一种基于 XML 的主动应用程序描述符 AADX

周悦芝,张尧学,王 勇
(清华大学计算机科学与技术系,北京 100084)

摘 要: 传统网络只能依照预先定义的协议被动地传输数据.随着网络应用的不断扩大,新的用户需求和应用也不断涌现,而新的协议及服务却很难在现行网络上快速实施.主动网络为解决传统网络存在的问题提供了一条新思路.主动网络的主要优点是用户可以根据需要向网络节点中插入程序,以完成定制的网络协议和服务.因此,主动网络的核心功能是按照用户要求识别,传输,加载并运行主动应用程序,以完成用户所需要的计算.本文提出了一种基于 XML 的主动应用程序描述符 AADX(Active Application Descriptor based on XML).该描述符可以用来标识主动程序并为其传输和加载提供基础.它为主动网络提供了一种有效灵活的主动程序识别和分发机制.

关键词: 主动网络; 主动应用程序; XML; 代码分发

中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2002) 12A-2057-04

AADX: A New Active Application Descriptor Based on XML for Active Network

ZHOU Yue-zhi, ZHANG Yao-xue, WANG Yong
(*Dept. of Computer Science & Technology, Tsinghua University, Beijing 100084, China*)

Abstract: Traditional networks can only transport data passively according to the predefined protocols. With the extension of network applications, new requirements are demanded on the network by end users. However, it is difficult to deploy new network protocols and services quickly in traditional networks. Active networking provides a novel way to solve the problems of traditional networks. Programs can be inserted into the nodes of active networks to perform customized computation. Therefore, the core functionality of active networks is to identify, transfer, load and execute active applications to perform the computation required by the end users. This paper presents a new active application descriptor based on XML named as AADX (Active Application Descriptor based on XML), which can be used as a common method of identifying active applications and a basement for transferring and loading. It provides an effective and flexible identification and distribution mechanism of active applications in active networks.

Key words: active networks; active application; XML; code distribution

1 引言

在过去的几十年中,传统网络获得了巨大的成功.主要原因是它通过定义一系列标准协议获得了可扩展性,灵活性,高效性和简单性.但是传统网络是按照一种"端到端"的思想设计的,在网络节点中不关心网络应用程序的状态和需求,只是依照预先定义的网络协议被动地传输数据.然而,随着硬件和软件技术的发展,计算机网络的应用也日益广泛,各种网络协议和应用层出不穷,传统网络被动的统一对待所有数据包的缺点日益突出.比如,基于 Internet 的多媒体应用和其它实时应用,就要求网络提供一定的网络服务质量(Quality-of-Services)保证.为了在传统网络中提供服务质量保证,人们提出了两种服务质量体系结构,集成服务(Integrated Services)[1]和区分服务(Differentiated Services)[2].但是这些服务模型和协议需要所有网络节点的支持,在 Internet 日益庞大的今天,部署和更新协议是一件非常困难的事情.另一方面,这些模型和协议或多或少还与传统网络的"端到端"观点相冲突.比如这些模型和协议,比如 RSVP 协议[3],都要求在网络节点中保持一定的与应用相关的状态.

为了解决传统网络在协议部署中的困难,以快速创建新服务,满足网络应用的各种需求,人们提出了一种新型的网络体系结构,即主动网络[4].主动网络是一种可编程网络,用户可通过编程来控制网络节点的行为,比如资源的分配,路由算法等等.因为用户能向网络节点中插入程序,因此主动网络能快速创建,部署和管理新的网络服务,以满足用户需求,提供服务质量的保证.该领域的研究成果将对网络用户,服务提供商,以及设备生产商产生广泛而深远的影响.

目前,对主动网络的研究主要还集中在网络体系结构,网络服务创建方法和工具等方面的研究.基于这些研究成果,主动网络研究小组(Active Network Working Group)发布了主动网络的参考体系结构[5],以及主动节点操作系统规范[6].该体系结构把网络节点分为四个部分,分别为底层硬件,节点操作系统(Node operating system, nodeOS),执行环境(Executions environment, EE)和主动应用程序(Active application, AA).节点操作系统统一管理和分配节点的系统资源,并为 EE 提供接口.EE 是主动应用程序的虚拟执行环境.主动应用程序,通过调用 EE 提供的接口,与用户程序协同工作,实现"端到端"的服务.当主动包到达主动节点时,将导致相应的主动应用程序的执行并对主动包进行评估,以决定下一步的动作.因此,主动程序的核心功能就是如何标识,识别,传输,加载和执行主动应用程序对主动包进行处理,以完成用户所定制的计算和网络

收稿日期:2002-06-06;修回日期:2002-09-20
基金项目:国家 863 高科技发展计划(No.863-306-ZD05-01);国家 973 计划(No.G1998030409)

服务.

因此,主动代码分发协议是主动网络研究中一个很值得研究的问题,但是目前对主动代码分发协议的研究还十分有限[7].本文针对这一问题,在主动网络参考体系结构和主动网络节点操作系统规范的基础上,提出了一种基于 XML 的主动应用程序描述符 AADX(Active Application Descriptor based on XML).与其它方法相比,该描述符为主动应用程序的识别提供了一种有效、灵活的方法,并为传输和加载提供了基础.同时,我们用 XML 来描述与应用程序相关的组件信息,从而可以为应用程序的分发和部署提供一种组合机制.

2 AADX 的描述和应用

AADX 是一种描述主动应用程序的描述符,它由位于主动包中的 AADX 标识符和位于远程或本地代码服务器上的 AADX XML 文档描述两部分组成.AADX 标识符用来在主动包中标识用来处理该主动包的应用程序并指明在本地缓存中没有该程序的时候到什么地方去下载该主动应用程序.AADX XML 文档描述则详细描写了与主动应用程序相关的所有信息,比如生产商信息,版本信息,安全信息,以及代码的分发和更新机制.

2.1 AADX 标识符

AADX 标识符是主动应用程序的唯一标识.它用来在主动包中标识对该主动包进行处理的程序,也可用在其它场合作为主动应用程序的标识,如在主动程序的传输和存储中.为了满足在全球范围内对主动应用程序进行永久的独立于位置的唯一的资源标识,我们采用统一资源名称(URN[8])来命名主动程序.把 AADX 标识符作为统一资源名称的一个名字空间.这就解决了主动程序的命名问题.

当第一个主动包经过主动节点时,主动节点要对 AADX 标识符进行解析,以在本地缓存中没有该主动程序时能将该标识符映射为其所对应的程序资源(在本文中指 AADX XML 文档).有两种机制可将 AADX 标识符映射为所对应的程序资源,一种是利用目录服务,如 DNS[9]系统,另一种是在 AADX 标识符中显式指明 AADX XML 文档描述的位置.

综合上面两点,AADX 标识符可用 ABNF[10]描述语言定义如下:

AADXIDNT = "urn:"NID":"NSS["|"URL]
NID = "anaa"
NSS = ProviderId":"NameId":"VersionId
ProviderId = stringname/domainname
stringname = string
domainname = *(domainlabel".")toplabel
domainlabel = alphanum/alphanum * (alphanum/"-") alphanum
toplabel = alpha/alpha *(alphanum/"-")alphanum
NameId = string
VersionId = "v"Version
Version = *(1 * digit".")1 * digit
string = 1 *(alphanum/other)
alphanum = alpha/digit
alpha = %x41-5A/%x61-7A ;A-Z/a-z
digit = %x30-39 ;0-9
other = "+"/"/"/"."/"-"/"@"/"_"/";"
URL = httpurl/ftpurl/fileurl/otherurl

对各部分的说明如下:"anaa"是指 AADX 标识符对应的 URN 名字空间 ID.NSS 是名字空间的字符串.ProviderID 是指软件生产商的名称.该名称可以是在全球范围内的唯一名称(需要注册以保证其唯一性),也可以是其域名.NameId 是应用程序的名称,软件生产商可以自己定义软件命名的格式和规范.VersionId 表示软件的版本.在本文中建议用"."号隔离的数字来表示.URL 是统一资源定位符(Universal Resource Locator).为节约篇幅起见,其中 httpurl,ftpurl 和 fileurl 等的定义请参见文献[11].

2.2 AADX 对 ANEP 的扩展

AADX 的主要目的就是对主动包要求的主动程序进行标识.因此在主动包中,特别是初始主动包,必须携带 AADX 标识符信息,这样主动节点的执行环境才知道加载什么程序以及到什么地方去索取程序以对主动包的内容进行处理.ANEP 协议[12]是由主动网络工作组提出来的主动网络封装协议.但是它只指明了主动包该由什么类型的 EE 来进行处理,而没有说明主动包达到 EE 后该用什么主动程序来处理,而这在初始主动包达到的时候是很重要的.指定对主动包进行处理的程序的方式有两种,一种就是事先在节点中定义,但是这种方式缺乏灵活性.另一种就是在主动程序包中显式(指在主动包中携带有进行处理的主动程序的标识信息,如 AADX 标识符)或隐式(同一连接上的后继主动包可以不携带主动程序标识信息,因为前面的包已经表明了该主动应用程序)进行指定.

ANEP 的格式定义如图 1 所示.其中括号中的数字表示字节数.具体含义请参见文献[12].

Version(1)	Flags(1)	Type ID(1)
ANEP Header Length(2)		ANEP Packet Length(2)
Options(variable)		
Payload		

图 1 ANEP 主动包封装格式

显然,要在主动包中携带主动应用程序的标识信息,可以有两种方法.一种是定义新的选项,即主动应用程序选项,用来表明对主动包进行处理的主动应用程序.另一种方法就是对 ANEP 进行扩展,也就是进一步扩展对负载的定义.扩展后的 ANEP 格式如图 2 所示.

Version(1)	Flags(1)	Type ID(2)
ANEP Header Length(2)		ANEP Packet Length(2)
Options(variable)		
FLG	AADX Length	AADX Payload(variable)
Payload(variable)		

图 2 AADX 对 ANEP 进行扩展后的包封装格式

其中 FLG(2 bits)表示该负载(Payload)是否带有 AADX 标识符,如果没有,则该域和 AADX Length(14 bits)域均为 0.如果该

主动包携带有 AADX 标识符,则 AADX Length 则表示标识符所占用的字节数.

2.3 AADX XML 文档描述

AADX XML 文档详细描述了主动应用程序的生产商信息,版本信息以及代码的分发信息等等. XML 模式(Schema)[13]描述了 XML 文档的生成规则. AADX XML 文档的模式定义如下:

```
<? xml version = "1.0" encoding = "UTF - 8"? >
<! - - edited with XML Spy v4.3 U (http://www.xmlspy.com) by zyz (tsinghua unv.) - - >
< xs: schema xmlns: xs = "http://www. w3. org/2001/XMLSchema" elementFormDefault = "qualified" >
  < xs:element name = "aadx" >
    < xs:complexType >
      < xs:sequence >
        <! - 主动程序的信息部分,说明主动程序的标识符,生产商信息,和描述信息 - - >
        < xs:element name = "information" type = "informationType"/ >
        <! - 主动程序的资源部分,说明主动程序的编程语言,以及打包方法 - - >
        < xs:element name = "resources" type = "resourcesType"/ >
        <! - 主动程序的构造部分,可包含一个或多个组件及其分布信息 - - >
        < xs: element name = "constructs" type = "constructsType"/ >
        <! - 主动程序的调用方法部分,说明主要组件及入口 - - >
        < xs:element name = "invoke" type = "invokeType"/ >
        <! - 主动程序的安全部分,说明安全信息 - - >
        < xs: element name = "security" type = "securityType" minOccurs = "0"/ >
      </xs:sequence >
      < xs:attribute name = "spec" type = "xs: string" use = "required"/ >
    </xs:complexType >
  </xs:element >
  < xs:element name = "component" type = "xs:string"/ >
  < xs:element name = "compress" type = "xs:string"/ >
  < xs:complexType name = "constructsType" >
    < xs:sequence >
      < xs:element ref = "component" maxOccurs = "unbounded"/ >
    </xs:sequence >
    </xs:complexType >
    < xs:element name = "description" type = "xs:string"/ >
    < xs:element name = "identifier" type = "xs:string"/ >
    < xs:complexType name = "informationType" >
      < xs:sequence >
        < xs:element ref = "identifier"/ >
        < xs:element ref = "vendor"/ >
        < xs:element ref = "description" minOccurs = "0" maxOccurs = "1"/ >
      </xs:sequence >
    </xs:complexType >
  < xs:complexType name = "invokeType" >
    < xs:sequence >
      < xs:element ref = "component"/ >
      < xs:element ref = "starpoint"/ >
    </xs:sequence >
  </xs:complexType >
  < xs:complexType name = "languageType" >
    < xs:simpleContent >
      < xs:extension base = "xs:string" >
        < xs:attribute name = "version" type = "xs:string" use = "required"/ >
      </xs:extension >
    </xs:simpleContent >
  </xs:complexType >
  < xs:complexType name = "resourcesType" >
    < xs:sequence >
      < xs:element name = "language" type = "languageType"/ >
      < xs:element ref = "compress"/ >
    </xs:sequence >
  </xs:complexType >
  < xs:element name = "securerole" type = "xs:string"/ >
  < xs:complexType name = "securityType" >
    < xs:sequence >
      < xs:element ref = "securerole" maxOccurs = "unbounded"/ >
    </xs:sequence >
  </xs:complexType >
  < xs:element name = "starpoint" type = "xs:string"/ >
  < xs:element name = "vendor" type = "xs:string"/ >
</xs:schema >
```

2.3 AADX 主动包的处理过程

AADX 主动包的逻辑处理过程如图 3 所示:

当 AADX 主动包经过 NodeOS 被传送到 EE 时,如果 EE 发现在本地缓存中没有该 AADX 的主动应用程序,则调度器将请求 AADX 处理器去检索该程序. AADX 处理器首先根据 AADX 标识符的位置信息获取 AADX XML 文档描述信息,然后再利用 XML 解

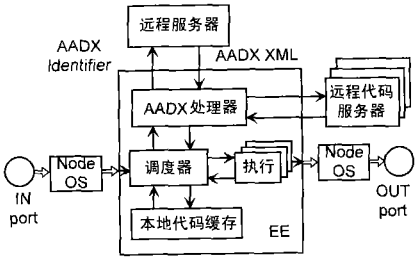

图 3　AADX 主动包的处理过程

析器对 XML 文档进行解析并根据解析结果进行处理，即利用某种传输协议获取主动应用程序的所有组件，并进行相应的安全处理及组合处理，然后把信息传递给调度器．调度器再根据相应的信息产生该主动应用程序的线程实例．

3 实现及性能评价

与普通的主动程序描述和分发机制不同，AADX 引入了 XML．但是 XML 是比较成熟的技术，因此在实现的时候可利用已有的技术．原型系统的硬件和软件环境可描述如下：CPU PIII 866，RAM 128M，OS Linux 2.4，JDK1.2.2．在 XML 解析器的实现上，我们采用了 IBM 的 XML Parser for Java Version 4.0.1[14]．该解析器能支持 XML 模式 1.0，XML DOM（Document Object Model）级别 2[15]，因此利用该解析器就可以对 AADX XML 文档进行解析并利用 DOM API 接口遍历节点并进行 XML 指令的处理．

图 4 是我们在上述实验环境中对 AADX 处理的性能测试结果．从图中可以看出，对 AADX 的处理要花费一定的时间，而且解析文档的时间所所占的比重很大．因此，在实际应用中，首先要优化解析器或使用专门的解析器，其次可以使用探路主动包来在沿途各主要节点预先加载主动应用程序．

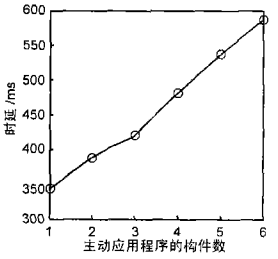

图 4 AADX 处理的性能

4 相关工作

对主动应用程序的描述，ANTS[16]提出用指针来进行描述．ASP[17]提出了一种利用 AAspec 对象来描述主动应用程序的方法．但是这些方法都只能对特定的主动应用程序进行标识，而且它们没有提供一种通用的命名机制，也不能对由多个协议构件组成的主动应用程序进行描述，因此都不具有扩展性．AADX 利用较短的适合于放在主动包中的 AADX 标识符，以及较长的灵活的可扩展的 XML 文档来描述主动应用程序，具有良好的可扩展性．而且，AADX 可以对主动应用程序的组成构件进行描述，从而为服务组合提供了一条可行的途径，同时避免了相同协议构件的重复下载和缓存，也为主动应用程序的分发提供了一种分布式的分发机制．

5 结束语

本文针对主动应用程序描述的问题，提出了一种新的基于 XML 的描述方法．该方法具有良好的可扩展性，从而为主动应用程序的标识和分发提供了一种很好的机制．但是 AADX 的解析开销比较大，我们下一步将对 AADX 的解析进行优化，以达到更好的性能．

参考文献：

[1] RFC 1633, Integrated Services in the Internet Architecture: an Overview [S].

[2] RFC 2475, An Architecture for Differentiated Services [S].

[3] Zhang L, Deering S, Estrin D, etc. RSVP: A new resource reservation protocol [J]. IEEE Network, 1993, 7(5): 8 - 18.

[4] D Tennenhouse, J Smith, W Sincoskie, et al. A survey of active network research [J]. IEEE Communications Magazine, 1997, 35(1): 80 - 86.

[5] K L Calvert, ed. Architectural Framework for Active Network, Version 1.0 [DB/OL]. Active Network Working Group, July 27, 1999.

[6] AN Node OS Working Group. NodeOS interface specification [DB/OL]. http://www.cs.princeton.edu/usg/papers/nodeos.ps, January 2001.

[7] W Lau, S Jha, M Hassan. Current directions in active programmable network [A]. Bob Werner. Proc. of IEEE International Conference on Networks (ICON) [C]. Bangkok, Thailand: IEEE Computer Society, 2001: 240 - 245.

[8] RFC 2141, URN Syntax [S].

[9] RFC 1464, Using the Domain Name System to Store Arbitrary String Attributes [S].

[10] RFC 2234, Augmented BNF for Syntax Specifications: ABNF [S].

[11] RFC 1738, Uniform Resource Locators (URL) [S].

[12] D Alexander, et al. Active Network Encapsulation Protocol [DB/OL]. http://www.cis.upenn.edu/~switchware/ANEP/, July 1997

[13] David C Fallside, et al. XML Schema Part 0: Primer. W3C Recommendation 2 [DB/OL]. http://www.w3.org/TR/xmlschema-0/, May 2001.

[14] IBM Alphaworks. XML Parser for Java Vwesion 4.0.1 [DB/OL]. http://www.alphaworks.ibm.com/tech/xml4j.

[15] Arnaud Le Hors, et al. Document Object Model (DOM) Level 2 Core Specification v1.0. W3C Recommendation [DB/OL]. available online at http://www.w3.org/TR/2000/REC-DOM-Level-2-Core-20001113, W3C, November 2000.

[16] D Wetherall, et al. ANTS: A toolkit for building and dynamically deploying network Protocols [A]. Aurel A. Lazar. Proc. of IEEE OPENARCH'98 [C]. San Francisco, CA, USA: IEEE, Inc, 1998. 117 - 129.

[17] B Braden, B Lindell, et al. The ASP EE: An active network execution environment [A]. Bob Werner. Proceedings of DARPA Active Networks Conference and Exposition (DANCE'02) [C]. San Francisco, CA, USA: IEEE, Inc, 2002. 238 - 254.

作者简介：

周悦芝 男，1975 年出生于湖南衡阳，1998 年于北方工业大学获学士学位，同年进入清华大学计算机系攻读硕士学位，2000 年提前攻读博士学位，主要研究领域为主动网络、家庭网络等．E-mail: zyz00@mails.tsinghua.edu.cn

张尧学 男，1956 年出生于湖南常德市，工学博士，清华大学计算机系教授，博士生导师，主要研究领域为计算机网络路由器、网络协议工程、主动网络和程序挖掘等．