第 34 卷 第 4 期 计 算 机 工 程 2008 年 2 月
Vol.34 No.4 Computer Engineering February 2008
·网络与通信· 文章编号：1000—3428(2008)04—0141—02 文献标识码：A 中图分类号：TP393

基于 Web Services 的 SNMP Agents 集成与管理

张文杰，朱程荣，熊齐邦
（同济大学计算机系，上海 201804）

摘 要：提出一种基于 Web Services 的解决方案，能够无缝地集成 SNMP agents 并对之进行管理。通过现有的 Internet 协议，将对 SNMP agents 的管理操作转化为基于 Web Services 的方法调用，并通过 SOAP 协议传输管理信息。给出了该方案的设计与实现及相应的结果分析。
关键词：Web 服务；无缝集成；SNMP-to-WS 转换

Integration and Management of SNMP Agents Based on Web Services

ZHANG Wen-jie, ZHU Cheng-rong, XIONG Qi-bang
(Dept. of Computer Science and Technology, Tongji University, Shanghai 201804)

【Abstract】 This paper proposes an approach based on Web Services for seamless integration and management of SNMP agents, which is done by converting management operations on SNMP agents into method invocations of Web Services, based on existing Internet protocols and, transferring the management information through SOAP. The design and implementation is presented, and analysis of the results is related.
【Key words】 Web Services; seamless integration; SNMP-to-WS convertion

一直以来，基于SNMP的管理架构占据主导地位，广泛应用于IP网络的管理。但是随着管理要求的不断提高，SNMP的不足逐渐暴露出来。例如：没有标准的格式来存储和处理管理数据，传输大量数据时的性能较差，配置管理不能得到充分彻底的解决等。XML的出现促进了管理技术的发展，它定义了规范的编码和语法规则，使数据的表示具有统一的形式和格式，已成功地运用到了网络管理中[1]，并形成基于XML网络管理的标准[2]。Web Services(WS)以XML为核心。采用基于XML的WSDL(Web Services描述语言)描述服务接口；数据编码和绑定采用基于XML的SOAP实现。

1 Web Services 概述

Web Services是一种新的分布计算技术，由W3C(万维网联盟)制定和发展。Web Services致力于在Web组件上树立起一种开放结构和相关服务，使得符合Web Services标准的应用程序能够具有互操作性；能被各种分布式应用程序通过现有的Internet标准协议调用，具有编程语言无关性、可扩展性和跨平台特性。Web Services采用面向服务的体系结构[3]。服务提供者创建Web Services接口、实现方法的细节并提供接口的描述；Web Services接口通过统一描述、发现和集成(UDDI)注册到服务代理；服务请求者通过UDDI发现所需要的Web Services并绑定到要调用的方法。

Web Services 提供的功能及其技术架构使得它具有诸多特点和优势：(1)Web Services 基于现有的 Internet 标准协议被调用。(2)Web Services 采用基于 XML 的描述和数据表示。(3)语言和平台的无关性。(4)Web Services 在分布式应用的集成上体现出极大的优点，具有很好的可扩展性和可伸缩性。基于 XML 的 SOAP 消息传输和编码机制提供了各种系统和不同应用之间的互操作性。

利用 Web Services 技术的这些优势，容易实现 SNMP 到 Web Services 方法调用的转换，实现对 SNMP agents 的管理。

2 Web Services 在网络管理中的可用性

由于CORBA的流行，TMF和OMG成立了联合域间管理(JIDM)工作组以建立SNMP SMI/OSI SMI GDMO和CORBA IDL间的静态映射和支持通用网关的动态映射，使得CORBA已经运用于TMN的管理[4]。通过表 1 的比较可知，WSDL可以认为是CORBA中的IDL；URI相当于CORBA中的IOR；SOAP相当于CORBA中的GIOP；基于XML编码的消息通过SOAP传输，而SOAP被承载在Internet协议中，如HTTP，此时HTTP即相当于CORBA中的IIOP [4]。

表 1 CORBA 和 Web Services 的相似处

比较项目	CORBA	Web Services
接口描述	IDL	WSDL
操作接口	IOR	URI
访问协议	GIOP	SOAP
操作协议	IIOP	HTTP/TCP/IP
服务发现	Naming and Trading Services	UDDI

既然CORBA已经能够在网络管理中得到应用，那么Web Services也具有可用性，因为它具备 CORBA 的所有特征。此外，通过表 2 的比较，可以看到 Web Services 还具备 CORBA 没有的优势。

表 2 CORBA 和 Web Services 的不同点

比较项目	CORBA	Web Services
客户端和服务端的耦合	紧耦合	松耦合
交互	对象间的交互必须通过 ORB	只需通过现有的 Internet 协议
编写	复杂	容易
部署	繁琐，必须部署 CORBA 平台	简单，只需部署到 Web 服务器

作者简介：张文杰(1983－)，男，硕士研究生，主研方向：信息安全及容错计算；朱程荣，副教授；熊齐邦，教授
收稿日期：2007-02-25 **E-mail**：wjzhangb@163.com

松耦合使 Web Services 具有高可扩展性和可伸缩性；对于调用 Web Services 的应用程序而言没有任何编写的限制；无论是服务端或是客户端的开发、部署和维护都要比 CORBA 容易。Web Services 的运用将会为实现可用性高、可伸缩性强、可扩展性好的管理系统带来可能。

3 设计实现

笔者用一个模块实现 SNMP-to-WS 的转换，作为 SNMP-to-WS 网关，其体系结构如图 1 所示。

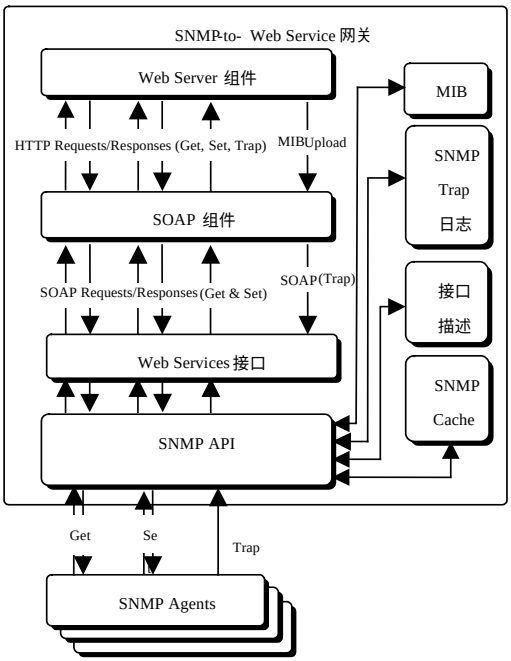

图 1　SNMP-to-WS 网关体系结构

Web Server 组件是一个运行 HTTP Server 的进程实体，它用于 HTTP 请求/响应的处理。SOAP 组件封装了远程方法调用的 XML 消息，以 SOAP-over-HTTP 的方式承载在 HTTP 协议数据单元内。Web Services 接口中包含对 SNMP API 的调用，实现基于 SNMP 的操作。

在某些情况下，agents 中的被管对象可能被多次读取。此时不必每次都从 agents 抽取数据，而是利用网关中的 SNMP Cache 获得多次重复的对象的值，从而减少网络流量，提高性能。

SNMP-to-WS 网关也能够处理 SNMP Trap。agents 所发送的 Trap 被网关记录在 SNMP Trap 日志中。Trap 日志通过两种形式报告给 Manager：(1)一旦有 Trap 就主动推(push)给 Manager，立即通知 Manager 发生故障的点。(2)Manager 采取拉(pull)的方式向网关索取 Trap 日志。这种方式可以成批地获取 SNMP Trap，但 Manager 需要轮巡网关以求日志的更新。

SNMP-to-WS 网关采用 ASP.NET Web Services 实现。SNMP API 部分采用 Adventnet SNMP API for .NET Platform。它已经能够成功运行在 Windows Server 2003/IIS 6.0 平台上。以下给出 Web Services 接口，它集成了 SNMP 的基本操作：

```
public SnmpResponse Get(SnmpRequest GetRequest);
public SnmpResponse GetNext(SnmpRequest GetNextRequest);
public SnmpResponse GetBulk(SnmpRequest GetBulkRequest);
public SnmpResponse Set(SnmpRequest SetRequest);
```

实现这些接口的关键类是 SnmpServiceHelper，SnmpOperation 方法是其中最核心的一个方法。它创建 SNMP API 实例和 SNMP 会话实例，根据参数 request 设定访问 agents 所必需的属性值，然后装载管理信息库，如下：

```
protected SnmpPDU SnmpOperation(SnmpRequest request)
{
SnmpAPI api=new SnmpAPI();
SnmpSession session=new SnmpSession(api);
UDPProtocolOptions sessionOption=new UDPProtocolOptions();
sessionOption.RemoteHost=request.RemoteHost;
sessionOption.RemotePort=request.Port;
mibOperations.loadMibModules(AppSettingsReader.GetValue(request.MibModule, typeof(string)).ToString());
```

接着完成变量绑定。如果是Get、Get-Next和Get-Bulk操作，则只需要将OID填入协议数据单元；如果是Set操作，则必需将待设定的值绑定到对应的OID，即 OID_j=$Value_j$，j=1,2,…,n，再填入协议数据单元。

```
for each(string oid in request.Oid){
if(request.RequestType == SnmpAPI.SET_REQ_MSG){
try{
SnmpVar var=node.getSyntax().createVariable(oid);
  varBind=new SnmpVarBind(snmpOID, var);
}catch{…}
pdu.AddVariableBinding(varBind);
  }else{pdu.AddNull(new SnmpOID(oid));}
}
```

最后发送 SNMP 请求到被管对象。返回值是一个 SNMP 协议数据单元。

```
session.Open();
try{responsePdu=session.SyncSend(pdu);}catch{…}
return responsePdu;
}
```

当结果返回时，Web Services 组件会将其串行化为 XML。

```
response.Value=mibOperations.varBindsToString(responsePdu);
return response;
```

基于 SNMP-to-WS 网关，给出一个实用的 3 层架构管理模型，如图 2 所示。

图 2　一个实用的 3 层架构管理模型

Manager 通过 HTTP 访问 SNMP-to-WS 网关，其位置可以就处在被管设备所在网络中，此时 SNMP-to-WS 网关作为代理；也可以通过网络在远端管理，此时该网关的作用类似 RMON，它监视网络设备并把管理信息收集在 SNMP Cache 和 SNMP Trap 日志中。Manager 可以用拉的方式来访问网关，网关也可以采取推的方式把采集的管理信息发送给 Manager。

4 测试与分析

通过 SNMP 和 Web Services 分别对某网络设备实施 Get 操作，记录从 Get 请求发出到请求响应返回至 Manager 的时间，以比较 SNMP 和 Web Services 的性能。实验的参数如下：

agent	路由器(Quidway S8016)	
MIB	RFC1213-MIB(MIB II)	
OID for 1 MO	1.3.6.1.2.1.1.3.0(sysUpTime)	
OID for NMOs, N=7	1.3.6.1.2.1.1.1.0	1.3.6.1.2.1.1.2.0
	1.3.6.1.2.1.1.3.0	1.3.6.1.2.1.1.4.0
	1.3.6.1.2.1.1.5.0	1.3.6.1.2.1.1.6.0
	1.3.6.1.2.1.1.7.0	
OID for SNMPWalk	1.3.6.1.2.1.2.2.1	

(下转第 145 页)