嵌入式Web服务模型的设计及初步实现

马 毅,高 岭,张 林
(西北大学 信息与科学技术学院,陕西 西安 710127)

摘 要:设计一个可以嵌入到其他应用系统的嵌入式Web服务器,可以为远程监控、数据传输、分布式计算提供一个平台。利用Socket接口、TCP/IP协议、HTTP协议标准,根据Java Servlet规范作为支撑理论实现了基于C语言的Servlet容器。阐述了系统的实现方案、实现标准的HTTP协议,并设计出模型,该模型给出了Servlet的基本运行环境。实现此模型,并在此模型上进行系统应用开发,实践证明此系统具有高性能、可扩展的特点。

关键词:嵌入式Web服务器;HTTP协议;Servlet;Socket
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2009)01-0029-03

Initial Implementation and Design of Embedded Web Server Model

MA Yi, GAO Ling, ZHANG Lin
(Information Science and Technology Institute, Northwest University, Xi'an 710127, China)

Abstract: Design an embedded Web server that can be embedded in other applications, it also provides a platform for remote monitoring, data transmission, and distributed computing. Use Socket interface, implement the Servlet containers based on Java Servlet specification by C language and TCP/IP protocol and the standard HTTP protocol. Focused on the realization of program and implements the HTTP protocol by Socket and system design model and the model provides the basic running environment for Servlets. Implement the model and development the system application on this model. Practice has proved that this system has the performance, scalability characteristics.

Key words: embedded Web server; HTTP; Servlet; Socket

0 引 言

计算机技术飞速发展,特别是计算机技术专业化分工明显,计算机技术的任务扩展到传统的电子系统领域,使计算机成为进入人类社会全面智能化时代的有力工具,嵌入式系统的飞速发展使得嵌入式系统得到了长足的发展。嵌入式系统的特点是面向应用、软硬件可裁减、体积小、耗电少、应用广泛。目前有现成的标准协议,如比较成熟的HTTP,TCP/IP协议[1,2],与系统提供的socket[3,4]编程,可以开发出稳定的系统。用户的界面选择的是传统的Web浏览器,而Web浏览器就是Web客户端,那么Web服务器就是看作一个嵌入式系统。文中在嵌入式Web结构设计上参照了文献[5],当前要解决的问题是在现有的传输协议下让嵌入式系统能生成包含动态实时数据的网页,使得用户和服务器可以得到实时的交互。并在此模型上开发出稳定的Web服务器[6,7]系统,为网络上各个应用系统以及智能设备提供基础的分布式计算[8,9]平台。

收稿日期:2008-04-14
基金项目:陕西省自然科学基金资助项目(2005f36)
作者简介:马 毅(1981-),男,陕西绥德人,硕士研究生,研究方向为计算机应用技术;高 岭,教授,博导,研究方向为计算机科学与技术。

1 嵌入式Web服务模型的设计概述

本系统研究、分析嵌入式Web服务及客户的形式,支持基本的Web规范,并集成了简单的分析引擎和内置若干嵌入式应用开发人员常用的基本功能。系统分析引擎是基于简单的嵌入式标记替换和动态加载函数的方式,Web网页的设计和负责业务逻辑的外部函数可独立开发,同时还可灵活地支持外部函数和嵌入式标记的对应,并可方便地精确控制与外部函数的交互行为。此Web应用服务可以独立嵌入其他系统中,如嵌入式Web服务应用到分布式应用系统中,嵌入Web服务可以达到服务方和被服务方交互的目的。

1.1 嵌入式Web服务器的设计思路

嵌入式Web服务器的出现主要是取代现有的通用Web服务器,可极大节省系统资源,简化系统管理,提高应用系统的运行效率,所以嵌入式Web服务器要能够尽量轻便灵巧,设计中要注意以下几个方面:

首先在设计原则上要保证系统的精简。在保证实

现网络通信基本功能的前提和满足系统要求的情况下尽可能地精简协议,确定协议的必要部分,省略当前不需要的。

其次是协议接口层次明确。TCP/IP 协议分布在链路层、网络层、传输层和应用层上,是分层实现的,每一层只负责处理通信过程中的一部分问题,其它层不能实现其功能。采用模块化的设计思路,如果需要修改哪个协议,只需修改相应模块的功能,其它模块不用改动。协议分层简化了程序的设计和调试,每层的协议相互独立,使协议的开发高效,易于扩展。

最后嵌入式 Web 服务器要能够嵌入到其他应用,可以应用于分布式计算系统,可以与其它系统集成,可以移植到新的平台上,可以进行安全认证配置。

1.2 协议标准的选择

嵌入式 Web 服务器技术的核心是 HTTP 引擎。HTTP 协议是 Web 应用的标准协议,其已经从 HTTP1.0 发展到 HTTP1.1[10],性能有很大改变,增加了缓存功能,两者区别就是 TCP 连接形式的改变。HTTP1.0 在每次 HTTP 请求中都需要 TCP 连接。一个典型的页面可能含有许多单独的 HTTP 请求,如基本页面请求、每个 HTML 框架请求、每个图形请求等。建立每个请求并且产生每个 TCP 连接需要占用大量的 CPU 和内存资源。而 HTTP1.1 标准可以为多个 HTTP 事务在浏览器和服务器之间只保持一个 TCP 连接,这样就大大提高网络和系统的性能。所以,在嵌入式 Web 服务器中为了得到一个稳定的用户界面而又不影响嵌入式系统的 CPU 和内存资源,应该使用 HTTP1.1 标准。

2 嵌入式 Web 服务器模型

嵌入式 Web 服务器由嵌入式 Web 应用管理单元,以及 HTTP 协议处理单元两部分构成。HTTP 协议处理单元负责解析 HTTP 请求,并且将 HTTP 请求分派到一定的 Web 应用逻辑处理。嵌入式 Web 应用管理单元负责 Web 应用、会话、虚拟文件系统以及 Servlet 的管理。其中,Web 应用是构成一个完整应用的一组 Servlet,共享变量及子应用的集合,是一个递归的定义;会话,与通常通用 Web 服务器所谓会话定义相类似,即每个客户完成某项 Web 应用业务逻辑相关的上下文的集合;虚拟文件系统则是存储静态内容或者动态生成内容的存储管理系统;Servlet 则是针对某次请求的处理模块,每个 URI 有且仅有一个 Servlet 与之对应,而每个 Servlet 则可有任意数量 URI 与之对应。整个嵌入式 Web 服务器的结构简图如图 1 所示。其中:1)HTTP 请求编码解码单元负责解析 HTTP 协议,将本次 HTTP 请求传递给 Servlet 调度单元;2)Servlet 调度单元负责管理 URI 与 Servlet 映射关系,一个 Servlet 即一个 HTTP 请求处理单元,对任意 URI,均有一个 Servlet 与之相对应;3)HTTP 响应编码解码单元,Servlet 调用该单元以构造 HTTP 响应。

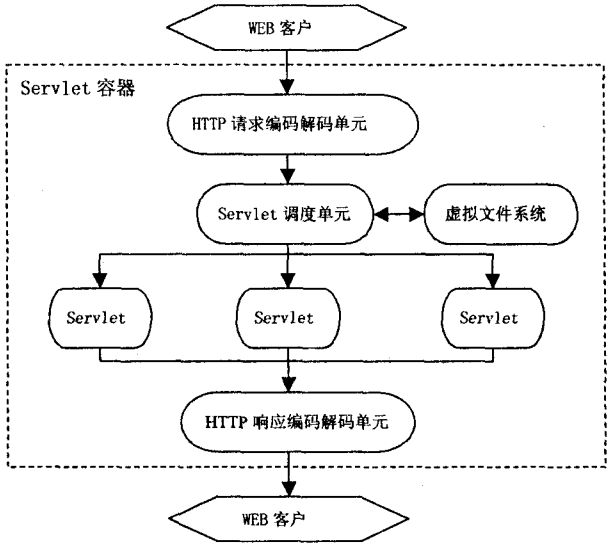

图 1 嵌入式 Web 服务器结构

2.1 HTTP 请求编码解码单元

HTTP 请求指的是从客户向服务所发送的消息,消息的首行即请求行包含了应用到所请求资源的方法,所请求资源以及所用到的 HTTP 协议版本。紧随首行的是消息头,HTTP 请求的消息头由通用头、请求头以及实体头构成,消息头头域之间用 CRLF 分割,空行 CRLF 表示消息头结束,接下来是消息体。HTTP 请求格式定义如下:

```
Request = Request-Line
          *((general-header
          | request-header
          | entity-header) CRLF)
          CRLF
          [message-body]
Request-Line = Method SP Request-URI SP HTTP-Version CRLF
```

http_request 接口是对 HTTP 请求的抽象,实现了 HTTP 请求的编码解码功能。

表 1 是该接口部分方法的介绍。

表 1 HTTP 请求接口

函数名称	函数功能
set/get_resource()	设置/获取请求所对应的资源
set/get_method()	设置/获取 HTTP 请求方法
set/get_version()	设置/获取 HTTP 协议版本
set/ get_parameter()	设置/获取 HTTP 请求参数
set/get_header()	设置/获取 HTTP 消息请求头

2.2 HTTP 响应

编码解码单元 HTTP 响应指的是从服务向客户所返回的信息,所谓返回,即客户必须向服务先发请求信息后,服务器生成一个响应。消息的首行包含了 HTTP 协议的版本,HTTP 响应状态编码,HTTP 响应状态的原因。紧随首行的是消息头,HTTP 响应的消息头由通用头、响应头以及实体头构成,消息头头域之间用 CRLF 分割,空行 CRLF 表示消息头结束,接下来是消息体。HTTP 响应格式定义如下:

```
Response = Status - Line
           *((general - header
           | response - header
           | entity - header) CRLF)
           CRLF
           [message - body ]
```

接口 http_response 是对服务响应的抽象,实现了 HTTP 响应的编码解码功能。

表 2 是对 http_response 接口的部分方法的介绍。

表 2　HTTP 响应接口

函数名称	函数功能
set/get_version()	设置/获取 HTTP 协议版本
set/get_status()	设置/获取响应状态码
set/get_reason()	设置/获取响应状态成因
set/get_header()	设置/获取 HTTP 消息响应头
set/get_content()	设置/获取响应内容

2.3 Servlet 调度单元

Servlet 的调度单元的实现形式是服务器的一个线程,由 Servlet 上下文和服务器上下文共同作用实现。服务器上下文包含了多个 Servlet 上下文,Servlet 上下文包含了多个 Servlet 。

Servlet 调度过程如下:

1) 服务器启动时负责初始化服务器上下文。

2) 新建 Servlet 上下文。

3) 初始化 Servlet 上下文。

4) 对于每个 TCP 连接的传入,服务器启动一个线程来处理该 TCP 连接。

5) 解析 HTTP 请求消息,提取请求资源。

6) 提取并执行该资源对应的 Servlet。

2.4 Servlet 接口

Servlet 是资源的生产者,即动态生成资源的代码片断。

http_servlet 接口是对 Servlet 的抽象,service()方法是 http_servlet 接口的一个重要方法,任何一个 Servlet 都必须实现该方法。如以下的 helloword_servlet 即是一个合法的 http_servlet 实现,实现了向浏览器输出"Hello World!"的功能。

```
void helloworld_servlet(http_request_t * req, http_response_t * res)
{
    char * page = NULL;
    int page_length = 0, page_size = 0 , grow_size = KB;
    byte_array_write_string(&page, &page_length, &page_size,grow_size,"<html>");
    byte_array_write_string(&page, &page_length, &page_size,grow_size,"<head>");
    byte_array_write_string(&page, &page_length, &page_size,grow_size,"<title>Hello World! </title>");
    byte_array_write_string(&page, &page_length, &page_size,grow_size,"</head>");
    byte_array_write_string(&page, &page_length, &page_size,grow_size,"<body>");
    byte_array_write_string(&page, &page_length, &page_size,grow_size,"<h1>Hello World! </h1>");
    byte_array_write_string(&page, &page_length, &page_size,grow_size,"</body>");
    byte_array_write_string(&page, &page_length, &page_size,grow_size,"</html>");
    res->set_header(res,CONTENT_TYPE,"text/html");
    res->set_header_with_int(res, CONTENT_LENGTH, page_length);
    res->set_content(res,page,page_length);
    free(page);
    res->write(res);
}
```

3 结束语

嵌入式 Web 服务有其必要性及优势,研究并分析嵌入式 Web 服务及客户的形式,如实现嵌入式 Web 服务器,支持基本的 Web 规范,并集成了简单的分析引擎和内置若干嵌入式应用开发人员常用的基本功能。为了方便应用开发,并且由于分析引擎是基于简单的嵌入式标记替换和动态加载函数的方式,Web 网页的设计和负责业务逻辑的外部函数可独立地开发,同时还可灵活地支持外部函数和嵌入式标记的对应,并可方便地精确控制与外部函数的交互行为。

嵌入式 Web 服务作为一个独立的系统,可以应用到其他应用产品中,如 Web 应用服务可以独立嵌入其他系统中,从而达到服务方和被服务方可以交互的目的,嵌入式 Web 服务 API 负责处理嵌入式 Web 客户 API 的请求的应用。

参考文献:

[1] 朱　敏,丁秋林. 基于 SOAP 的 Web 服务程序设计[J]. 计

(下转第 35 页)

4　实验仿真结果

为了说明本算法的有效性,这里采用 lena.bmp 图像来作为仿真对象(见图 5)。

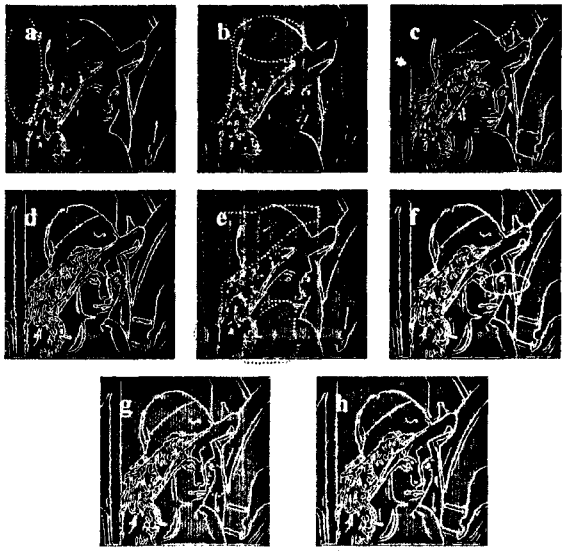

图 5　Lena 图像边缘检测仿真结果

图中,(a)为 Sobel 算子结果;(b)为 Robert 算子结果;(c)为 Log 算子结果;(d)为 Cany 算子结果;(e)为 Prewitt 算子结果;(f)为高斯滤波和 Canny 算子结果;(g)为将前六种结果进行或运算的结果;(h)为本算法结果。

由图 5 可以看出常用的 Sobel 算子边缘检测、Robert 算子边缘检测、Log 算子边缘检测以及 Prewitt 算子边缘检测结果都很不理想,有部分边缘没有检测出来。这里 Sobel 算子边缘检测时使用的阈值为 0.12365,Robert 算子边缘检测使用的阈值为 0.13899,Log 算子边缘检测时使用的阈值为 0.0059765,Prewitt 算子边缘检测使用的阈值为 0.12049。而 Canny 算子边缘检测结果(如图 5 中的 d)相比前述的效果好一些,因为采用的阈值不是一个,而是一个阈值域[0.05625, 0.14063]。(f)子图为经过高斯滤波器将图像平滑以后再用 Canny 算子进行边缘检测的结果,可以看出边缘信息比较明显,所有的边缘都可以清晰地看到,但是还有很多的噪音。(g)子图为将前面的所有的结果进行逻辑运算(或,与)后得到的结果,效果优良了很多,噪音也少了很多,所以在逻辑运算对二值图像的优化能力还是比较强的。(h)是文中所提出的算法边缘检测结果,相比之下,边缘信息比较完整、清晰,足以证明本算法的有效性和实用性。

5　结束语

提出一种效果较好的边缘检测算法,本算法将具有良好边缘检测能力的小波变换和灰度数学形态学结合起来,并且在二值化的时候采用最优阈值,使得边缘检测结果更为清晰。由实验结果可以看出,比较常用的几种边缘检测方法,本算法具有边缘细节丰富、边缘轮廓准确和抗噪强等优点。

参考文献:

[1] Gonzalez R C, Woods R E, Eddin S L. 数字图像处理(Matlab 版)[M].阮秋琦,译.北京:电子工业出版社,2006.

[2] 林世毅,苏广川,陈　东,等. 基于小波变换和数学形态学的边缘检测法[J]. 仪表仪器学报,2004,25(4):685-687.

[3] 孙圣和,陆哲明,牛夏牧,等. 数字水印技术及应用[M]. 北京:科学出版社, 2004.

[4] 章毓晋. 图像工程(上册)[M].第 2 版.北京:清华大学出版社, 2006.

[5] Pei S C, Chen F C. Hierarchical Image Representation by Mathematical Morphology Subband Decomposition[J]. Pattern Recogit Lettm,1995,16:183-192.

[6] 范立南,韩晓微,张广洲. 图像处理与模式识别[M].北京:科学出版社,2007.

[7] 林　卉,杜培军,舒　宁,等. 基于多结构元素的数学形态学的遥感图像边缘检测方法[J]. 遥感技术与应用,2004,19(2):114-118.

(上接第 31 页)

算机应用,2003,23(52):173-174.

[2] Stevens W R. TCP/IP Illustrated, Volume 1: The Protocols [M].北京:机械工业出版社,2004.

[3] 张曾科.计算机网络[M].北京:清华大学出版社,2003.

[4] Strvens R W. UNIX network programming volume 1: The Sockets Networking API[M]. Third Edition.北京:机械工业出版社,2004.

[5] Hong-Taek Ju.Embedded Web server architecture for Web-based element and network management [D]. Korea: DP&NM Lab,CSE,POSTECH,2001.

[6] Jones M T. 嵌入式系统 TCP/IP 应用层协议[M].路晓村等译.北京:电子工业出版社,2003.

[7] Tabara D, Rijanto H, Sabbattini B. Embedded Web technology: adding a new dimension to protection and control[J]. ABB Review,2001(2):16-19.

[8] Liu M L. 分布式计算原理与应用[M].顾铁成 等译.北京:清华大学出版社,2004.

[9] Spitzner L. Know your Enemy: Passive Fingerpringing[J/OL].2002-03. http://project.honeynet.org/paper/figer.

[10] W3C. Hypertext Transfer Protocol[S]. RFC2616,1999.