基于 XML 的网络配置管理的研究与实现方案

俞 斌,熊齐邦
(同济大学 计算机科学与技术系,上海 200331)

摘 要:互联网上标准的管理构架一直是基于简单网络管理协议(SNMP)的。尽管在许多监测任务方面,SNMP 仍然是一个首选的协议标准,但由于其自身的缺陷,对于配置管理它却并不是一个理想的技术。随着 XML 技术的发展和完善,尤其是 XML-RPC 和安全传输协议的结合,能够克服原有缺陷,提供安全、高效的网络配置管理。为了验证新尝试的可行性和有效性,文中描述了如何采用 Netconf 协议和模块化设计的思想,来设计并实现一个完全基于 XML 的配置管理系统。

关键词:配置管理;可扩展标识语言;基于 XML 的远程过程调用;Netconf 协议
中图分类号:TP393.07 **文献标识码**:A **文章编号**:1673-629X(2007)02-0168-04

Design and Implementation of XML-Based Configuration Management for Networks

YU Bin, XIONG Qi-bang
(Computer Science and Technology Department, Tongji University, Shanghai 200331, China)

Abstract: Internet standard management framework has been based on the simple nerwork management protocol (SNMP). Although it is likely that SNMP will remain the protocol of choice for many monitoring tasks, it is not the ideal technology for tasks such as configuration management. With the development and advancement of extensible markup language (XML) technology, especially XML-RPC mechanism, and together with secure transport protocol, it will overcome the shortcomings and provide a secure and efficient network configuration management. To validate this new effort, this paper presents the design and implementation of an XML-based configuration management system based on Netconf and module design.

Key words: configuration management; XML; XML-RPC; Netconf protocol

0 引 言

当前计算机网络的发展特点是:网络规模不断扩大,复杂性不断增加,网络的异构性越来越高。一个网络往往由若干个大大小小的子网组成,集成了多种 OS 平台,包括不同厂家公司的网络设备和通信设备等。如果没有一个高效的网络管理系统对网络进行管理,那么很难保证为广大用户提供令人满意的服务。SNMP 是目前最常用的环境管理协议,提供了从网络上的设备中收集网络管理信息的方法,也为设备向网络管理工作站报告问题和错误提供了方法。它采用 UDP 传送,实现简单,技术成熟,几乎所有的网络设备生产厂家都实现了对 SNMP 的支持。但是 SNMP 在安全可靠性、管理操作效率、交互操作和复杂操作实现上还存在缺陷,不能满足管理需求。

收稿日期:2006-05-19
作者简介:俞 斌(1982-),男,浙江人,硕士研究生,研究方向为计算机网络管理;熊齐邦,教授,硕士生导师,研究方向为计算机网络与分布式系统。

近些年来,随着 XML(可扩展标记语言)技术的发展,基于 XML 技术的网络管理系统变得越来越受人关注[1]。与 XML 相关的各种技术[2]:XML schema, XML Path Language (XPath), XSL (Extensible Style-Sheet Language), XML-RPC[3] 以及 WSDL (Web Services Description Language),都能够应用到网络配置管理中,来解决 SNMP 具有的缺陷。IETF(互联网工程任务组)正在进行对 XML 技术用于网络设备配置管理的一系列标准化和规范化,对应的网络配置简称为 Netconf[4]。Netconf 协议起草不久,管理对象模型建立任务繁重,协议的定义以及完善需要一定的时间,而对象模型建立和协议实现的实践过程也相当的重要,它对 Netconf 由理论转向应用起着至关重要的作用。文中在研究基于 XML 的网络配置管理的工作原理之后,提出了以模块化思想实现的一个实践方案。

1 几个关键技术

文中基于 XML 的网络配置管理的实现方案,是

以 IETF 正在研讨制定的 Netconf 协议模型作为理论基础。Netconf 协议为管理网络设备定义了一套简单的基于 XML－RPC 的机制,通过它能够获得设备的配置数据、状态信息和系统提示,并且能够对配置数据进行更新操作。协议在概念上大致可以分为 4 个层次(如图 1 所示)。

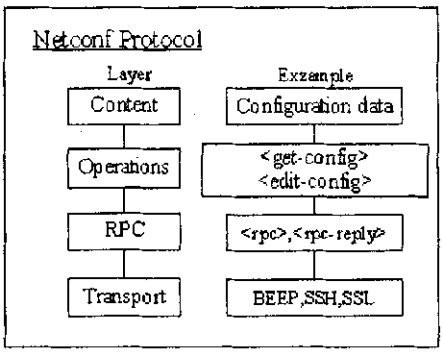

图 1　Netconf 协议层次

把 Server 定义为一台网络设备,而 Client 则是网络管理者所运行的一个程序或脚本,通过建立在可靠连接之上的 RPC 机制来实现 Client 与 Server 之间的通信。

1.1　XML－RPC

XML－RPC 是一套允许运行在不同操作系统、不同环境的程序实现基于 Internet 过程调用的规范和一系列的实现。这种远程过程调用使用 HTTP 作为传输协议,XML 作为传送信息的编码格式。XML－RPC 的定义尽可能地保持了简单,但同时能够传送、处理、返回复杂的数据结构。一个 RPC 系统,必然包括 2 个部分:

(1) rpc client,用来向 rpc server 调用方法,并接收方法的返回数据;

(2) rpc server,用于响应 rpc client 的请求、执行方法,并回送方法执行结果。

rpc client 的工作原理:rpc client 根据 URL 找到 rpcserver→构造命令包,调用 rpc server 上的某个服务的某个方法→接收到 rpc server 的返回,解析响应包,拿出调用的返回结果。

rpcs server 的工作原理:启动一个 webserver(在使用内置的 webserver 的情况下)→注册每个能提供的服务,每个服务对应一个 Handler 类→进入服务监听状态。

XMC－RPC 的工作原理如图 2 所示。

1.2　SSL

SSL[5] 协议是网景公司设计的基于 Web 应用的安全协议,它指定了在应用程序协议(如 HTTP, Telnet 和 FTP 等)和 TCP/IP 协议之间进行数据交换的安全机制,为 TCP/IP 连接提供数据加密、服务器认证以及可选的客户机认证,其协议栈如图 3 所示。

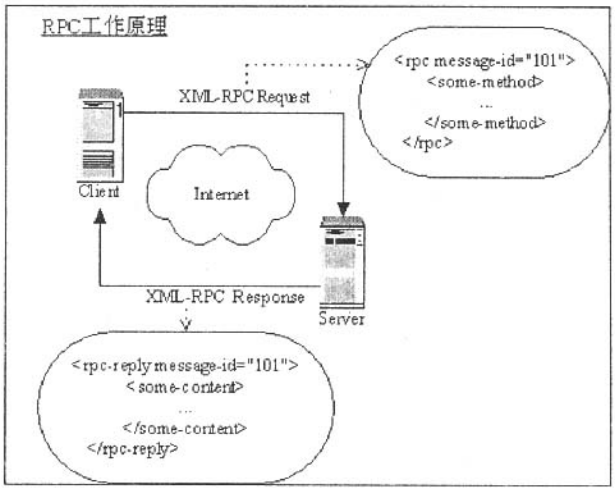

图 2　XML－RPC 工作原理

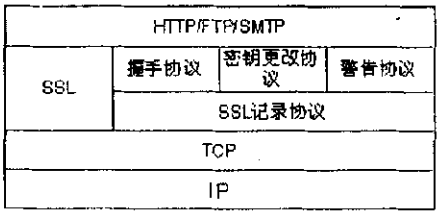

图 3　SSL 协议栈

SSL 协议是由 SSL 记录协议、握手协议、密钥更改协议和告警协议组成,它们共同为应用访问连接提供认证、加密和防篡改功能。作为应用层协议,SSL 使用公开密钥体制和 X.509 数字证书技术保护信息传输的机密性和完整性。SSL 安全功能组件包括三部分:认证(在连接两端对服务器或同时对服务器和客户端进行验证)加密(对通信进行加密,只有经过加密的双方才能交换信息并相互识别);完整性检验(进行信息内容检测,防止被篡改)。保证通信进程安全的关键步骤就是对通信双方进行认证,SSL 握手协议负责这一进程的处理。

2　实现方案

2.1　系统整体构架

采用模块化设计的思想,整个配置系统的结构可以分为三大组成部分(如图 4 所示)运行在 client 端的管理器(Manager),运行在 server 上调用各模块的代理(Agent),以及实际运行管理操作的各个功能模块。事实上管理器本身并不拥有配置数据,而是把从代理传来的 XML 格式的数据以图形树的方式显示出来,使得操作者能够容易理解和操作。运行在 client 部分的结构则复杂得多,配置任务由不同的模块来完成相应的操作(如负责配置管理 interface 的接口模块,配置管

理 route 的路由模块)。Agent 仅仅提供与远程 Manager 的通信层以及 XML 数据的解析功能,所有的管理操作均由 RPC 调用的并且能够使用的模块完成。

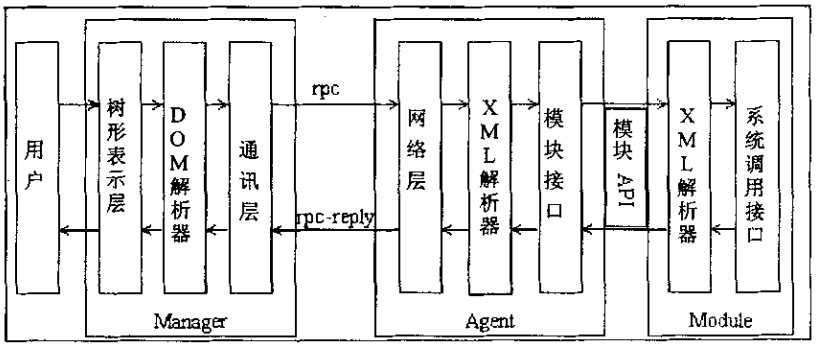

图 4 系统构架

Manager 实现终端用户与远程 Agent 的连接功能,它由三个部分构成:一个树形的表示层,一个 DOM 解析器(用来解析 XML 格式的数据)以及一个通信层(用来与 Agent 之间发送和收取 XML 格式的消息)。当通信层收到消息后,所有的数据都被提交到 DOM 解析器,根据 Netconf 协议中定义的语法结构,来判断消息的有效性。随后,利用从 DefaultMutableTreeNode 继承而来的 Java 类,把转换出来的树形结构的 XML 数据用一棵图形树表示出来。通过这个方式,终端用户只需在图形树上对不同的配置信息进行监视和操作,通过改变一些树节点的属性,来触发 DOM 解析器产生一系列相应的 XML 代码,并由通信层发送到 Agent 调用相应的模块完成配置操作。

同样,Agent 在体系结构中的目的是在 Manager 和各个操作模块之间进行消息的提取和传递的作用。通过网络层,Agent 与 Manager 之间进行获取或者发送 XML 请求。当 Agent 接受到消息之后,XML 解析层对消息进行解析和检查,汇集之后的信息用来定义请求的种类(获取/设置操作等)。与此同时,XML 解析层必须确定由哪个 Module 来完成实现用户的请求。根据消息的 XML 语法标记,RPC 请求被传递到相应的模块(若请求带有 <all/> 标记,则传递给每一个模块)。从这个角度来说,Agent 的各模块接口是用来告诉被调用的模块执行哪些操作。反过来,当被调用的模块完成了指定的操作(或者产生错误信息),由 XML 解析层产生一条 XML 消息,再通过网络通信层传回 Manager。通过使用 SSL,保证传输的安全性和可靠性。下列代码可以概括整个过程:

```
int bytes;
char * answer = Null;
bytes = SSL_read ((SSL *)ssl,buff,sizeof(buff));
buff[bytes] ='/0 ';
get_req (buff);
answer = strdup (set_output ());
SSL_write ((SSL *) ssl,answer,strlen(answer));
```

每个模块的结构十分清晰和类似。正如前面所述,每个模块只完成为它设计的某个范围的功能。由 Agent 来决定由哪个模块执行哪些任务,并向它发送一个 XML 格式的用户请求。根据请求的操作类型(已由 Agent 的 API 判定),模块进行系统调用,来获取或者编辑一个特定设备的属性和状态信息。此外,模块能够对系统操作进行日志记录,它能够通过使用系统日志(Syslog)机制来实现,这样使管理者能够对设备所完成的系统操作进行跟踪记录 通过日志的方式,也能够让其他分析工具来检测 Agent 和模块的行为。

存储模块属性的接口类型可以定义如下:

```
typedef struct {
  char * name;          /* 模块的名字 */
  char * desc;          /* 模块的描述 */
  void * prop;          /* 模块的属性 */
  int nr;               /* 模块描述符的数目 */
  notify_f notify;      /* 模块的 notify()函数 */
  list_f list;          /* 模块的 list_properties()函数 */
  savecfg_f savecfg;    /* 模块的 save_cfg()函数 */
  restcfg_f restcfg;    /* 模块的 rest_cfg()函数 */
  get_request_f get_request;/* 模块的 get_request()函数 */
} net_module_t;
```

2.2 XML 信息流的交互

在 Server 端接收到一个 XML 请求以及回复被发送的过程中,Agent 与它的模块之间发生的交互是基于 XML-RPC 实现配置管理的核心部分。整条 XML 消息被网络层接收后被提交到 XML 解析器,其中一部分消息被提取到 Agent 的接口模块。以一条 XML 请求消息为例:

```
<?xml version="1.0"encoding="UTF-8"?>
<rpc message-id="1">
  <get-config>
    <source>
      <running/>
    </source>
    <config>
      <interface/> <!-被提取到模块接口->
    </config>
  <format>xml</format>
  </get-config>
</rpc>
```

整条消息被网络层接收后,前半部分被 XML 解

析层解析,直到遇到<interface/>,Agent 据此通过模块接口层来判定哪个有效的模块负责处理这个请求,在发现有一个模块也叫 interface 后,请求被提交给这个模块。此时,interface 模块知道要把所有有效接口的属性值列表返回给 Agent(根据<get-config>标记)。随后 interface 模块通过一些系统调用获取当前的相关配置信息。

请求的回复则是一个相逆的过程:当模块获得所有请求的系统配置信息后,再把它们返回到 Agent 的接口模块层,XML 解析层为回复加上"头"和"尾"。回复的主要部分都是由模块自身产生的,Agent 模块接口仅仅负责向 XML 解析层通告新属性结点的加入:

```
<!-- 由 XML 解析层添加 -->
<?xml version="1.0"?>
<!DOCTYPE netconf.dtd>
<rpc-reply message-id="1">
  <config>
    <!-- 由 interface 模块接口传递而来 -->
    <interface>
      <iface>
        <name>lo</name>
        <address>127.0.0.1</address>
        <address6>::4624:240:41:240</address6>
        <broadcast>0.0.0.0</broadcast>
        <netmask>255.0.0.0</netmask>
      </iface>
      <iface>
        <name>eh0</name>
        <address>192.168.0.116</address>
        <address6>::48fc:408:41:240</address6>
        <broadcast>192.168.0.255</broadcast>
        <netmask>255.255.255.0</netmask>
      </iface>
    </interface>
    <!-- 由 XML 解析层添加 -->
  </conifg>
</rpc-reply>
```

回复的消息生成后,再递交给网络层,并发送给远端的 Manager。

3　结束语

使用 XML 技术代表着网络管理协议发展的方向,尤其在设备配置管理等应用上。文中以 IETF 正在研讨的 Netonf 协议为基础,具体设计了一个完全基于 XML 消息格式的网络配置管理方案,并提出了模块化实现的构架。利用 SSL 的传输协议,提高了系统的安全性和可靠性。在将来的工作中,需要进一步加强模块的功能,并根据 SNMP 所使用的 MIB 库的定义,扩充新的模块。通过对新协议的模拟实践,分析协议实际应用的可行性。

参考文献:

[1] Schonwalder J, Pars A, Martin-Flatin J P. On the Future of Internet Management Technologies[J]. IEEE Commun Mag, 2003, 41(10): 90-97.

[2] Choi M J, Hong J W, Ju H T. XML-Based Network Management for IP Networks[J]. ETRI J, 2003, 25(6): 445-463.

[3] Dissanaike S, Wijkman P, Wijkman M. Utilizing XML-RPC or SOAP on an embedded system[C]// Proceedings of the 24th International Conference on Distributed Computing Systems Workshops. [s.l.]: [s.n.], 2004: 438-440.

[4] Enns R. NETCONF Configuration Protocol draft-ietf-netconf-prot-10[S]. [s.l.]: IETF, 2005.

[5] Chou W. Inside SSL: accelerating secure transactions[J]. IT Professional, 2002, 4(5): 37-41.

(上接第 167 页)

参考文献:

[1] Snir M, Otto S, Huss-Lederman S, et al. MPI: The Complete Reference[M]. London: MIT Press, 1996.

[2] OpenMP Standards Board. OpenMP: a Proposed Industry Standard API for Shared Memory Programming [EB/OL]. 1997. http://www.openmp.org/openmp/mp-documents/paper/paper.Html.

[3] Flynn M J, Rudd K W. Parallel architectures[J]. ACM Computing Surveys, 1996, 28(1): 67-70.

[4] McGraw J R, Axelrod T S. Exploiting multiprocessors: Issues and options[C]// In Babb R G. Programming Parallel Processors. MA: Addison-Wesley, 1988: 7-25.

[5] PVM news group. PVM: Parallel Virtual Machine[EB/OL]. 2006-07-19. http://www.csm.ornl.gov/pvm/.

[6] Quinn M J. Parallel Programming in C with MPI and OpenMP[M]. [s.l.]: McGraw-Hill Companies, Inc, 2004.

[7] OpenMP Architecture Review Board. OpenMP 2.5 Draft Specification Released for Public Comments[EB/OL]. 2004-11-04. http://www.openmp.org/.

[8] AN Hong, CHEN Guo-liang. Parallel Programming Models and Languages[J]. Journal of Software, 2002, 13(1): 118-124.

[9] Pitman E B. AMDAHL'S LAW FOR PARALLEL SPEEDUP[EB/OL]. 2000-09-13. http://www.math.buffalo.edu/pitman/courses/cor501/HPC1/node11.html.