

基于 Java 的主动网络接口编程

张 鹏

(东华大学 信息学院, 上海 200051)

摘 要:在主动网络研究过程中,各个研究机构提出各种不同的执行环境(Execution Environment, EE)。不同的 EE 有各自的优点和缺点,而它们之间的互操作性问题越来越成为制约主动网络发展的问题之一。针对这个问题,文中提出了主动网络接口编程(ANSP)模型,它是一系列接口的集合,简便、透明,位于现有的执行环境之上,基本解决了上述互操作问题。

关键词:主动网络接口编程;执行环境;主动应用

中图分类号: TP311.1

文献标识码: A

文章编号: 1005-3751(2006)03-0026-03

Active Network Socket Programming Based on Java

ZHANG Peng

(Information College, Donghua University, Shanghai 200051, China)

Abstract: During the research of active network, some graduate schools introduce different execution environment (EE). Each EE has its own advantage and disadvantage, but the interoperability among them mostly prevent development of active network. So found an active network socket programming (ANSP) model, which is muster of a series of socket. It is simple and transparent. It is upon existing EE, and settle the interoperability problem above basically.

Key words: active network socket programming; execution environment; active application

0 引 言

主动网络的本质是使网络可编程,即网络中的主动节点可以处理由用户编写的主动代码,执行各种用户期望的应用。节点系统结构可分为3个层次,分别是节点操作系统(Node OS, Node Operation System);执行环境(EE, Execution Environment);主动应用(AA, Active Application),它们分别为前者运行的环境^[1]。

在主动网络的研究过程中,各种研究机构纷纷提出了各种执行环境,比如南加州大学 ISI 提出的 ASP 项目^[2]和麻省理工的 ANTS 项目等等。这些执行环境基于节点操作系统,为主动应用的运行提供了条件。但在一个大型的主动网络实验环境中,所有的节点不会安装同样的执行环境 EE,而在某一 EE 下编写的主动应用 AA 则只能在本 EE 上运行,在装有其它 EE 的主动节点上则无法执行,这就出现了操作性问题。

主动网络执行环境 EE 的互操作性问题在一定程度上制约了主动网络的发展和普及^[3]。为此文中提出了基于 JAVA^[4]的主动网络接口编程(ANSP)模型,以整合所有节点上的 EE,解决上述互操作性问题。ANSP 被设计成在所有 EE 之上的一层,不用对现有 EE 做任何改动,简

便、透明,并且可根据需要装载。

1 设计 ANSP 的几个问题

ANSP 的设计类似于提供翻译机制的中间件编程,但统一的接口只是 ANSP 的一部分,还有其它需要考虑的问题:

1) 多数的 EE 是基于主动包编程模型的,每种模型都有一整套已定义好的构件,每个构件都有自己的程序接口。尽管 ANTS 和 ASP 等都是基于这种模型,但它们各自的构件及接口却不同。因此,在一个 EE 上写的主动应用程序无法在其它 EE 上运行。

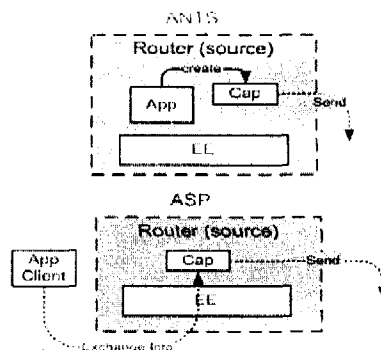


图1 ANTS和ASP编程模型

如图1所示,模型 ANTS 包括3个截然不同的构件: Application, Capsule 和 EE。用户编写的主动应用 App 生成一个 Cap,然后发送出去,它们和 EE 之间都有接口。而

收稿日期:2005-06-04

基金项目:国家自然科学基金资助项目(70271001)

作者简介:张 鹏(1981-),男,山东人,硕士研究生,研究方向为主动网络与移动计算。

ASP 则包括 App Client, EE 和 AAContext。App Client 可以运行在主动或非主动节点上,作用是由用户引发一个主动应用;AAContext 的功能类似于 ANTS 中 App 在源\目的节点上和 Cap 在路径节点上综合起来的作用。

为了解决上述互操作性的问题,ANSP 模型需要新的一整套统一的接口并将其配置到所有 EE 上去。

2)为了保证每一个 EE 运行的独立性,控制每一个主动应用的接入,ANTS 为每一个 AA 生成一个虚拟机,占用了太多资源。而 ASP 只建立一个虚拟机,但对它的接入设限。因为各类型 EE 的线程机制的不同,ANSP 须建立一个新的线程库来统一接入。

3)多个封包可以通过软件库在网络中交换信息,但当一网络服务可以在同一个路由器上的不同 EE 上执行时,问题出现了,尤其是一个服务请求在同一路由器上的不同时间、不同 EE 上装载,然后找回自己的数据时,由于不同 EE 不可以交换信息,于是上述服务便无法找回自己的数据。因此 ANSP 必须要建立统一的软件库机制以保证服务随时随地的接入。

4)当原有的小且分割的网络并入 ANSP 模型时,清晰的拓扑结构就显得异常重要,然而不同的 EE 有不同的地址表和私有路由拓扑。因此,ANSP 应建立统一的地址表,无缝地融入各种 EE。这样拓扑信息就可以在不同 EE 间传递,每个 EE 也就有了全面的网络拓扑视角。

5)基于 JAVA 语言的动态装载能力,大部分 EE 模型是基于 JAVA 的,如 ANTS, ASP 等。但也有例外,如 PLAN 是由 OCaml 编写的。它们各自编写的主动应用现在还不能互用。因此语言无关性也是 ANSP 设计的一个主要目标。

6)各种 EE 下定义的主动封包头结构不同,它们携带相关重要信息,每种 EE 都无法读取其它 EE 下封包头的信息,以致封包失效。故 ANSP 须建立合适的编译机制以使目标 EE 可以解读各种封包头信息。

2 ANSP 模型

本模型提供了一整套程序接口,可使 AA 运行在各种 EE 上。接口层由独立的两层构成,上层为 AA 提供统一的编程模型,下层提供合适的翻译机制,两层之间是 EE。ANSP 还提供了一套服务请求程序来代表可用的服务和资源。通过一系列一致且透明的接口,ANSP 应用可以访问不同 EE 下的同一程序资源,于是表面上看,ANSP 是直接面向应用的,而实质上 ANSP 只是在 AA 和各种 EE 之间提供了通用的接口层。

ANSP 模型包括 4 个部分,主动代码封包(简称封包)、主动服务库、应用客户端和应用缓存。如图 2 所示。

1)封包(ANSPCapsule)。封包携带用户代码(包括前向程序),在从源节点到目的节点上执行前向程序,最终在目标节点上运行(见图 2)。新的封包对象通过扩展核心类 ANSPCapsule 得到,包含的主要方法如下:

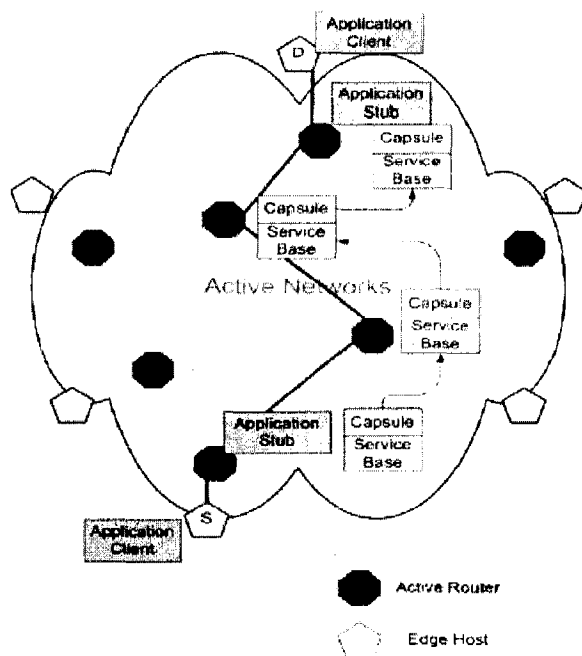


图 2 ANSP 模型

ANSPXdr decode()

ANSPXdr encode()

Int length()

Boolean execute()

主动路由器调用 decode()方法将接受到的二进制字节流重组成封包;调用 execute()方法访问封包的前向程序;当封包完成任务并转向下一跳时,调用 send()方法,同时 encode()将封包转成二进制流;而 length()用在 encode()方法里来决定产生字节流的长度。

2)主动服务库(ANSPBase)。在主动节点中,主动服务库为可用的 EE 资源和功能提供了统一的输出接口。这些服务包括线程管理、节点查询和软件库操作等,方法定义如下: Boolean send(Capsule, Address), AN - SPAddress getLocalHost(), Boolean isLocal(ANSPAddress), createThread(), forName(PathName)等。

3)应用客户端(ANSPClient)。

boolean start(args[])

boolean start(args[], runningEEs)

boolean start(args[], startClient)

boolean start(args[], startClient, runningEEs)

应用客户端是用户和最近的主动节点之间的接口,除了提供人机界面外,还负责在某些 EE 中找出应用代码的名称和位置,并将应用初始化(如在可用的 EEs 中选取合适的 EE)。每个主动应用可以通过扩展核心类 ANSPClient 创建一个应用客户端,继承了方法 start(),包含参数 args[]和可选的参数 runningEEs 等(见图 3)。

4)应用存储池(ANSPApplication)。应用存储池位于源和目的节点上,负责应用客户端之后的 ANSP 应用的初始化。它按照应用客户端的要求接受并处理网络中的封包。它来自核心类 ANSPApplication 的扩展,包括处理程

序 receive(ANSPCapsule)的定义。

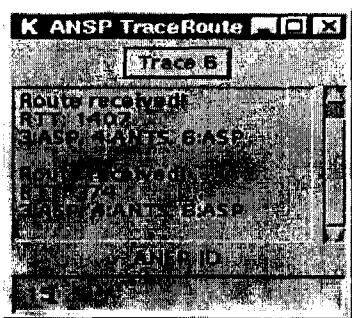


图 3 应用客户端上某路径追踪程序 GUI

3 一个 ANSP 例子:路径追踪

为验证 ANSP 可以整合不同的 EE,做了如下实验,其中的节点 EE 设置上包括 3 种类型:ASP,ANTS 和两者兼有。试验原型机是由 JAVA 编写的,包括一个路径追踪试验程序,这个程序记录了所有它所访问过的中间节点的 EE 类型以及它们之间的 RTT。图 3 展示了应用客户端上的 GUI(图形用户界面),可知它找到了 3 个 EE:ASP,ANTS 和 ASP。实验方案及程序的执行过程如图 4 所示。

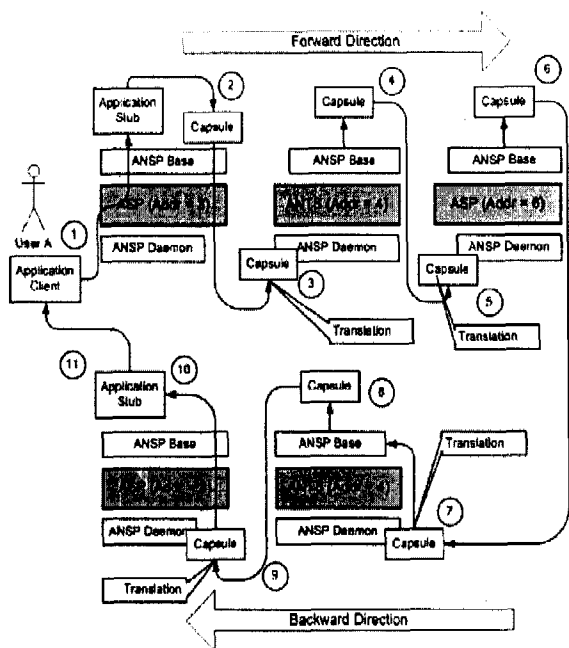


图 4 路径追踪封包的流程

路径追踪封包程序代码通过扩展核心类 ANSPCapsule 得到,当调用 execute()方法时它检查布尔变量 returning 来判断本程序是否需要到达目的节点时返回。在从源节点到目的节点的途中,封包程序通过调用 addHop()方法记录下其访问路由器的地址和 EE 类型。当它到达目的节点时,将 returning 设置成 false,并调用 send()返回源节点。当回到源节点时,它调用 deliverToApp()进入应用存储池。封包通过调用 encode()和 decode()将网络中所得数据用 XDR 表示方法进行封装,并携带回源节点,ANSP 的 XDR 的句法遵循了 ANTS 的 XDR 的句法库。Length()方法返回数据的长度,另外利用 XDR 中的初始

类也可以计算数据长度。

4 实验

为了验证 ANSP 对主动网络延迟率和带宽的影响,利用 NS-2 仿真以及 JAVA 语言^[5]模拟了一主动网络,共 16 个主动节点,测得节点 EE 分别为 TCP(无主动网络),ANTS,ASP 和 ANSP 时的延迟率和带宽,如图 5 所示。

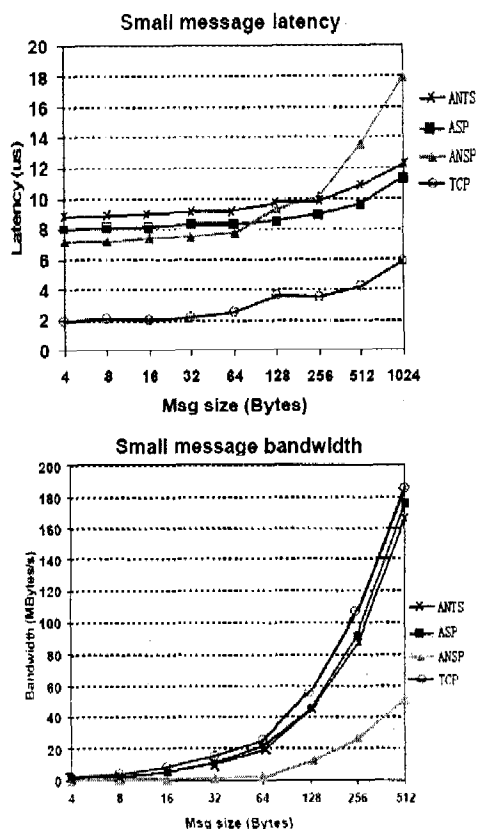


图 5 延迟率和带宽测量结果

由上图可知,ANSP 虽然将各个 EE 融合在一起,提高了互操作性,但也增加了主动节点的复杂程度,增加了网络延迟率,降低了带宽,但这种网络性能上的损失一般是可以接受的。

5 结论

文中提出了一个新的分层的主动网络结构——主动网络接口编程(ANSP)模型。它使主动应用只被编写一次就可以在多 EE 的主动网络环境中运行。实验表明,在不对原有结构做任何改动的情况下,ANSP 可以成功地与 ANTS,ASP 正常协作,它的设计被证明是成功的,尽管对网络性能有一定影响。同时,ANSP 中的统一编程接口层简洁且可根据需要动态配置。

参考文献:

- [1] Tennenhouse D L, Wetherall D J. Towards an active network architecture [A]. Proceeding of Multimedia Computing and Networking [C]. San Jose, California, USA: [s. n.], 1996. 80-83. (下转第 148 页)

数据的类型字段实现对数据流的解析。通过类型字段来判断当前流中的对象类型,实现数据的读取过程。

3.3 地下管网的显示

通过从数据库中获得的管线起始点坐标(X_0, Y_0),计算构成管线其他点的坐标(X_i, Y_i), X 坐标由西向东, Y 坐标由北向南。结合高程数据形成(X_i, Y_i, Z_i)的数组。根据已知的管线类型、形状,可利用三角形作为基本面素来逼近管线表面。用三角形来表示管线表面的好处是操作和变换都可以针对三角形的顶点来进行。

管网三维图形的绘制就是一个将几何模型从数字表达形式绘制到硬件设备上并输出显示的过程。三维显示实际上是三维模型在二维平面上的投影^[7,8]。初始化的管线表面要经过一系列的变换才能在屏幕上显示三维效果。这些变换主要包括两种形式:

一种是几何变换,由变换矩阵来完成。变换矩阵轮流对三角形每一个顶点的坐标进行变换,形成新的三维效果。例如通过平移、缩放、旋转等变换操作,来改变三维图形在屏幕中的相对位置,将目标放入观察范围内。OpenGL中有一个坐标变换矩阵栈(ModelView),栈顶就是当前坐标变换矩阵,进入OpenGL管道的每个坐标(齐次坐标)都会先乘以这个矩阵,结果才是对应点在场景中的世界坐标。OpenGL中的坐标变换都是通过矩阵运算完成的。

另一种变换就是投影变换。投影变换又包括正射投影(即没有“近大远小”)和透射投影(“近大远小”)两种^[9]。正射投影使用 glOrtho 创建一个平行视景体的矩阵,即投影射线是平行线。把第一个矩形视景投影到第二个矩形视景上,并用这个矩阵乘以当前矩阵,以完成变换。透射投影使用 glFrustum 创建一个形如棱台的视景体,其近截取面由 left right bottom top znear 确定;远截取面由从视点投影近截取面到 Z 轴 $zfar$ 位置决定。要把三维管网图最终显示在窗口上,还需要进行视口变换。视口变换使用 glViewport 函数定义一个视口,通过给出视口在屏幕窗口坐标系中左上坐标及宽和高等参数来实现。注意使用中视口长宽比例的调整会导致图像变形。因此 reshape() 中要检测窗口尺寸,修正视口大小,保证图像不变形^[7,8]。

3.4 事件处理

在 OpenGL 中,一个事件意味着信息从管网图中的一个节点传递到另一个节点。在管网图之外的一个输入行为将引起管网图中节点及管线的运动,交互和动画能够使

管网图或管网图中的形体对象对外界介入动作产生反应。这种介入可能是键盘上的按键操作、鼠标的移动、对象的碰撞、时间的改变或者其它一些事件。变化的产生包括管网图中对象数量的增加、减少,属性的改变,对象的重新排列以及这几种情况的组合等。交互和动画的区别在于交互是对用户输入行为的反应,而动画则是对象随时间变化而产生的改变。OpenGL 中使用双缓冲技术来实现动画^[9]。每一帧都在画面外的缓冲区绘制,完成之后再交换到屏幕上。在 C 语言编程机制下,主要使用 GLUT 库函数来实现动画。基于以上原理,对三维图形进行缩放、漫游是一种交互行为,是对用户介入行为的反应,表现为图形在屏幕中的变换;而沿固定路线飞行则是一种动画的效果。

4 结束语

文中提出的基于 OpenGL 的可视化算法,结合 Oracle 数据库及地理信息系统开发工具,能够有效地解决城市地下管网地理信息系统的可视化问题。加之 OpenGL 是目前应用最为广泛的开放式图形编程标准,使它在许多方面优于其它的三维图形绘制库,有着广阔的应用前景。

参考文献:

- [1] 龚建华. 地学可视化——理论、技术及其应用[R]. 北京: 中国科学院国家计划委员会地理研究所, 1997.
- [2] 李翔, 李成名, 王继周. 基于 Java3D 的地形 3 维可视化技术[J]. 测绘通报, 2003(10): 19-21.
- [3] 张杰. JAVA 3D 交互式三维图形编程[M]. 北京: 人民邮电出版社, 1999. 26-50.
- [4] 乔林, 费广正, 林杜, 等. OPENGL 程序设计[M]. 北京: 清华大学出版社, 2000. 85-96.
- [5] 和平鸽工作室. OpenGL 高级编程与可视化系统开发[M]. 北京: 中国水利水电出版社, 2003. 90-103.
- [6] 吴波. 城市地下管网信息系统的设计与实现[D]. 西安: 西北大学, 2002.
- [7] 周雪梅, 杜世培. GIS 中三维可视化的模型构造及算法设计研究[J]. 贵州工业大学学报, 2003, 32(4): 55-57.
- [8] 吕志慧. 地理信息三维可视化系统应用研究[D]. 郑州: 郑州大学, 2002.
- [9] Hearn D. Computer Graphics with OPENGL(3e)[M]. 蔡士杰译. 北京: 电子工业出版社, 2005.

(上接第 28 页)

- [2] Braden B, Cerpa A, Faber T, et al. Introduction to the ASP Execution Environment[EB/OL]. www.isi.edu/activesignal/ARP/index.html, 2001.
- [3] Descasper D, Parulkar G, Plattner B. A scalable high performance active network node, IEEE Network (New York USA) [J/OL]. http://www.arl.wustl.edu/Publications/1995-99/ieeener99dd.pdf, 1999.
- [4] Gosling J, McGilton H. The java language: a white paper, Technical Report, Sun Microsystems[EB/OL]. http://java.sun.com/docs/white/langenv, 2001.
- [5] 彭晨阳. Java 实用系统开发指南[M]. 北京: 机械工业出版社, 2004.