

WAP 协议栈在 Elastos 平台上的设计与实现研究

张捷^{1,2}, 陈榕²

(1. 安徽师范大学, 安徽 芜湖 241000; 2. 同济大学基础软件工程中心, 上海 200092)

摘要: Elastos 是基于构件的嵌入式操作系统, 应用于手机等数字移动设备中。WAP 协议代表了主流的基于移动通信的互联网服务无线访问技术的发展方向。为了满足数字设备的无线应用需求, 分析了现有的 WAP 协议规范及其应用模型, 讨论了 WAP 中 WSP 和 WTP 两个部分的结构与技术细节, 以 Elastos 系统平台为基础设计了构件化的 WAP 协议栈。并结合具体的 CAR 构件编程模型, 依托 Elastos 的构件技术优势实现了具备基本功能的 WAP 协议栈。目前, WAP 构件模块已稳定运行在实际手机应用项目中。

关键词: “和欣”操作系统; CAR 构件技术; WAP 协议栈

中图分类号: TP311.5

文献标识码: A

文章编号: 1673-629X(2011)02-0094-04

Research of Design and Implementation of WAP Protocol Module on Elastos Platform

ZHANG Jie^{1,2}, CHEN Rong²

(1. Anhui Normal University, Wuhu 241000, China;

2. System Software Center, Tongji University, Shanghai 200092, China)

Abstract: Elastos is a component-based embedded os. It applies in telephone and another mobile equipments. WAP protocol is developing direction of internet wireless access technique based on mobile communication. To meet requirement of wireless application in digital devices, analyse standard and application model of WAP protocol, discuss the structure of WSP & WTP module in WAP and details of them, and design the WAP protocol component in Elastos platform. Then combining with the CAR programming model, implement the WAP module based on technical superiority of component in Elastos. Now this module can work stably in actual mobile projects.

Key words: Elastos; CAR component technique; WAP protocol

0 引言

“和欣”^[1] (Elastos) 操作系统是由科泰世纪公司开发的, 具有中国自主知识产权的面向服务的新型嵌入式网络操作系统, 是国家“863”计划的重要成果。Elastos 是完全面向构件的操作系统, 操作系统提供的功能模块全部基于构件技术, 都是可拆卸的构件, 并以此来扩充操作系统和应用程序的功能。

构件技术因其本身的灵活性、实时性以及安全性等特点, 在嵌入式操作系统开发中受到越来越多的重视。Elastos 平台上采用了称为 CAR^[2] 的构件技术。CAR 是一套完整的构件支撑平台, 其上涵盖了面向构

件的编程技术以及构件间相互调用的方法与标准。采用 CAR 标准的二进制构件能够在运行时动态链接。

WAP(无线通讯协议)是在互联网与数字移动电话、个人数字助理乃至未来的信息家电之间进行通讯的全球性开放标准。WAP 协议可以将因特网服务或其它数据业务以智能信息传送的方式引入各种数字终端平台, 具备兼容性、可靠性和互操作性的要求, 是目前使用广泛的一种用于跨平台数据信息传送的无需授权协议^[3]。包括微软、Yahoo、AOL 等在内的各大软件、信息和服务提供商也都相继建立起可通过手机或其它移动数字终端访问的网站, 为用户提供无线访问内容服务。事实上, 移动访问服务已成为当前发展最为迅速的一类信息增值服务^[4]。

1 WAP 的应用模型

WAP 是针对无线网络带宽窄、无线通讯可靠性低的特点而设计的独立与底层承载的应用协议, 它的应

收稿日期: 2010-06-02; 修回日期: 2010-09-27

基金项目: 国家“863”计划资助项目(2001AA113400)

作者简介: 张捷(1980-), 男, 安徽芜湖人, 硕士, 研究领域为嵌入式操作系统、系统软件支撑技术; 陈榕, 教授, 博导, 研究领域为网络操作系统、嵌入式操作系统、构件技术。

用模型如图 1 所示。

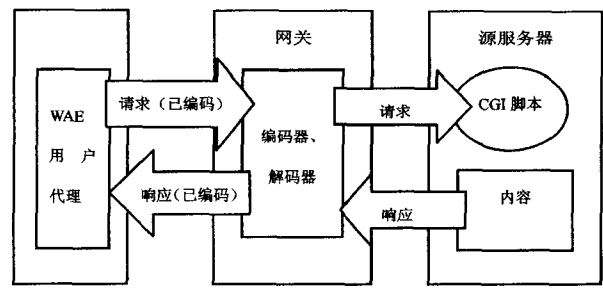


图 1 WAP 应用模型

由图 1 所示,在用户和 WAP 网关之间的是无线网络,在国内一般指蜂窝移动通讯网,在其上运行 WAP 协议。在 WAP 网关和服务器之间的是有线网络,可以利用现有的 Internet 进行通信。这一模型说明了 WAP 是建立在 Internet 基础上,考虑了现有的 Internet 协议,把用户的移动接入性引入其中^[5]。

用户代理包括 WML 用户代理和 WTA 用户代理。其中,WML 用户代理的作用是对 XML 编码以及对 WMLScript 的编译;WTA 用户代理允许用户和移动电话进行访问和交互作用。源服务器响应用户的各种请求,其上存放着用户浏览的信息。应用于 Internet 上的 Web 服务器完成了这部分功能,而目前应用在 Web 服务器上的成熟技术如 CGI、ASP、JSP 等,在此依然可以适用^[6]。

2 WAP 协议栈的设计

2.1 总体结构

这里主要指的是 WSP 和 WTP 在 Elastos 平台上所具备的结构特征,WSP 和 WTP 是整个 WAP 协议簇中最为重要的两个模块。依据其功能划分,它们的结构如图 2 所示。

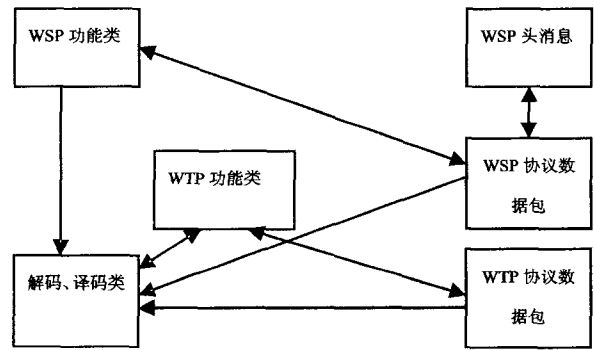


图 2 WAP 协议栈模块结构

具体的协议类定义以及相关成员函数实现这里不再重述。

针对 Elastos 平台的特点,将各模块设计成具有相同特征行为的一组构件,每个构件需满足一个或多个功能的特定服务需求。同时构件本身应隐藏自身具体

实现,而通过接口的方式为用户提供服务^[7]。考虑到现行的终端智能设备对 WAP 协议栈支持有一定局限性,所以采用最小元素集合的设计方法,同时也兼顾了可扩展性的要求。在 WAP 协议的下层支持方面采用了 UDP 技术,而 UDP 协议目前是使用广泛而又可靠的。此外针对手机应用的特点,必须要扩充媒体格式以支持彩信等一系列应用,这部分工作将在同属于 WAP 协议簇的 WAE 模块上实现。

2.2 WSP 协议层与 WTP 协议层

WSP 是会话层(Session Layer)的通讯规定,提供两种服务,包括架构在 WTP 之上的面向连接服务和架构在 WDP 之上的无连接服务。提供以下功能:在压缩的超空间编码中的 HTTP/1. 功能和语义;长久对话状态;通过对话移动暂停和恢复;可靠和不可靠数据的普通设备;协议特性流通^[8]。

以和 WAP 网关建立会话连接为例。在 WAP 协议栈中,建立会话需通过标准原语 s_connect 向 WTP 层传递相关参数,由 WTP 向 WAP 网关建立起 socket 连接;一旦 socket 连接建立成功,网关通知 WSP 层准备建立会话,继而再发起一个事务。因需不断监测数据包的发送,网关服务器上 CWTPSocket 模块将单独启动一个线程来监听数据包,如果监听到一个数据包的到来,便使用相应的解码函数去对这个数据包进行解码处理,反之则进行编码处理。

再者,当一个会话结束行将断开之时,由 WAP 网关服务器通知用户该会话将不能维持。断开一个对话可以由本地服务使用者,或者对等端的服务使用者或是服务提供者发起,标准原语 s_disconnect 提供这一方法。在会话断开指示之后,与此会话相关的服务原语将不再出现。

WTP 是架设在 WDP 之上的交易层(Transaction Layer)的通讯规定,是为了小型客户端界面(如移动电话)所设计的,类似于 WWW 中的 TCP 协议,在不可靠的数据报服务之上为上层提供可靠的服务。WTP 协议层被规定必须运行在数据报服务之上,所以设计时必须着重注意要提供端口号以为将来的数据报路由到 WTP 层,以及必须向上反馈到达 WTP 层的 SDU 的长度信息等,这些都是为了满足可靠性需求。^[8,9]

此层中一个重要的问题是协议本身的状态控制,保持状态控制的准确性非常重要,否则会影响到整个事务的处理。这里较难处理的是事务的发起者与响应者在某些时刻并不同时存在的问题,比如 WTP 用户是 WSP 客户端协议,则 WTP 提供者可能仅需要支持事务的发起,也就是说可以没有响应者。在功能设计时需注意到一个 WTP 事务的实现并不总是需要发起者和响应者都存在,而这会影响到状态的自然转换,在实

现中要作特别处理。

WTP 层对上层的接口由标准原语来定义,而它对下层因为需提供的典型的数据报服务,故仅需要用到单元数据的指示和请求原语。来自上层的请求和响应与来自下层的指示统称为外部事件。除了外部事件外还存在内部事件,例如计时器定时事件以及内部错事件。事件在被处理之前都是有效的,若不采取特殊的操作,这些内外事件的处理过程就需要根据一张状态表来完成。状态表是 WTP 层核心的处理算法,它的设计需要保证任一事件处理的正确性与及时性需求,具体的状态表较为庞杂,这里不再给出。

2.3 构件化 WAP 协议栈

Elastos 与其它嵌入式操作系统的区别在于它采用完全面向构件的平台架构,能有效地支持 WEB 服务,可以更好地适应网络时代的嵌入式应用需求。Elastos 同时也是基于构件容器的概念管理各功能模块的,不同的构件独立性强,解耦性好。而且不同构件之间的交互性和远程通信问题通过列集/散集机制解决且又不会互相产生干扰,安全性高^[10]。Elastos 的中间件平台功能模块全部基于 CAR 构件技术,支持链接库 DLL 与系统模块的动态加载。它使用构件技术的优势还体现在具备先进 GUI 事件处理机制的图形系统、使用时实时加载的设备驱动系统,以及更安全的集约化文件系统等等之中。^[11]

在设计构件化的 WAP 协议栈时,一个关键点在于如何划分各个子构件的粒度以及如何设计各子构件接口,使得同类型的构件及接口是特征相关的。还有重要的一点是怎样设法提高网络通信效率,使得网络带宽瓶颈不足以对整个应用起决定性影响。考虑以层次为粒度来划分构件,将上述的 WSP 层和 WTP 层分别设计成独立的主要子构件。这样在进行后续开发与维护时,使得相关人员通常只需关心某一层的协议而不需考虑整体的繁杂情况。例如若对会话层协议很熟悉,用户可以利用自身相关知识或者针对一些特定的需求,来对此层协议进行相应的改进或优化。

通过对 WAP 协议栈的结构分析,设计构件接口的首要问题是能实现 WAP 的基本功能以满足用户的一般需要。因目前的应用主要集中在手机端接发彩信上,所以构件接口功能设计体现在下述方面:WAP 协议栈初始化、检测 WAP 连接是否成功、下载 WAP 数据、发送 WAP 数据、退出 WAP 协议栈以及对协议栈进行资源释放等。

在 Elastos 平台下具体处理时,为 WAP 构件实现了三个主要接口:

IWSP 接口:主方法是和 WAP 网关建立连接,该接口提供了连接参数设置、建立会话连接和断开会话连

接的系列操作;

IWTP 接口:提供 Get/Post 等数据事务操作;

IWDP 接口:主方法是在开始建立会话前保证低层网络的连通性,如建立 socket 等。

3 WAP 构件的实现

对构件的定义如下:

```
module www.elastos.com/car/ezWap.dll
{
    interface IWDP {
        SocketInit([ in ] WString wsWapGatewayAddress,
            [ in ] Int32 nGatewayPort);
    }
    interface IWTP {
        BeginToLoad([ in ] WString wsURL,
            [ in ] WString wsHeader);
        ReceiveContent(.....);
        Send(.....);
    }
    interface IWSP {
        Setparam([ in ] WString wsWapGateway,
            [ in ] Int32 nWapPort, [ in ] Int32 nMaxCsdu,
            [ in ] Int32 nMaxSsdu);
        Connected();
        Disconnect();
    }
    class CWapStack {
    public:
        interface IWDP;
        interface IWTP;
        interface IWSP;
    };
}
```

接口(interface)是用来定义一种程序的协定。实现接口的类或者结构要与接口的定义严格一致。有了这个协定,就可以抛开编程语言的限制(理论上)。接口可以从多个基接口继承,而类或结构可以实现多个接口。接口可以包含方法、属性、事件和索引器。接口本身不提供它所定义的成员的实现。接口只指定实现该接口的类或接口必须提供的成员。

以 interface IWDP 为例,它定义了主方法 SocketInit,其功能是在会话建立连接前,保证网络已连接上(建立 socket 端口);之后通过 WSP 层 s_connect 原语向 WTP 层传递相应参数,WTP 层便和 WAP 网关建立 socket 连接;连接建立成功后,方法通知 WSP 层准备建立会话,继而发起一个事务。

此构件定义在 Elastos 平台下被编写为一个称为 CAR 的文件,用相应的工具生成源代码如下:

```
#include "CWapStack.h"
#include "_CWapStack.cpp"
ECode CWapStack::SocketInit(
    [ in ] WString wsWapGatewayAddress,
```

```
[in] Int32 nGatewayPort)
{
    // TODO: Add your code here
    return E_NOT_IMPLEMENTED;
}
.....
```

在生成的文件框架中,方法的返回的 ECode 值为 E_NOT_IMPLEMENTED,表示此方法未实现。可以进一步根据上述的功能定义填写实现各方法,并适当修改返回的 ECode 值。需要说明的是 CAR 构件标准规定,接口方法的返回值类型应全部都是 ECode,其值定义成 4 字节二进制整数,用以跟踪方法或者函数的调用错误。

此外因 interface IWTP 主要提供数据事务操作,需运行在数据报服务之上,故具体实现时需要解决 WTP 协议层的一些具体问题:消息传送、重传直到确认、用户确认、最后确认的信息、事务标识符及定时器的使用等。该接口的 BeginToLoad() 方法是开始接收,参数是输入参数目标地址和内容的头。ReceiveContent() 是接收内容的方法,参数为输入参数事务标识符和输出参数具体内容。Send() 是发送内容的方法,参数为输入参数,分别为地址、主题、头和类型。

而 interface IWSP 内含与 WAP 网关建立连接以及断开的系列操作,它的 Setparam() 方法是在会话连接前首先设置参数,包括网关端口等;Connected() 方法是连接会话;Disconnected() 方法是断开会话。这些方法的具体实现这里不再给出。

此 WAP 二进制构件具体的生成方法需要依托和欣嵌入式交叉开发平台完成,这里不在叙述。如果其它平台构件或者客户的普通应用程序需调用此组构件的功能,它首先需要实例化一个进程内 WAP 构件对象,然后通过对象的接口方法去调用它所提供的服务。注意当所有的服务使用结束后,客户程序不应再保留 WAP 构件对象,而需及时释放掉对象本身以及对象所占有的资源。

目前二进制构件模块 ezWap.dll 已具备完整的功能实现,它的结构功能设计符合 WAP 协议栈的开放标准,模块初步版本也已经过实际项目的检验,在可靠性和资源利用率等方面均表现良好。在实际使用中,需注意 WAP 构件模块和客户程序的协调性问题,主要反映在二者运行在同一进程或不同进程中。运行在同一进程时,因为 ezWap.dll 和客户镜像处于同一地址空间,当客户程序与 WAP 构件程序建立起通信关系后,客户程序得到的实际构件接口(指针)即为构件对象的虚函数表入口。对 DLL 文件的处理为运行时载入内存,故可保证 DLL 本身的相对独立,它不依赖于

客户程序。客户程序在具体使用时可以利用这一特点尽可能最大共享 DLL 模块,以达到节省内存的目的,这符合嵌入式环境下的优化需要。另外在同一进程内对 ezWap.dll 使用采取动态连接的方法可省却查询操作而直接使用接口方法。使用 DLL 文件形式作为构件实现的载体,可以让应用程序跟底层模块相对独立,便于彼此的更新与升级,也使得存储空间的利用效率增加。

而对于跨进程的操作,二进制构件使用信息的获取方法有点特殊。在 Elastos 平台下,每个 CAR 构件在编译生成时都会伴随产生一个全局唯一的元数据文件,其内含有此构件的自描述信息,主要是它对外有多少接口以及各个接口方法的说明等。根据此元数据文件,客户调用者程序会在自己内部先生成一个代理构件 proxy,因代理构件的接口结构与实际被调用的远程构件的接口一致,这样调用者通过内部 proxy 的接口就可以使用实际远程构件的各种方法。具体来说这会牵涉到跨进程的远程通信以及散/列集机制等问题,代理构件的使用需配合系统平台内对应的存根代码(stub)来完成,实现较为复杂,详细过程这里不再叙述^[12]。

4 结束语

构件化的 WAP 协议栈具有可动态加载、可动态升级以及方便用户的开发和调试等特点。这种模型突破了传统模式中,一旦需求更改,整个系统几乎都要改变的弊端,体现了构件化编程模型的优势。目前,对 Elastos 平台上的 WAP 协议栈的实现只是部分的实现,下一步的工作就是对已实现的 WAP 协议栈进行维护。其中还存在着一些问题,例如:对于各层的状态机的控制、还有对分割重组的处理等。这些都需要进一步的研究与完善。

参考文献:

- [1] Koretide. Elastos 2.1 Operating System Manual[EB/OL]. 2009. <http://www.koretide.com.cn>.
- [2] Koretide. CAR's Manual[EB/OL]. 2009. <http://www.koretide.com.cn>.
- [3] 王 琦. 移动互联网的基石-WAP[J]. 现代通信, 2002(11):21-22.
- [4] 杨书卷. 透视 WAP 通信协议的内含[J]. 数据通信, 2000(3):41-45.
- [5] 范绍山. 面向连接的 WAP 传输层协议[J]. 电信技术, 2000(11):22-24.
- [6] 刘 兵. WAP 协议及其应用[J]. 电脑学习, 2006(1):47-49.

第四步,令 $A = \emptyset$, 将 Red 中的属性按标签属性下标的降序排列依次加入到 A 中, 再将 R_1 中属性按标签属性下标的升序排序依次加入到 A 中, 转第二步;

第五步, 返回约简 Red。

通过算法 2 对例 1 进行属性约简过程如下:

首先, 通过函数计算得出非空标签属性集 $NEF = \{a_1, a_2, a_3, a_4\}$, $R_1 = \{a_1, a_2, a_3, a_4\}$; 将 a_4 加入到约简集 $Red = \{a_4\}$, 此时 $R_1 = \{a_1, a_2, a_3\}$, 重新生成属性集 $A = \{a_4, a_1, a_2, a_3\}$ (注: a_4 的标签值已经改变, 变为最小的一个)。

第二次循环, $NEF = \{a_4, a_1, a_2, a_3\}$, $R_1 = \{a_4, a_1, a_2, a_3\}$, 将 a_3 加入到约简集 $Red = \{a_3, a_4\}$, 此时 $R_1 = \{a_1, a_2\}$, 重新生成属性集 $A = \{a_4, a_3, a_1, a_2\}$ 。

第三次循环, $NEF = \{a_4, a_3, a_1, a_2\}$, $R_1 = \{a_4, a_3, a_1, a_2\}$, 将 a_2 加到约简集 $Red = \{a_2, a_3, a_4\}$, 此时 $R_1 = \{a_1\}$, 重新生成属性集 $A = \{a_4, a_3, a_2, a_1\}$ 。

第四次循环, $NEF = \{a_4, a_3, a_2, a_1\}$, $R_1 = \{a_4, a_3, a_2, a_1\}$, 将 a_1 加到约简集 $Red = \{a_1, a_2, a_3, a_4\}$, 此时 $R_1 = \emptyset$, 返回约简 $Red = \{a_1, a_2, a_3, a_4\}$, 结束。

通过实例对比, 证实了算法 1 和算法 2 的运算结果是一致的, 但是算法 2 的空间复杂度为 $O(|A|)$, 较算法 1 在空间复杂度上更体现出优势。

5 结束语

算法 1 在处理小数据集时, 更加方便、直观; 对于大数据集的处理, 需要改进算法 1 的效率, 降低算法 1 的空间复杂度。算法 2 在不依托于分辨矩阵的条件下, 具有更小的空间开销, 通过与灰集理论的结合, 为处理大数据集的约简提供了较好的思路。本文对于属性序下基于优势关系的灰色信息系统属性约简提供一个探讨比较的过程, 后续的工作将着力于改进算法 2 的效率, 对属性序下优势关系灰色信息系统的属性约简作进一步的研究。

参考文献:

[1] Pawlak Z. Rough set [J]. International Journal of Computer

and Information Sciences, 1982, 11: 341-356.

[2] 邓聚龙. 灰色控制系统[M]. 武汉: 华中理工大学出版社, 1993.

[3] 邓聚龙. 灰预测与灰决策[M]. 武汉: 华中科技大学出版社, 2002.

[4] 钟波. 基于灰色相容关系的灰色粗糙集及其在数据挖掘中的应用研究[D]. 长沙: 中南大学, 2008.

[5] 吴顺祥. 灰色粗糙集模型及其应用[M]. 北京: 科学出版社, 2009.

[6] 杜晓. 基于粗糙集理论与灰色理论的属性约简算法[J]. 计算机技术与发展, 2008, 18(1): 154-156.

[7] Hu Xiaohua, Cercone N. Learning in relational database: A rough set approach [J]. International Journal of Computational Intelligence, 1995, 11: 323-338.

[8] 王国胤, 于洪, 杨大春. 基于条件信息熵的决策表约简[J]. 计算机学报, 2002, 25(7): 759-766.

[9] Hoa N S. Some efficient algorithms for rough set methods [C]// Proceedings of the Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems. Granada, Spain: [s. n.], 1996: 1451-1456.

[10] Skowron A, Rauszer C. The discernibility functions matrices and functions in information systems [C]// Proceedings of the Intelligent Decision Support - Handbook of Applications and Advances of the Rough Sets Theory. Dordrecht: Kluwer Academic Publisher, 1991: 331-362.

[11] 吴顺祥. 基于灰色信息系统的优势关系及属性约简方法[J]. 厦门大学学报, 2009, 48(4): 488-492.

[12] 袁修久, 何华灿. 优势关系下广义决策约简和上近似约简[J]. 计算机工程与应用, 2006, 42(5): 4-7.

[13] 徐伟华, 张晓燕, 张文修. 优势关系下不协调目标信息系统的上近似约简[J]. 计算机工程, 2009, 35(18): 191-193.

[14] Wang Jue, Wang Ju. Reduction algorithms based on discernibility matrix: The order attributes method [J]. Journal of Computer Science and Technology, 2001, 16(6): 489-504.

[15] 胡峰, 王国胤. 属性序下的快速约简算法[J]. 计算机学报, 2007, 30(8): 1429-1435.

[16] 何利, 胡峰. 属性序下基于容差关系的约简算法[J]. 计算机应用, 2008, 28(9): 2443-2445.

(上接第 97 页)

[7] Szyperski C, Gruntz D, Murer S. Component Software: Beyond Object-Oriented Programming[M]. 北京: 电子工业出版社, 2003.

[8] 胡晓燕. 无线应用协议(WAP)的体系结构分析[J]. 科技信息(科学教研), 2007(29): 70-102.

[9] Singhal S, Bridgman T, Surranarayana L, et al. The Wireless Application Protocol[M]. [s. l.]: Addison Wesley Longman, 2001.

[10] 赵军. WAP 的现状和存在的问题[J]. 电信世界, 2008(8): 16-19.

[11] 陈榕, 刘艺平. 基于构件、中间件的因特网操作系统及跨操作系统的构件、中间件运行平台(863 课题技术鉴定文件)[R]. 上海: 科泰世纪有限公司, 2003.

[12] 上海科泰世纪有限公司. CAR 原理及应用[R]. 上海: 科泰世纪有限公司, 2007.