是连接的端点。服务是一个定义良好的、自我包含的、不依赖于环境或其他服务的功能单元。从服务提供者和服务请求者的观点看，服务是一套整体化的行为，具有 3 个特性：服务接口是平台独立的，任何客户端都可以享用服务；依靠目录服务动态发现和调用服务；服务是自包含的，服务能自我维护状态。

消息：服务提供者和消费者通过消息通信。服务呈现接口，而接口又定义了服务的行为、接收和反馈的消息。接口是平台和语言独立的，用于定义消息的技术应该不属于任何平台或语言，因此，消息用遵循 XML 模式的 XML 文档构建。XML 提供了消息机制需要的功能、粒度和可扩展性。

动态发现：SOA 由 3 个核心组件即服务提供者、服务消费者、目录服务组成。目录服务是提供者和消费者之间的中介。提供者用目录服务注册，而消费者查询目录服务去发现服务提供者。如图 1 所示。

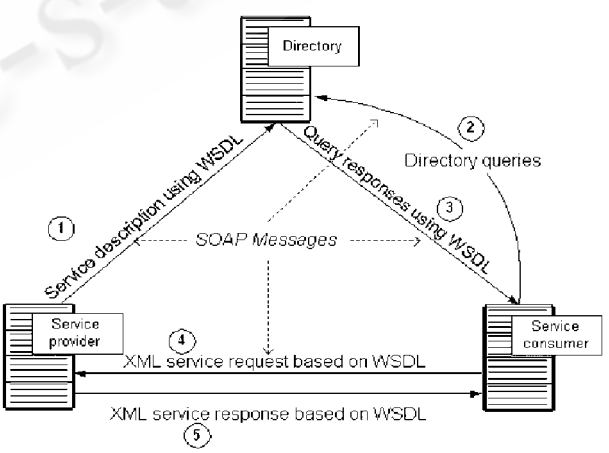

图 1

WS 晚于 SOA 问世，但因建于著名的平台独立协议（如 HTTP、XML、UDDI、WSDL 和 SOAP）之上。WS 满足了 SOA 的主

要需求,是 SOA 的主要实施。它用 UDDI、WSDL 和 SOAP 动态发现和调用服务,用 XML 实现了平台独立的接口,用 HTTP 实现了互操作性。

W3C 对 WS 的定义是:WS 是一个软件应用,由 URI 识别,其接口和绑定用 XML 文件定义、描述和发现。WS 用 XML 消息与其他软件直接交互。

WS 有以下八个主要特征:

WS 是自包含的。在客户端,只需要支持 XML 和 HTTP 的编程语言;在服务器,只需要 Web server 和 servlet 引擎。

WS 是自描述的。客户端和服务器只用关心请求和响应消息的格式和内容。消息格式的定义与消息同在,不用额外的元数据或代码来说明。

WS 是模块化的。WS 是通过 Web 部署和提供对业务功能访问的技术,J2EE、CORBA 和其他标准是实施 WS 的技术。

WS 能跨 Web 发布、定位和调用。用到的标准有 SOAP、WSDL 和 UDDI。

WS 是语言中立的和互操作的。提供者和请求者之间的交互是语言和平台独立的,只需要用 WSDL 定义接口和描述服务。

WS 是开放的和基于标准的。XML 和 HTTP 是 WS 的技术标准。大部分 Web service 技术用开源项目构建,独立和互操作性是其现实目标。

Web 是动态的。用 UDDI 和 WSDL,Web service 的描述和发现能自动进行。

WS 是可以组合的。用工作流技术或调用低层 WS,简单 WS 能聚合成高级 WS。

3 WS 体系结构

WS 体系结构的协议栈是:

传输协议 HTTP/HTTPS;

数据编码协议 SOAP 和 XML 模式;

接口描述协议 WSDL;

服务描述和发现协议 UDDI;

安全协议 WS – Security、XML – Signature 和 XML – Encryption 等。

下面主要介绍 SOAP 和 WSDL 两个主要协议。

3.1 WSDL

用结构化方式描述通信协议和消息格式是 WS 成功的关键。WSDL 把网络服务描述成能交换消息的通信端点的集合体。WSDL 为分布式系统提供了技术文档,并为自动处理应用通信提供了机制。与 IDL 一样,WSDL 是一种接口语言,但具有更高的灵活性和扩展性。

如图 2 所示,WSDL 文档把服务定义为网络端点或端口的集合体,把端点的抽象定义和消息与具体网络部署或数据格式绑定分离,从而实现了抽象定义如消息和端口类型的复用。

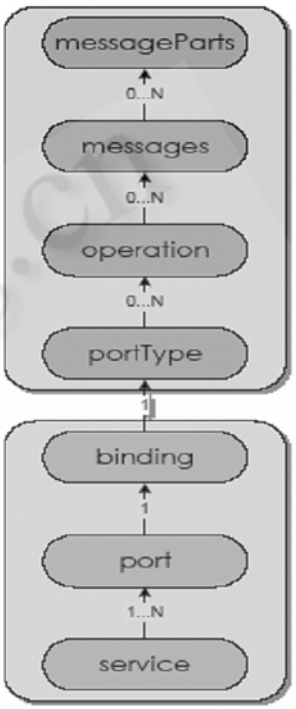

图 2

WSDL 定义了一个通用绑定机制,用于把某个协议或数据格式(结构)与一个抽象消息、操作或端点联系起来。这样就允许抽象定义的复用。用于某个端口类型的具体协议和数据格式即构成了一个可复用的绑定。

WSDL[2] 文件在定义服务时要用到的关键元素如下:

- Types:使用某些类型系统(如 XSD)定义数据类型的容器。
- Message:交换的、数据的、抽象的和类型化的定义。
- Operation:服务支持的、行为的抽象描述。
- Port Type:一个或多个端点支持的、操作的抽象集。
- Binding:用于某个端口类型的具体协议和数据格式规范。
- Port:被定义为绑定和网络地址的结合的单个端点。
- Service:相关端点的集合体。

WSDL 没有引进新的类型定义语言,WSDL 承认需要丰富的类型系统以便描述消息格式,并把 XML 模式规范(XSD)作为规范的类型系统。然而,WSDL 也可扩展其他类型定义语言。

3.2 SOAP

简单对象访问协议[3]（SOAP）是分布式环境中交换信息的协议。SOAP 定义了传递 XML – encoded 数据时的统一方式。它包括四个部分：

SOAP 封装定义了描述消息中的内容是什么，是谁发送的，谁应当接受和处理，以及如何处理的框架；

SOAP 编码规则表示应用程序需要使用的数据类型的实例；

SOAP RPC 表示远程过程调用和应答的协定；

SOAP 绑定则使用底层协议交换信息。

SOAP 封装模型如图 3 所示。

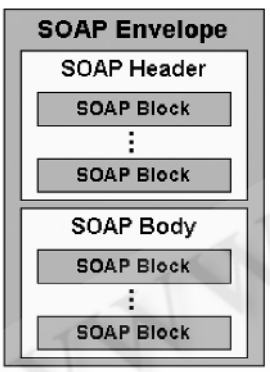

图 3

SOAP 是一个开放协议，使用 HTTP 传送 XML，XML 是一个开放、健全、有语义的信息机制，而 HTTP 可避免许多防火墙问题，从而使 SOAP 得到了广泛的应用。

在 SOAP1. 2 规范中，SOAP 定义甚至没有涉及对象，SOAP 是一种在分散，分布环境中用来交换结构化信息的轻量级协议。SOAP 使用 XML 技术来定义一个可扩展的消息框架，这个消息框架提供了一个能在各种下层协议上进行交换的消息构造。框架被设计成独立于任何特定的编程模型和其他实现的具体语义。其特点是：简单性、可扩展（松散耦合）、能使用多种下层网络协议、独立于编程模型。

SOAP 的应用环境（context）如图 4 所示。

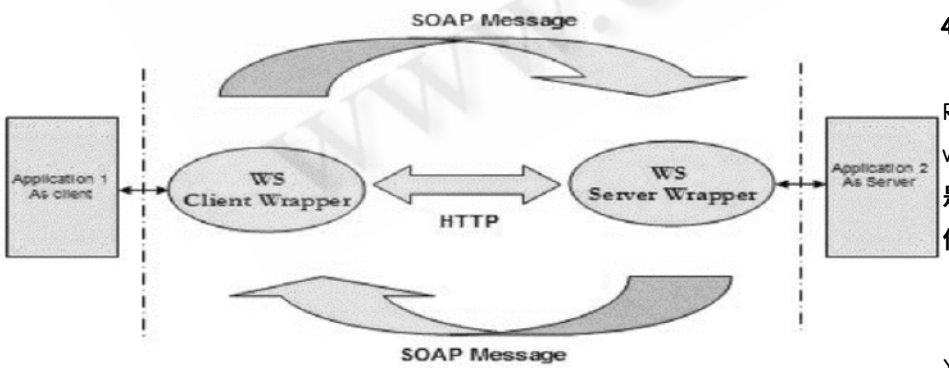

图 4

4 WS 的一种典型实现——Axis

WS 的实施主要包括编程模型、部署模型和消息引擎。编程模型解决如何编写访问 WS 的客户端程序、如何编写服务实施和如何处理 SOAP 规范的其他部分如头和附件等问题。部署模型则要实现服务实施和 SOAP 消息的部署描述符之间的映射、参数的类型映射等。消息引擎则要接收 SOAP 消息，调用服务实施并把 JAVA 类型映射为 XML。

Axis.[4]（Apache eXtensible Interaction System）是一个优秀的开源 SOAP 消息处理引擎，具有灵活的和可扩展的体系结构。它基于 JAX – RPC 客户端系统和 JAX – RPC 服务器系统（基于 Servlet）编程模型。如图 5 所示，Axis 有 6 个主要组件：

Axis 引擎：SOAP 处理器的主要入口点

Handlers：Axis 内的构件块，把 Axis 与后端连接起来

Chain：Handlers 的有序集合

Transports：SOAP 消息流入和流出的机制

Deployment/Configuration：描述 WS 如何通过 Axis 可用

Serializers/DeSerializers：把本地数据类型转换成 XML 和后端的代码

4.1 消息引擎简介

Axis SOAP 引擎提供了产生和解析 SOAP 消息的类，部署、管理和运行具有 SOAP 服务的服务器端基础设施，调用 SOAP 服务的客户端 API。

SOAP 引擎的处理顺序是：

从传输层接收消息、检查 SOAP 的语义、处理 SOAP 头、Deserialize 消息、把消息路由给服务、服务 Serialize 响应、处理响应的 SOAP 头、最后把响应送回到传输层。

4.2 JAX – RPC 编程模型

Axis 支持 JAX – RPC，JAX – RPC 是用于 WS 编程模型的新 Java 标准。设计 JAX – RPC 的目的是完全实现代码的可移植性即任何客户端和服务实施在支持 JAX – RPC 的软件之间是可移植的。

所谓 JAX – RPC 是指基于 XML 的远程程序调用的 Java API，它定义了 Java types 和 XML types 之间的映射，后两种类型旨在隐藏 XML 的细节并提供了流行的

方法调用范例。

JAX – RPC 主要包括:JAX – RPC 客户端系统、(基于 Servlet 的) JAX – RPC 服务器系统和 SAAJ 实施。JAX – RPC 构建于 SAAJ(Soap Attachments API for Java)之上。它允许 JAX – RPC 客户端与按各种语言编码并被部署在不同平台上的 Web service 通信。

JAX – RPC 为调用方式、客户端产生方法、参数方式、WSDL/XML 与 Java 之间的类型映射、客户端 APIs 调用 WS 提供了规范:

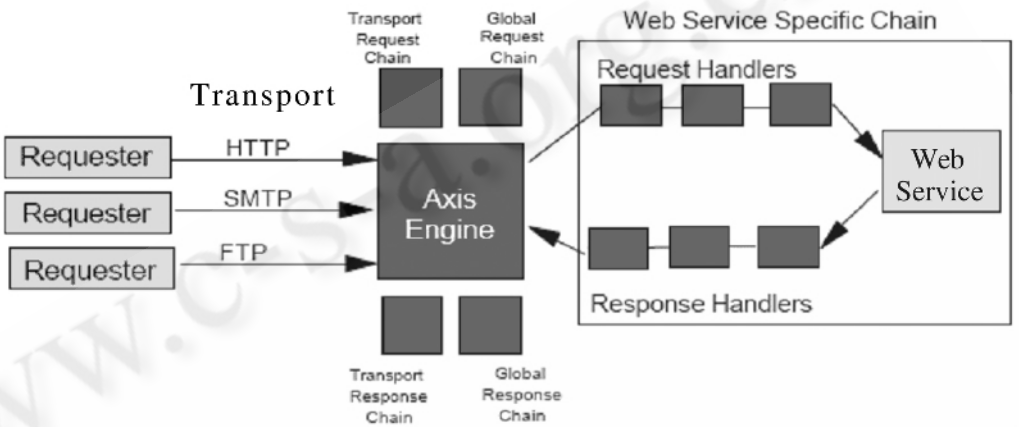

图 5

JAX – RPC 有三种调用方式,分别是同步请求 – 响应、单向 RPC 方式和非阻塞 RPC 方式。JAX – RPC 客户端产生方法有静态(从 WSDL 产生的) stub 调用、动态 Proxy 调用和动态(同步/异步) DII 调用。基于 JAX – RPC 的 WS 调用使用 pass – by – copy 参数传递语义,支持的参数类型有:IN、OUT 和 IN OUT 三种。WSDL/XML 与 Java 的类型映射、可扩展的 Type 映射、SOAP 绑定映射。SOAP 操作库为同步和异步发送和接收 SOAP 消息(可带附件)提供了一种模式。Servlet 容器服务端点是说明接收 SOAP 消息的 servlets 的标准方法。

访问 Stub 或 Proxy 的两种方法:即容器管理访问:客户端代码不知道目标 WS 的 URL,而只有 SEI 或 WSDL 的 PortType 和 bindings;客户端管理访问:客户端知道服务的 WSDL 完整描述(含 Port),即知道目标 WS 的确切 URL。

编程实例可以参考 Axis 技术文档[4]。

5 WS – Addressing 及其挑战

最近,业界提出的 Web 服务寻址(WS – Addressing)规范[6],是标准化指定 Web 服务的位置的方法。实际上,Web 服务寻址(WS – Addressing)只包括两个新概念即端点引用(endpoint reference,EPR)和 SOAP 结构的消息信息(message information,MI)头。但这个规范具有从根本上改变 SOAP 处理模型的潜力。W3C 已经提出了基于这种规范的新模型即 WS – Resource Framework,这个新模型也将改变网格计算的实施标准 Globus Toolkits[7]。这些对 WS 协议及其实施、网格服务标准及其实施都提出了新的挑战。

参考文献

1 Sayed Hashimi, Service – Oriented Architecture Explained, http://www.ondotnet.com/pub/a/dotnet/2003/08/18/soa_explained.html.

2 W3C, Web Services Architecture, http://www.w3.org/TR/2003/WD – ws – arch – 20030808/ .

3 W3C, SOAP Version 1.2, http://www.w3.org/TR/soap12 – part1/.

4 Apache, Web Services – Axis, http://ws.apache.org/axis/.

5 IBM Software Group, Apache Axis and JAX – RPC, www.cs.unc.edu/Courses/comp190/docs/ lessons – others/ws_axis_kyle_brown/02 – AxisBasic.pdf.

6 Doug Davis, Web 服务寻址(WS – Addressing)对 SOAP 的隐式影响, http://www – 900.ibm.com/developerWorks/cn/webservices/ws – address.shtml.

7 Jarek Gawor, Sam Meder, GT4 WS Java Core Design, http://www – unix.globus.org/toolkit/docs/development/wsrf/3.9.1/WSRFDesign.pdf, May 24, 2004.