

# 基于 SOAP 的 KQML 语言通信的研究与实现<sup>①</sup>

## Research and Implementation of KQML Communication Based on SOAP

马建红 王万森 (首都师范大学信息工程学院 100037)

卫权岗 (河南新乡市气象局 453000)

**摘要:** SOAP 作为一种新的与平台无关的通信协议,可以实现异构系统间信息的交换,使得各种孤立系统的通信变得容易。本文在分析 SOAP、Web Service 等技术的基础上,结合 KQML 语言提出了一种基于 SOAP 协议的 KQML 语言通信实现方法。

**关键词:** SOAP Web Service KQML 通信

### 1 引言

Agent<sup>[1,2]</sup> 技术已被广泛用于人工智能和 Internet 领域的研究活动中。人们期望 Agent 可以如同人在网上进行合作。Agent 要在网上进行合作,首先必须具有在网上进行通信的能力,因而异构多 Agent 系统 (Multi-agent System, MAS) 之间的通信方法成为了人们研究的一个热点。在 Agent 的通信研究方面,有 KQML、FIPA ACL、AgentTalk 等 Agent 通信语言问世。很多组织成功的基于 KQML 或 ACL 构建自己的多 Agent 系统。国内的研究大都是在 KQML 或 ACL 的基础上,实现了 Agent 或者 Mobile Agent 的通信模型<sup>[3,4]</sup>。但是,当前的分布式通信协议 DCOM / COM、CORBA 和 Java RMI 要求服务器端和客户端有明确的同类型、同构架的对等协议,从而使基于它们构建的 MAS 间形成了通信障碍,而 SOAP 是基于 XML 和 HTTP 的分布式对象的通信协议,它不仅可以包容这些协议,实现这些协议间的互通,而且简单清晰、有优秀的开放集成的能力。基于 SOAP 的 Web Service 因为 SOAP 的跨平台的信息表示和数据交换的特点而满足互操作性、普遍性和低进入屏障,使得各种孤立系统的通信变得容易。本文从 SOAP、KQML 等相关标准和协议出发,提出了一种基于 XML 的 SOAP 协议实现 KQML 通信的方法。

### 2 简单对象访问协议 SOAP

SOAP<sup>[5]</sup> (Simple Object Access Protocol) 是一种基

于 XML 的与平台无关的通信协议。XML (Extensible Markup Language) 是 W3C 组织在 1996 年设计和定义的一种跨平台的数据交换规范,已经成为被广泛接受的数据编码和数据处理标准。当前最新的 SOAP 版本是 W3C 于 2003 年 6 月 24 日发表的 SOAP 规范 1.2 版。SOAP 为在一个松散的分布式环境中使用 XML 对等地交换结构化和类型化的信息提供了一个简单的轻量级机制。它把成熟的基于 HTTP 的 Web 技术与 XML 的灵活性和扩展性组合在一起,使现有的软件不论基于什么样的编程模型都可以通过 Internet 通信。SOAP 本身并不定义任何应用程序语义,如编程模型或特定语义实现,而只定义一种简单的机制,通过一个模块化的包装模型和对模块中特定格式编码的数据重编码机制来表示应用语义。SOAP 的这项能力使得它可被很多类型的系统用于从消息系统到 PRC (Remote Procedure Call) 的延伸。SOAP 本身没有定义任何底层的传输协议,被“默认”绑定到 HTTP 协议,因此 SOAP 是相当强大和自由的。SOAP 是跨平台的纯消息协议,通过 SOAP 编码规则,可以把任何与应用程序相关的数据编码成 XML 文件,并封装成 SOAP 消息,使用基于 TCP/IP 的应用协议 HTTP、SMTP、FTP、JMS 或自定义的 P2P 等,进行传输,可以与现有通信技术最大程度地兼容。

#### 2.1 SOAP 协议的内容

SOAP 协议主要由 4 部分组成: SOAP 封装 (SOAP Envelop), 定义了整体 SOAP 消息的表示框架,可用于

<sup>①</sup> 基金项目:北京市教委科技项目 (2002kj098)

表示 SOAP 消息的内容、发送者和接收者; SOAP 编码规则 ( SOAP Encoding Rules ), 定义了一个编序机制用于交换应用程序定义的数据类型实例; SOAP RPC 表示 ( SOAP RPC Representation ) 定义了一个用于表示远程调用和响应的约定; SOAP 绑定 ( SOAP Binding ) 定义了一个使用底层传输协议来完成在节点间交换 SOAP 封装的约定。这 4 部分在功能上是正交的, 且彼此独立的。

SOAP 消息 ( message ) 是在对等的 SOAP 节点 ( SOAP nodes ) 间通信的基本单位。SOAP 消息是由一个强制的 SOAP Envelope、一个可选的 SOAP Header 和一个强制的 SOAP Body 组成的 XML 文档: SOAP Envelope, Envelope 是表示该消息的 XML 文档的顶级元素, 必须作为第一个元素出现在文档中; SOAP Header, Header 用于为 SOAP 消息添加特征, 可用来实现认证, 路由等; SOAP Body, Body 为该消息的最终接收者提供了一个容器承载传输的数据。Body 元素的典型应用包括 PRC 以及出错报告等。

和 SOAP1. 1 不同的是, SOAP1. 2 规范不允许 < body > 元素后再出现元素。

## 2.2 SOAP 与其他通信协议的比较

在 SOAP 出现前的分布式通信协议如 OMG 组织制定的公共对象请求代理体系结构 ( CORBA ), 微软公司提出的分布式组件对象模型 ( COM/DCOM ), 以及 SUN 公司提出的远程方法调用 ( RMI ), 这些协议都可以用自己的方式解决问题, 但没有一种协议可以相互兼容, 它们的优越性只是在单一厂商条件下才能得以充分的体现, 这意味着基于 CORBA 和 DCOM 的解决方案必须依赖于单一的应用程序, 又因为每个系统使用不同的二进制编码, 系统间的互操作就很难实现。虽然 DCOM 和 CORBA 适合于服务器与服务器间的通信, 但是对于客户端与服务器之间的通信十分脆弱, 尤其当客户程序分布在 Internet 上更是如此。虽然 Java 应用程序可以使用 RMI 与 CORBA 连接, 但是与 DCOM 却无法通信。SOAP 的出现为这个领域带来了结合统一的可能性, 它不仅可以把现有的 CORBA 或 COM 应用变成可以基于 Internet 调用的服务, 并且可以把 CORBA 和 COM 整合起来, 成为二者之间的通信桥梁。

由于 SOAP 一般是基于 HTTP 进行传输使得它对防火墙比其他协议更容易发挥作用。防火墙绝大多数

都对 80 端口开放, 而该端口正是 HTTP 传输网页的端口, 所以 SOAP 可以穿过防火墙。传统方法使用 CORBA 或 COM 通信时常有不知道或不常用的端口被使用, 防火墙不允许其通过使得通信失败。

总之, SOAP 和其他通信协议的关系并不是竞争, 而是相互协作的。SOAP 自身并不能代替组件技术它仍然需要 CORBA 或 COM 对象在服务器端进行真正的处理, SOAP 仅仅定义了它们之间进行互操作的一种标准。

## 3 Web Service

在 SOAP/HTTP 方式下, 服务器端处理 SOAP 请求的程序叫 Web Service<sup>[6]</sup>。Web Service 是自包含、组件化的分布式应用与服务, 这些应用与服务基于一组通用的标准与协议, 其中包括 XML、SOAP、WSDL ( Web Service Description Language, Web Service 描述语言 ) 和 UDDI ( Universal Description 、 Discovery and Integration, 统一描述、发现和集成协议 )。WSDL 是一个描述接口、语义及 Web 服务为了响应请求需要经常处理的工作的 XML 文档。UDDI 是一套面向 Web Service 的信息注册中心的实现标准和规范, 利用 SOAP 消息机制 ( 标准 XML/HTTP ) 来发布、编辑、浏览及查找注册信息。Web Service 主要解决了异构平台上不同系统之间互相通信和共享数据的问题。根据图 1 所示的结构图显示了这三种技术的工作过程。

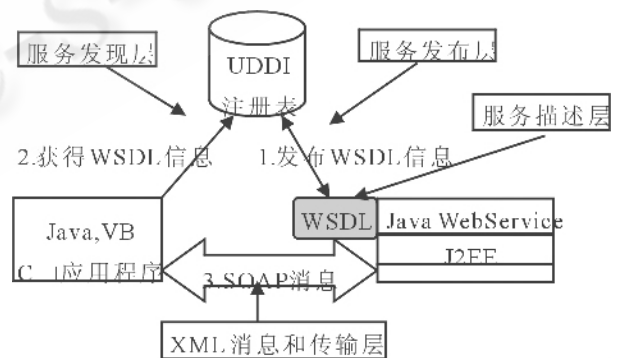


图 1 Web Service 技术结构

在图 1 中, Web Service 提供者首先使用 UDDI 目录注册自己提供的服务, 之后 Web Service 请求者通过查询 UDDI 目录来定位自己需要的服务, UDDI ( Web

Service 代理) 返回符合条件的服务的 WSDL, 请求者根据该服务的 WSDL 生成 SOAP 消息, 并按 SOAP/XML 格式调用 Web Service, 最后运行结果从提供者反馈给请求者。

在 SOAP 应用中, XML、SOAP、WSDL 都是独立于平台的, 而最后的本地映射则与平台有关, Web Service 以此实现跨平台的可互操作性。又因为 Web Service 具有跨防火墙通信的能力, 更便于在 Web 环境下, 实现松散耦合的、跨平台的, 与语言无关的, 与特定接口无关的异构系统间的通信。

## 4 基于 SOAP 协议的 KQML 消息表示

### 4.1 知识查询操纵语言 KQML

由美国 DARPA 知识共享组织的外部接口工作组开发的知识查询操纵语言 (Knowledge Query and Manipulation Language, KQML) 是一种用于软件 Agent 之间的通信语言, 目前已经成了 Agent 通信语言的实际标准。KQML 提供了一系列消息 (performative) 类型实现 Agent 间信息的交流和知识的共享。KQML 规定了消息格式和消息传送系统, 为多 Agent 系统通信和协议提供了一种通用框架。

概念上, 我们可以把一条 KQML 消息分为三层: 通信层、消息层和内容层。全部技术通信参数都在通信层规定, 消息层规定与消息有关的言语行为的类型, 内容层规定消息内容。习惯上, 一条 KQML 行为原语也称为一条消息, 典型的 KQML 消息例如, Agent A 发送一个动作表达式到 Agent B:

```
(ask - all
  : language Prolog
  : ontology foo
  : sender A
  : receiver B
  : in - reply - to id0
  : reply - with id1
  : content " bar(x,y) ")
```

它想知道内容 bar(x,y) 是否为真。

其中 " : sender, : receiver, : in - reply - to, : reply - with" 构成了通信层; "ask - all, : language, : ontology" 构成了消息层; " : content" 为内容层。

### 4.2 基于 SOAP 的 Agent 通信消息结构

虽然 KQML 是第一个使用的 Agent 通信语言, 经过多年的使用和检验, 但还是有许多不足之处: 不同的 KQML 系统不能互操作, 没有一个固定的规范标准去实现通信机制。为解决这一问题, 我们在充分利用 Web Service 技术的基础上运用了一种基于 SOAP 协议的 KQML 通信实现方法。在此方法中, 用 XML 来表示 KQML 语言的执行原语, 并把这些执行原语放在 SOAP 协议的 Body 之中, 最后通过 SOAP 协议在 Agents 之间进行 KQML 通信。由于 XML 是基于数据内容的而不是基于数据结构的, 同时目前流行的编程语言, 比如 Java 等都支持 XML 解析器, 因此在 KQML 通信实现中不需要定义自己的 KQML 解析器。这种方法的消息结构由三层组成: SOAP 层、KQML 层和 Content 层, 如图 2 所示。

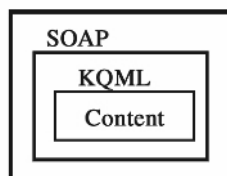


图 2 基于 SOAP 协议的消息结构

KQML 消息 ask - all 的例子如第 4.1 节, Agent A 使用 SOAP 协议向 Agent B 发送一个消息, 该消息用 XML 表示, 并封装到 SOAP 协议的 Body 中。下面给出绑定到 HTTP 的 SOAP 消息的实现。

POST /StockQuote HTTP/1.

```
1 //消息最外层 SOAP 层
Host: www.stockquotesever.com
Content - Type: text/xml; charset = " utf - 8"
Content - Length: nnnn
SOAPAction: " Some - URI"
<? xml Version = "1.0" encoding = " UTF - 8" ? >
< SOAP - ENV: Envelope
  xmlns: SOAP - ENV = " http://schemas.xmlsoap.org/soap/envelope/"
  SOAP - ENV: encodingStyle = " http://schemas.xmlsoap.org/soap/encoding/" >
  < SOAP - ENV: Body >
    < message xmlns: m = " Some - URI" > //消息中间层 KQML 层
      < operation value = " ask - all" / >
      < language name = " Prolog" / >
      < ontology name = " foo" / >
      < sender name = " A " / >
      < receiver name = " B" / >
      < in - reply - to value = " id0" >
```

```

<reply with value = " id1" / >
  <content > //消息最内层 Content 层
    <item name = " bar" >
      <para value = " a" / >
      <para value = " b" / >
    </item >
  </content > //消息最内层 Content 层
</message > //消息中间层 KQML 层
</SOAP - ENV: Body >
</SOAP - ENV: Envelope > //消息最外层 SOAP 层

```

Agent B 接到 Agent A 发来的消息后进行处理, 同样 Agent B 通过使用 SOAP 协议向 Agent A 发送一个回复。用上述的 SOAP 消息可以实现 Agent 之间的通信。

### 4.3 基于 Web Service 的 Agent 通信

Web Service 是建立可互操作的分布式应用程序的新平台。当 SOAP 客户端以 HTTP 传送一份请求消息至目标服务器, 该 Web 服务器会拦截该消息。Web Service 都是放在 Web 服务器后面的, Web 服务器会把该 SOAP 请求消息转发给 Web Service 请求处理器。Web Service 请求处理器解译该 SOAP 请求消息, 然后将客户所请求的过程名及参数映射为本地的在它的管辖范围的过程调用, 然后再生成相应的 SOAP 应答。Web 服务器得到 SOAP 应答后, 再通过 HTTP 应答的方式把它送回到客户端。Agents 通过 Web Service 进行通信的过程如图 3 所示。

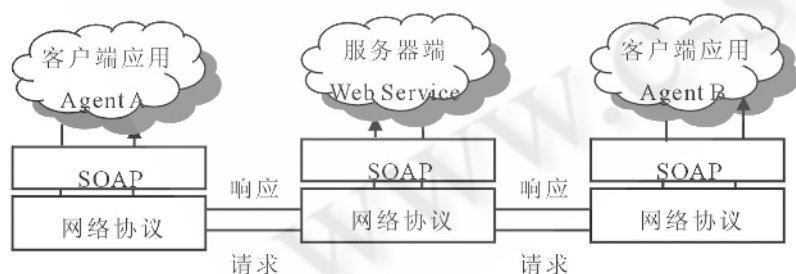


图 3 基于 Web Service 的 Agent 通信

客户端 Agent A 与客户端 Agent B 进行通信时, Agent A 使用 SOAP 协议用 XML 封装请求消息后通过 HTTP 传送到服务器端, 服务器端监听程序收到 SOAP 请求后解析出请求消息数据, 执行相应的服务请求, 把消息传送到客户端 Agent B, Agent B 接收到消息后进

行处理, 若需回复同样发送请求消息到服务器端, 服务器端将操作的结果数据同样以 SOAP 封装后返回给 Agent A。

## 5 结束语

采用基于 XML 的 SOAP 协议来实现 Agents 之间的 KQML 通信, 解决了跨平台的多 Agent 系统互连的接口问题。由于 SOAP 能够被应用于从消息传递到 RPC 的各种系统, 使得我们完全能实现异构平台间的组件调用和通讯, 有利于构建开放集成性的分布系统。SOAP 作为一种简单轻量的协议, 使系统的扩展和维护的复杂性降低。随着 SOAP 技术的日益成熟, 其优越的跨平台特性必将使其在基于 Web 异构分布式系统的服务构建方面得到更加广泛的应用。我们下一步的研究将关注 SOAP 消息的安全性, SOAP 的事务处理等方面。

## 参考文献

- 1 Nwana H S. Software agents: an overview[J]. The Knowledge Engineering Review, 1996, 11 (3) : 205 ~ 244.
- 2 Wooldridge M , Jennings N R. Intelligent agents: theory and practice[J]. The Knowledge Engineering Review. 1995, 10 (2) : 115 ~ 152.
- 3 Cao Hua2gong, Cao Q iang. A meeting scheduler system based on multiti - agent system[J]. Mini - Micro System. 2000. 21 (12) : 1312 ~ 1315.
- 4 Li Hong - bing, Ai Zao - yang, Chen Shi - fu. A framework for Java - based agent application development environment[J]. Mini - Micro System. 2000. 21 (1) : 35 ~ 39.
- 5 W3C Note 08. Simple Object Access Protocol ( SOAP ) 1.1 [ EB/OL ]. <http://www.w3.org/TR/SOAP/> , 2001 - 06.
- 6 Seely S, 杨涛, 杨晓云, 王建桥等译, XML 跨平台 Web Service 开发技术, 机械工业出版社: 2002 - 04 : 127 - 140.
- 7 史忠植, 智能主体及其应用, 北京科学出版社, 2001。