“和欣”浏览器网络子系统的设计与实现

Design and Implementation of Network Subsystem of ElastOS Web Browser

（申请清华大学工学硕士学位论文）

培 养 单 位 ： 计算机科学与技术系

学　　科 ： 计算机科学与技术

研 究 生 ： 李洪涛

指 导 教 师 ： 殷 人 昆 教 授

联合指导教师 ：

二〇〇五年五月

和欣浏览器网络子系统的设计与实现

李洪涛

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：_____ 导师签名：_____

日　　期：_____ 日　　期：_____

摘 要

随着计算机及相关技术的发展，计算变得越来越自由，在资源使用方面也越来越灵活，嵌入式系统得到广泛的应用。嵌入式浏览器是 Internet 技术在嵌入式系统中的关键应用，是嵌入式信息设备的核心技术之一。“和欣”操作系统是未来的网络操作系统，通过在二进制级引入元数据，提出了嵌入式领域的分布式解决方案。浏览器在“和欣”网络编程模型中扮演着重要的角色，将成为“和欣”操作系统的主要客户端，为用户提供服务。网络子系统是浏览器中的重要组成部分。

论文首先提出了“和欣”操作系统上浏览器体系架构设计，然后提出了基于面向对象技术的网络子系统的设计方案,该设计方案参考已有浏览器及相关 Internet 技术。论文详细阐述了“和欣”浏览器网络子系统的体系结构和设计思路，阐述了网络子系统的组成部分。基于设计方案，我们已经实现了一个功能完备的嵌入式浏览器网络子系统，充分考虑了可裁减性和高效性，该系统面向嵌入式领域、稳定、实用、功能完善。

在已有的设计与实现的基础上，论文进一步研究了嵌入式浏览器的网络子系统的构件化工作，提出了采用 CAR 构件技术实现浏览器网络子系统的设计方案，讨论了采用构件化思想实现浏览器网络子系统涉及的问题，包括构件的划分，构件接口的设计，协议构件的动态加载等。在设计的基础上，在“和欣”操作系统上实现了基于 CAR 构件技术的浏览器网络子系统，该系统具有良好的体系架构、支持协议的扩展与协议处理构件的动态加载的特点，方便用户增加新的通信协议的支持，同时有利于用户或者开发者积累更多可复用的构件。

最后，论文分析总结了嵌入式浏览器的网络子系统的构件化设计与实现工作，并提出进一步的工作展望。

关键词：“和欣”浏览器 网络协议 构件技术 CAR

Abstract

With the development of computer technology, the embedded systems are applying in more and more fields. Embedded web browser is the critical application of the Internet technique in embedded systems. And it is also one of the central technology of embedded information equipments。ElastOS operating system is the network operating system and by introducing data during the binary stage, it suggests a distributed solution in the embedded field. Browser plays an important role in the ElastOS network-programming model and will be the main application in ElastOS operating system. The network subsystem is a important component of web browser.

The thesis introduces firstly the framework design of ElastOS web browser, then the Object Oriented-based network subsystem design on the basis of analysis and research notable browsers and related Internet technologies that have been developed. The thesis details the architecture and the design consideration and specifies the components of ElastOS Browser network subsystem. Based on the design and under the fully thinking of flexibility and the high efficiency, we have completed the embedded browser network subsystem with the required function. The subsystem orients to the embedded fields and it has a lot of advantages, such as stableness and practicality and it's perfectly function.

Upon the design and the completion ,the thesis makes further studying the component of the embedded browser network subsystem and discuss the problems of the way based on which the CAR component technology to realize the browser network subsystem。The problems include the division of the component, the design of component's interface and dynamically loading network protocol components, etc. And put forward a design that adopts the CAR component technology and we have completed it on the ElastOS Operate System. The subsystem has a good architecture and supports the protocol extension and dynamically loading network protocol component, profits the user to add some new protocols and is good to the users or the developers to accumulate the reusable component components. At last, the thesis

summarizes the design and implementation of browser network subsystem.

Key words: ElastOS operating system, web browser, CAR，component technology

目 录

第 1 章　引言

1.1　背景介绍

计算机技术已经广泛应用到科学研究，工程设计，军事技术，各类产业和商业文化艺术、娱乐业及人们的日常生活，每一个人都在日常生活中利用计算机技术提供的服务。其中，嵌入式系统扮演着十分重要的角色。

嵌入式系统在 20 世纪 60 年代后期，先在通信领域中出现。七、八十年代后，被用在工业领域等。目前，在信息家电、移动通信、手持设备、以及在工业控制领域中嵌入式系统都得到了了广泛的应用。

嵌入式系统以应用为主要目的，将计算机技术、半导体技术、控制技术、电子技术等结合在一起的产物。它强调软、硬件的协同性和整合性，软、硬件易于剪裁，适用于对功能、可靠性、成本、体积、功耗有严格要求的计算机系统。在具体的应用中，主要实现实时控制、监视、管理移动计算机、数据处理以及辅助其他设备运转，完成各种自动化处理的任务。[1][2]

嵌入式技术的发展，大致经历了以下 4 个阶段。

第一个阶段是以单芯片为核心的可编程控制器形式的系统，同时具有与监测、伺服、指示设备相配合的功能。

第二个阶段是嵌入式 CPU 为基础、以简单操作系统为核心的嵌入式系统；

第三个阶段是以嵌入式操作系统为标志的嵌入式系统。这一个阶段系统的主要特点是：嵌入式操作系统能运行在各种不同类型的微处理器上，兼容性好；操作系统内核精巧、效率高，并且具有高度的模块化和扩展性；具备文件和目录管理、设备支持、多任务、网络支持、图形窗口以及用户界面等功能；具有大量的应用程序接口（API），开发应用程序简单；嵌入式应用软件丰富等特点。

第四个阶段是嵌入式设备与网络的结合。当前正处于嵌入式技术发展的第四个阶段，现阶段嵌入式系统的发展以基于Internet为标志，正在处于迅速发展的阶段。目前大多数嵌入式系统还孤立与Internet之外，但随着Internet的发展以及Internet技术与信息家电、工业控制技术等结合日益密切，嵌入式设备与Internet的结合将代表着嵌入式技术的未来。[3][4]

随着嵌入式设备与网络的日益结合，网络成为嵌入式系统中的重要组成部分，并且需求越来越大。针对嵌入式系统的浏览器应运而生。

嵌入式浏览器还没有一个准确、严格的定义，但从其可以完成的功能来看，可以从两个方面进行描述。第一，它必须是一个网络信息浏览器，必须支持 HTTP 或者 WAP 等其它传输协议、支持 HTML、JavaScript 或扩展 XML、WML 等标记语言，可以完成网页的浏览功能；第二，这个浏览器必须适合在非 PC 的嵌入式信息设备中存在、运行并完整实现通讯传输协议、标记语言所规定的功能，必须能够根据嵌入式设备的多样性需要而方便的进行裁减和修改，并满足信息设备使用者对获取文字、图像、声音、视频等信息的需求。

嵌入式浏览器可以广泛应用与 Internet－TV、iDVD、Web 终端、数字电视机顶盒、掌上电脑等各种信息电器和便携式网络终端中。

由于嵌入式系统、硬件设备的多样性与复杂性，嵌入式浏览器的有自己的特殊性。

1 从整体看，嵌入式浏览器具有多样性

嵌入式系统是针对各种具体设备设计的，设备的多样性决定了系统的多样性，也决定了嵌入式浏览器的多样性。根据具体应用的需求，对嵌入式浏览器定制是必须得，在浏览器的设计与实现中，要考虑到能方便地进行修改、裁减。

同时，与桌面系统不同，嵌入式浏览器市场没有出现处于垄断地位的浏览器。许多产品都是针对特定设备开发，不具有全部功能，不能通用。

2 从软件、硬件看，嵌入式浏览器要求特殊

从硬件来看，嵌入式浏览器要充分考虑到嵌入式设备的特殊性；从软件来看，嵌入式浏览器要考到功能需求。浏览器的基本功能就是要能够浏览各种网页，嵌入式浏览器也是如此，因此，它要支持 HTML 标记语言、HTTP 协议、英文显示、包括中文在内的双字节语言显示。并且，由于使用者一般都有获取文字、图像、声音、视频等信息的需求，嵌入式浏览器也要支持这些格式。

3 从系统看，嵌入式浏览器将是嵌入式系统的核心软件

在传统的计算机系统中，操作系统处于硬件环境和应用程序的中间层，是整个计算机系统中的核心部分，所有的硬件设备都通过驱动程序由操作系统统一管理。而在操作系统之上又会存在很多应用程序，操作系统必须提供很多应

用程序接口(API)来支持应用程序的运行、这就导致了操作系统越来越复杂，越来越庞大、这时候浏览器仅仅是运行于操作系统之上的一个应用软件。

在嵌入式系统特别是大家普遍关注的可以作为 Internet 网络终端的嵌入式系统中，嵌入式浏览器可能成为嵌入式操作系统最主要的应用软件，甚至有可能是唯一的直接支持的应用软件。其它应用如 MP3 播放器、MPEG 视频、交互式游戏、IP 电话、ICQ、股票接收分析、设置远程教育、办公软件都可以通过 Plug－in 接口或者直接用 HTML、JavaScript 及其扩展语言来实现。这样在实现互联网应用的嵌入式系统中，嵌入式操作系统的核心地位和图形用户界面的开发接口作用在逐渐淡化，嵌入式操作系统成为了浏览器和硬件系统之间的驱动和底层管理软件，而嵌入式浏览器将成为核心，向上层应用软件提供运行平台，支持各种应用。在这种情况下浏览器将成为嵌入式软件的核心，并且成为应用软件开发的平台。

1.2　国内外研究的状况

浏览器技术在沉寂了一段时间之后，随着人们不满足目前B/S模式所提供的用户体验，目前浏览器在提供网络编程解决方案的同时，本身也在向提供更好的用户体验。目前已经有很好的浏览器项目和技术出现，而且有很多开源的项目，比如Firefox[5]、Dillo[6]等，为研究提供了参考。

嵌入式浏览器的技术也正在不断的向前发展，除了继续跟随HTML，HTTP，SSL，JavaScript等协议或语言的新版本，继续升级以及支持更多的应用外，还在继续拓展其他的应用模式和领域。如浏览器技术与数字电视机机顶盒技术集成，实现完整的数字电视软件平台，浏览器技术和Java技术的结合，解决实时性强的动态视频处理。把浏览器技术应用到更多传统的电器领域，以提供交互式的应用等等。这些都是很好的发展方向，有着很好的发展前景[1][2]。

以下是对国内外比较著名的嵌入式浏览器的简单介绍：

1 Internet Explorer for WinCE

IE 的 Windows CE 版。WinCE 对硬件配置要求比较高，目前主要用于高端终端市场。随着微软势力在掌上终端领域的蔓延，WinCE 版的 IE 必将大行其道。

2 Access NetFront。

Access 是日本的浏览器厂商，占据了大部份的日本浏览器市场，其产品广泛应用于手持设备、机顶盒、游戏机等产品。NetFront 支持 XHTML 1.0、HTML4.01，支持 frames，cookies，JavaScript，CSS，DOM，Dynamic HTML，插件和 SSL2.0/3.0。和 IE 一样，对硬件配置要求比较高，目前主要用于高端终端市场。

3 中科红旗的嵌入式浏览器。

中科红旗和全球第三大浏览器厂商 Opera 公司达成战略合作协议。双方将针对中国嵌入式市场的需求特点，将基于嵌入式 Linux 的 Opera 浏览器进行裁减和定制，并与红旗嵌入式 Linux 系统相捆绑，形成一套完整强大的嵌入式浏览器解决方案。由于这两家公司都是很有实力的大公司，推出的嵌入式浏览器具有强大的市场竞争力，目前的版本对系统性能要求高，主要用于高端 PDA 产品。

4 iPanel（茁壮）浏览器特点

iPanel 嵌入式浏览器支持 HTML4.0、HTTP1.1、Images、Cookies、JavaScript1.3、WAV 以及 SSL3.0，并支持数字电视传输标准协议 DVB 和 TVHTML 语言格式。iPanel 嵌入式浏览器专门针对电视显示作了优化处理，采用独特的字体识别系统和抗闪烁算法，有效地降低利用电视机作为显示终端的闪烁现象，优化电视显示效果。可移植性高。

5 Dillo 浏览器

Dillo 浏览器是出色的开源浏览器项目，目标代码很少，速度很快，在一些嵌入式领域已经有一些应用。

浏览器进一步的发展方向为 RIA，RIA 方面的技术有 XUL, Avlaon，Flex 等。

网络子系统是浏览器中的重要组成部分。网络子系统的高效与灵活配置是嵌入式浏览器对网络子系统的要求。同时，作为在嵌入式领域，基于构件技术实现浏览器是一个比较新的课题。浏览器的网络子系统的构件化研究，将对实现基于构件技术的嵌入式浏览器提供参考。

1.3　课题研究的意义

嵌入式领域的发展，对嵌入式操作系统提出了更高的要求。在桌面系统上，以 Java、.NET 为代表的下一代网络编程模型成为主流。新的网络编程模型通过引入元数据与反射机制，很大地提高了程序的动态特性，为解决分布式应用提供了强大的支持。“和欣”运行平台，通过在 C/C++编译产生的二进制模块代码中引入元数据，提出了支持网络移动计算的方案，并且由于是在二进制级进行包装，效率和 C/C++相同，非常适合嵌入式领域。“和欣”运行平台是面向网络编程的下一代嵌入式平台。

浏览器的产生，带来了网络革命，使得世界各地网络上的资源，可以以一种简单通用的方式被访问。经过了一段长时间的沉寂，随着微软的 Smart Client 的提出，开源浏览器 Firefox 的发展，浏览器技术有了新的进步。浏览器将在传统的信息访问显示的基础上，为客户提供更丰富的体验，也就是富互联网应用（RIA）。在嵌入式领域，随着 3G 时代的到来，网络带宽将获得很大的增加，对嵌入式应用领域将提出更高的要求，要求嵌入式应用为用户提供更丰富的网络服务，这些发展，对嵌入式浏览器提出了更高的要求。

“和欣”操作系统是为 3G 量身定做的操作系统，其中微内核结构，“点击运行”，自滚动下载等理念充分体现了网络时代 3G 操作系统的特色。“和欣”操作系统的这些特征为提供更好的嵌入式应用的体验创造了条件。

在下一代的嵌入式平台“和欣”系统中，“浏览器”将实现运算透明化的目标，达到网络就是计算机的目的，实现“程序就是数据”的设想。在“和欣”系统上，“和欣”浏览器将实现运算的透明化，通过在浏览器中实现“点击运行”机制与增加浏览器的 CAR 构件支持，“和欣”浏览器在计算机无法在本地找到运行服务所需要的程序时，将在网络上实现透明地搜索、加载，完成用户所需要的服务，从而实现智能客户端，提供 Rich Internet 应用的解决方案。

“和欣”浏览器课题属于国家“863”项目“和欣操作系统”。该课题主要研究浏览器技术并在“和欣”操作系统上进行实现，一方面满足目前基于数据的访问，提供在“和欣”操作系统上访问目前的 Internet 资源的方法；另一方面，实现“程序就是数据”，实现新的 Internet 应用的解决方案。论文课题隶属于“和

欣”浏览器课题。在“和欣”浏览器中，网络子系统是其中的重要组成部分，是实现网络服务的基础。论文课题将研究“和欣”浏览器中网络子系统的设计与实现，并进行“和欣”浏览器网络子系统的 CAR 构件化进行研究，探讨如何实现基于 CAR 构件技术的浏览器网络构件库，为“和欣”浏览器与“和欣”操作系统提供网络服务，为“和欣”嵌入式浏览器的构件化实现与“和欣”浏览器的 CAR 构件化支持方案，为“和欣”浏览器实现“点击运行”，提供浏览器网络子系统方面的实践。

1.4　论文各部分的主要内容

嵌入式浏览器的研究，是嵌入式系统应用开发中的关键技术，而网络部分的实现又是网络时代嵌入式浏览器中重要的组成部分。本文以国家“863”软件专项“基于中间件技术的 Internet 嵌入式操作系统及跨操作系统中间件运行平台”为背景，探讨了嵌入式浏览器中如何设计与实现高效、灵活的网络系统，并在此基础上讨论如何实现基于 CAR 构件技术的浏览器网络系统。在“和欣”操作系统上，我们已经实现了嵌入式浏览器 ElaScope，并且实现了基于 CAR 构件技术的浏览器网络子系统。

第一章：　引言

介绍嵌入式系统浏览器的特点以及嵌入式浏览器的国内外发展现状。并介绍了课题研究的意义。

第二章：　相关技术与概念

总体介绍浏览器技术，介绍浏览器涉及的重要概念。并介绍了构件技术的特点及发展，并具体介绍了 COM 技术，以及由 COM 技术发展而来的 CAR 技术。

第三章：“和欣”浏览器体系架构设计

总体介绍我们在“和欣”操作系统上设计与实现的嵌入式浏览器 ElaScope 的体系架构。本章将首先介绍 ElaScope 浏览器的体系架构，然后介绍体系架构中 ElaScope 浏览器中浏览器引擎部分的模块通信机制以及 ElaScope 浏览器的基本工作流程，最后本章将简单介绍我们在“和欣”系统上 ElaScope 浏览器的实现。

第四章：浏览器网络子系统的设计与实现

本章首先介绍 ElaScope 浏览器中网络子系统的设计，该设计参考了现有的浏览器的网络子系统的设计，采用了面向对象技术。然后给出了网络子系统的具体实现。

第五章：浏览器网络子系统的构件化研究

第四章的基础上，讨论嵌入式浏览器网络子系统的构件化问题。首先介绍“和欣”浏览器的构件化目的，然后讨论如何完成构件化工作，在此基础上，给出了“和欣”浏览器构件化网络子系统的设计与实现，并讨论嵌入式浏览器的构件化问题。

第六章：工作总结与展望

总结本论文的研究工作，并对以后的研究工作提出了展望。

第 2 章　相关技术与概念

2.1　嵌入式浏览器

2.1.1　浏览器的基本功能模块

网络技术的飞速发展，使得网络成为了现代计算机系统中不可缺少的一部分。为了能够以更方便灵活的充分利用网络资源，产生了浏览器(browser)。浏览器功能强大、界面友好、操作简单，普通用户通过它，对网络资源进行透明操作，在浩如烟海的网络中畅游。

和通用的PC浏览器的一样，嵌入式浏览器的基本功能也是使用网络传输协议（HTTP协议,WAP协议等），通过网络，从服务器上得到HTML资源，对其进行解释、分析，并最后将其显示在屏幕上。现代浏览器的功能非常的强大，并且仍在不断的发展支持新的功能如对动态网页的支持，多媒体信息的支持等。[7]

下面是浏览器的核心功能的基本结构图（功能模块图示），从图 2.1 中可以看到它们之间的关系。

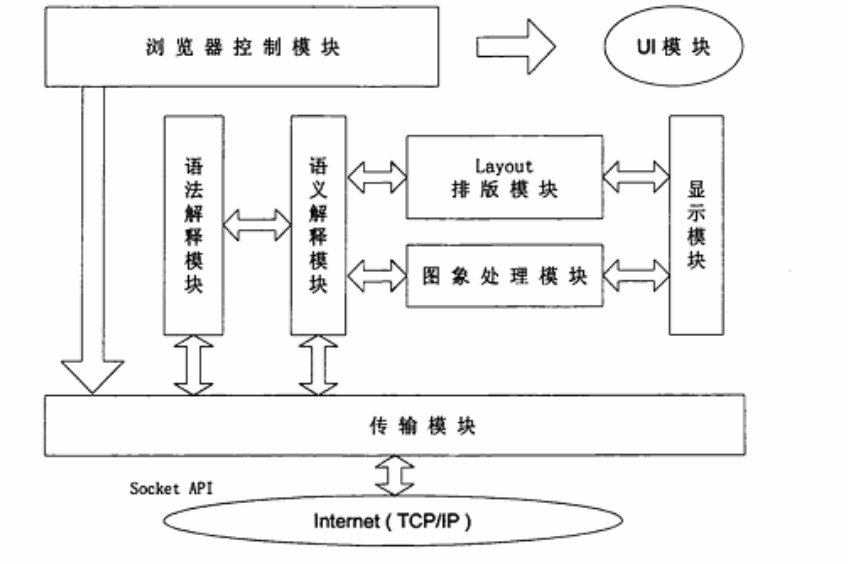

图 2.1 浏览器基本结构

从功能模块上划分，一个浏览器的基本功能的模块由以下几个模块组成：传输模块、语法语义解释模块、布局模块、显示模块、用户界面模块、媒体对

象处理模块（主要是图像处理模块）等几部分。除了用户界面模块以外，其它的模块为浏览器的核心模块，完成浏览器的基本功能。用户界面则提供给使用者更为友好的服务。在实现过程中，这些模块通常是紧密相连的，很难被严格的区分开来。

传输模块。模块负责完成本地文件的读写、网络数据的传输。在网络传输功能中，浏览器通过传输模块向服务器发出请求，并由它接收服务器的应答信息和数据，传送给浏览器的其它部分。传输模块是浏览器和网络的接口部分，是一个相对较独立的模块。网络的操作依赖一些标准的协议族。HTTP 协议是最常用的传输协议，在协议的相关文档中有详细的规定。

HTML 语法语义解释模块。HTML 是一种标记语言，浏览器接收到 HTML 文本后，HTML 文本经过语法分析生成语法树，对应网页结构。生成的语法树将贯穿整个页面的排版和显示的过程，只有网页被删除时，语法树才被同时销毁。

布局模块。经过语法分析得到语法树之后，需要进行排版（Layout）。如果把每个可显示的 HTML 对象所占地面积当成一个矩形，排版的过程就是将这些大小不等的矩形，堆放在一起，找到它们各自的位置。Layout 模块对 HTML 语法语义解释的结果进行处理，处理的结果用树形结构组织，该树的各个节点就是要显示的对象，它的数据结构里，带有改对象的显示坐标等属性信息。这棵树将被提交给显示模块，由显示模块将其显示在屏幕上。

显示模块，也称为 rendering 模块。当 Layout 确定了树上的 HTML 元素后，显示模块将接收到 Layout 模块的结果，根据显示的具体情况，对相应的数据进行调整，使页面在窗口上能够正确的显示在合适的位置上。一般显示模块和 Layout 模块是紧密相关的。

用户界面模块（UI 模块）。模块处理和界面相关的工作，如窗体显示、窗体元素显示。它直接和用户进行交互，用户通过 UI 模块来使用浏览器的各种功能。它要为用户提供方便的使用方法，要比较友好、易于操作。这部分比较独立，各个浏览器根据需要，采用不同的图形系统进行支持。

其它模块。现在的绝大部分嵌入式浏览器都是图形浏览器。既可以浏览图形又可以浏览文本。大部分的现代浏览器还能提供多媒体信息，包括声音、图

像等。不过，声音和图像中的一些格式需要 plug-ins。因此在基本的功能模块之外，现代的浏览器还有许多其它的功能模块，支持更加强大的功能。

2.1.2　浏览器涉及的重要概念

2.1.2.1　WWW 编程模型

Internet万维网（WWW）提供了一种非常灵活强大的编程模型，如图 2-1。应用和内容采用标准的数据格式来描述，通过Web浏览器来浏览。浏览器实际上是一个网络应用。浏览器发送请求给网络服务器，请求某一个数据对象，网络服务器响应请求，向网络客户端（浏览器）发送使用标准数据格式编码的响应数据[8]。

万维网标准定义了许多标准，用于建构通用应用环境，包括：

- 标准的命名模型:万维网上的所有服务器和资源内容都必须使用Internet标准统一资源定位符（URL）来命名(参考 RFC2396);
- 标准的内容类型：万维网上所有资源内容都有一个专门的类型，借助于类型标志，浏览器可以对万维网上的资源内容做出正确的处理(参考 RFC2045，RFC2048);
- 标准的内容格式：所有的 Web 浏览器都支持一组标准的内容格式，如超文本传输协议（HTML）(参考 HTML4 规范)，脚本语言等;
- 标准的传输协议：标准的网络传输协议使任何 Web 浏览器都可以同任何服务器进行交互。万维网上最常见的网络传输协议是工作在 TCP/IP 协议栈之上的超文本传输协议（HTTP）(参考 RFC2616);

浏览器作为应用程序，其底层需要协议栈来支撑，完成浏览器与服务器的交互过程。通常嵌入式浏览器直接交互的协议是 HTTP/W-HTTP 协议和无线会话协议（WSP）。通常浏览器内部的传输协议模块会对使用的协议进行封装，抽象协议接口，使得浏览器通过统一的接口架在协议栈上，协议栈对浏览器来说是透明的。

2.1.2.2　HTTP 协议

在传输协议中应用最为广泛的是 HTTP 协议。HTTP 是一个属于应用层的面向对象的协议，适用于分布式超媒体信息系统，是 WWW 的核心。它于 1990 年提出，经过几年的使用于发展，得到不断地完善和扩展。目前在 WWW 中使用的是 HTTP/1.1 版。作为新的传输协议 HTTP-NG（Next Generation of HTTP）的建议已经提出。

HTTP 协议具有以下特点：

- 支持客户端/服务器模式
- 简单快速。协议定义客户向服务器请求服务的方法，常用的方法有 GET、HEAD、POST 等，每种方法定义了客户端与服务器的具体交换操作。
- 协议简单。HTTP 服务器的程序规模小，通信速度快。
- 灵活。HTTP 允许传输任意类型的数据对象、正在传输的类型由 Content-Type 加以标记。
- 连接方式灵活。允许无连接的方式也运行 Keep-Alive 方式的连接。
- 无状态。HTTP 协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。每个客户端与服务器的连接都是一次独立的连接。如果后续处理需要前面的信息，则必须重传。另一方面，由于服务器不需要先去信息时，具有较快的应答速度。

HTTP 协议采用请求/回应的方式运作。当客户端与服务器通过 HTTP 协议通信时，客户端将首先建立一个 HTTP 连接，并向 HTTP 服务器发送一个请求消息（Request Message）。当服务器接收到请求消息以后，将返回一个 HTTP 回应消息（Response Message），其中将包含所请求的资源。

HTTP 协议中应用最广泛的版本是 HTTP/1.1，相对于 HTTP/1.0, HTTP/1.1 的改进主要表现在三个方面：HTTP/1.1 不同于 HTTP/1.0 需要为每一个 WEB 页面的获取建立一个新的连接，而是采用了可持续连接（Persistent Connection）的方法，即客户端建立持续连接后，进行第一个请求应答后并不立即关闭连接，而是可以进行多次请求应答后再关闭连接；HTTP/1.1 增加了对缓存的支持；HTTP/1.1 增加了 chunked 的传输编码方式。这样在服务器端，对于动态产生的

页面，在总长度未知的情况下，仍然可以发送响应的数据。但客户端必须进行相应的处理后才能得到原始的数据。

HTTP 协议是应用层协议，其具体实现往往与具体应用结合，根据应用的实际需要来做一些剪裁和优化处理。在嵌入式浏览器中，网络实现简化的 HTTP 协议，例如简化在 Dillo 浏览器中，对 HTTP 协议方法与缓存管理都做了简化。

2.1.2.2　HTML 语言

HTML（Hyper Text Markup Language 超文本标记语言）是一种用来制作超文本文档的简单标记语言。用 HTML 编写的超文本文档称为 HTML 文档，它能独立于各种操作系统平台（如 UNIX，WINDOWS 等）。自 1990 年以来 HTML 就一直被用作 World Wide Web 上的信息表示语言，用于描述

从语法上看，HTML 有两个基本的组成部分：标记（Tag）和数据。标记由“<”和“>”标记，有标记名和属性，如：<img src=”a.gif” height=30>，HTML 中常用的标记大约有 150 种。数据是 HTML 文档中标记以外的部分。从结构上来看，一个 HTML 文档由两部份组成：HTML HEAD 和 HTML BODY 组成。HTML HEAD 包含了一些一般信息，它的内容不作为文档的一部分显示，它使用的标记也只有几个。HTML BODY 中包含了文档要显示的所有内容，可以使用各种标记。HTML 发展至今，主要有：HTML2.0、HTML3.2、HTML4.0、HTML4.01 等几个版本。

2.2　CAR 构件技术

2.2.1　构件技术综述

软件复用（或软件重用）是指充分利用过去软件开发中积累的成果、知识和经验，去发新的软件系统，使人们在新系统的开发中着重于解决出现的新问题、满足新需求，从而避免或减少软件开发中的重复劳动。

近几年来，以面向对象为基础发展起来的软件构件技术，从某种层面上说，克服了以往的面向对象技术的某些缺陷，提高了软件复用程度。[9]

面向对象是将系统划分为多个对象（数据和相关方法的结合体），通过对象之间的通信和互操作形成耦合系统，它提供了客观世界与软件系统进行对应的编程思维方式，其具体技术包括：（1）封装性；（2）多态性；（3）继承性。

构件技术是在面向对象技术的基础上发展起来的。面向对象技术通过类的封装和继承成功实现代码级的复用。类和封装性，实现数据抽象和信息隐蔽，继承性，提高了代码复用性。但是面向对象的复用脱离不了代码复用的本质，对象之间的关系在编译时被固定，模块之间的关系是静态的，无法解决软件动态升级和软件模块动态替换。

构件技术通过二进制的封装以及动态链接技术解决软件的动态升级和软件的动态替换问题。面向构件技术对一组类的组合进行封装，并代表完成一个或多个功能的特定服务，同时为用户提供多个接口。整个构件隐藏了具体的实现，只用接口提供服务。这样，在不同层次上，构件均可以将底层多个逻辑组合成高层次上的粒度更大的新构件。构件之间通过约定的接口进行数据交换和信息传递，构件的位置是相互透明的，可以在同一个用户进程空间，也可以在不同的用户进程空间，甚至在不同的机器上，而且不同的构件可以用不同的语言编写，只要它们符合事先约定的构件规范。

构件是具有强制性、封装性、透明性、互操作性和通用性的软件单元。构件的粒度可大可小，可以是一个简单的按钮实现模型，也可以是潮流计算、状态估计等应用。构件使用与实现语言无关的接口定义语言（IDL）来定义接口。IDL 文件描述了数据类型、操作和对象，客户通过它来构造一个请求，服务器则为一个指定对象的实现提供这些数据类型、操作和对象。

目前有三种比较成熟的构件技术，它们是微软公司提出的组件对象模型（Component Object Model，COM）及其分布式扩展DCOM(Distributed COM)[10]、对象管理组织OMG的通用对象请求代理体系结构CORBA(Common Object Request Broker Architecture)[11]以及SUN的JavaBeans组件技术[12]。

2.2.2 COM技术[13][14][15][16][17]

COM 技术是由 Microsoft 提出的开放的构件标准，是一种以构件作为发布单元的对象模型，这种模型使软件构件可以用一种统一的方式进行交互，有很强

的扩充和扩展能力。COM 规定了对象模型和编程要求，使 COM 对象可以与其他对象相互操作。这些对象可以用不同的语言实现，其结构也可以不同。基于 COM，微软进一步将 OLE 技术发展到 OLE2。其中 COM 实现了 OLE 对象之间的底层通信工作，其作用类似于 CORBA/ORB。在 OLE2 中出现了拖—放技术以及 OLE 自动化。

COM 标准包括规范和实现两大部分，规范定义了组件和组件之间通信的机制，这些规范不依赖于任何特定的语言和操作系统，只要按照该规范，任何语言都可使用；COM 标准的实现部分是 COM 库，COM 库为规范的具体实现提供了一些核心服务。

COM 规范包括 COM 核心、结构化存储、统一数据传输、智能命名和系统级的实现（COM 库）。COM 核心规定了构件对象与客户通过二进制接口标准进行交互的原则，结构化存储定义了复合文档的存储格式以及创建文档的接口，统一数据传输约定了构件之间数据交换的标准接口，智能命名给予对象一个系统可识别的唯一标识。

在 COM 模型中，对象本身对于客户来说是不可见的，客户请求服务时，只能通过接口进行。构件对象模型 COM 内容复杂，主要包括：

接口：COM 对象间互相调用的一组语义相关的接口，每个接口有一个 128 位的唯一标识（UUID）。所有的接口皆直接或间接地从 IUnknown 接口继承而来，IUnknown 接口包括 QueryInterface、AddRef 和 Release。

COM 对象：即 CoClass 实例，提供接口的具体服务。CoClass 是一个或多个 COM 接口的实现。对 COM 对象的调用是通过一个指向其接口的指针实现的。

COM 服务器：是一个程序或库，包含 COM 对象，向客户提供服务。

类工厂（Class Factory）：用于创建、注册 COM 对象的特殊对象，为实例化 CoClass 提供一种标准机制。对 CoClass 进行实例化是通过调用全局 Windows API 函数 CoGetClassObject 或 CoCreateInstance 实现的。

类型库（Type Library）：一个二进制资源文件，包含 COM 服务器中对象与接口的类型信息，可以从 MIDL 或 ODL 转换而来。

COM 库包含了一些核心的系统级代码，使得对象和客户之间可以通过接口在二进制代码级进行交互。COM 库可以保证所有的组件按照统一的方式进行交

互操作，而且它使我们在编写 COM 应用时，可以不用编写为进行 COM 通信而必须的大量基础代码，而是直接利用 COM 库提供的 API 进行编程，从而大大加快了开发的速度。COM 库的另一个好处是，它往往实现了更多的特性，我们可以充分享受这些特性。

COM 技术有两大缺陷，一是 COM 没有试图去描述构件之间的依赖关系，在构件广泛应用的今天，不同构件之间的依赖越来越多，如果没有构件之间的依赖关系，构件的应用将会有很大的麻烦。另外一个缺陷是 COM 的接口描述方式缺少扩展性，因为 COM 的接口描述方式实际上是一个内存结构，或者是二进制层面上的。这样对于一些没有确切数据类型表示的编程语言来说是一个困难。

2.2.3　CAR技术[18][19][20]

CAR（Component Assembly Runtime）构件技术是面向构件编程的编程模型，它规定了一组构件间相互调用的标准，使得二进制构件能够自描述，能够在运行时动态连接。

CAR 兼容微软的 COM。但是和微软 COM 相比，CAR 删除了 COM 中过时的约定，禁止用户定义 COM 的非自描述接口；完备了构件及其接口的自描述功能，实现了对 COM 的扩展；对 COM 的用户界面进行了简化包装，易学易用。从该定义中，我们可以说 CAR 是微软 COM 的一个子集，同时又对微软的 COM 进行了扩展，在 CAR 平台 SDK 工具的支持下，使得高深难懂的构件编程技术很容易被 C/C++程序员理解并掌握。

相比于微软的 COM 技术，CAR 技术完全放弃对非自描述数据类型的支持。对于 WEB 时代的软件开发来讲，这是一个很好的选择。WEB 时代需要对数据进行远程传输，如果数据本身不带有对它自己的描绘的话，那在数据的传输和交换过程中就要付出更多的代价。为了支持字符串，数组，结构等非自描叙性数据，CAR 提供了一系列用于封装这些数据的自描叙数据类型。例如 EzStr、EzByteBuf，EzStrBuf、EzArray, EzVar 等。EzStr 一般用来存储用户的常量字符串，它有一个定长的存储区，可以存储用户的字符串，它还保存该字符串的长度。EzByteBuf 提供存储字节的缓冲区，可以存放任何数据，EzStrBuf 中存放的是一个 EzStr 对象，EzArray 用来定义一个多维、定长、自描述数据类型的数组，

EzVar 是一个通用数据类型，它可以存储任何类型的数据。

同时这些数据对象本身是与微软定义的自描叙数据类型是兼容的。这就为 CAR 构件能够在 NT 上面正常的跨地址空间，远程调用提供了基本的前提。而且 CAR 自描述数据类型与传统的数据类型之间转化更加灵活，它提供了一系列对字符串和字节流等进行自描述包装的数据类型和方法。另外，CAR 自描述数据类型不仅可以在堆上分配，而且可以在栈上分配，提高了系统的效率。

在 MS COM 中，构件的一些相关运行信息都存放在系统的全局数据库－注册表中，构件在能够正确运行之前，必须进行注册。而构件的相关运行信息本身就应该是构件自描述的内容之一，所以 CAR 技术选择了把该类信息封装在构件所在的二进制文件中。

MS COM 对构件导出接口的描述方法之一是使用类型库（Type Library）数据（Meta Data，用于描述构件信息的数据），类型库本身是跟构件的 DLL（Dynamic Link Library）文件打包在一起的。但类型库信息却不是由构件自身来来解释，而是靠系统程序 OLEAUT.DLL 来提取和解释，这也不符合构件的自描述思想。而 CAR 则可以通过构件.dll 本身提供的导出函数，非常容易的获得该信息。

在大多数情况下，一个构件会使用到另一些构件的某种功能，也就是说构件之间存在相互的依存关系。MS　COM 中，构件只有关于自身接口（或者说功能）的自描述，而缺少对构件依赖关系的自描述。在网络计算时代的今天，正确的构件依赖关系是构件滚动运行、动态升级的基础。在 CAR 的构件封装中，除了构件本身的类信息封装在构件内外，还对构件的依赖关系进行了封装。即把一个构件对其它构件的依赖关系也作为构件的元数据封装在构件中，我们把这种元数据称为构件的导入信息（ImportInfo）。

对于面向 WEB 服务的应用软件开发，以及开发操作系统这样的大型系统软件而言，采用 CAR 构件技术具有以下的意义：

- 不同软件开发商开发的具有独特功能的构件，可以确保与其他人开发的构件实现互操作。
- 实现在对某一个构件进行升级时不会影响到系统中的其它构件。
- 不同的编程语言实现的构件之间可以实现互操作。
- 提供一个简单、统一的编程模型，使得构件可以在进程内、跨进程甚至于跨网络运行。同时提供系统运行的安全、保护机制。

- CAR 构件与微软的 COM 构件二进制兼容，但是 CAR 的开发工具自动实现构件的封装，简化了构件编程的复杂性，有利于构件化编程技术的推广普及；
- CAR 构件技术是一个实现软件工厂化生产的先进技术，可以大大提升企业的软件开发技术水平，提高软件生产效率和软件产品质量；软件工厂化生产需要有零件的标准，CAR 构件技术为建立软件标准提供了参考，有利于建立企业、行业的软件标准和构件库。

第 3 章　“和欣”浏览器体系架构设计

在“和欣”操作系统上，通过一年多的研究工作，我们已经实现了一个嵌入式浏览器“和欣”浏览器（ElaScope）。本章将阐述我们的嵌入式浏览器 ElaScope 的体系架构设计。3.1 节将阐述 ElaScope 浏览器的体系架构。3.2 节阐述浏览器布局模块和显示模块的通讯机制。在我们的浏览器体系架构中，和其它模块之间的函数调用通信机制不同，布局模块与显示模块之间的通信采用了内部定义的消息机制来通信，这一节将重点介绍这个通信机制。3.3 节阐述 ElaScope 浏览器的基本工作流程。最后 3.4 节简单阐述我们在“和欣”操作系统上 ElaScope 的实现。

3.1　ElaScope 浏览器的体系架构

3.1.1　浏览器体系架构[1][2][7][21]

通过对一些已有的浏览器进行研究，可以发现浏览器主要提供通过解析请求取回数据，并进行解析显示的功能，同时还将提供一定的用户和页面交互的能力。进行归纳总结，可以详细描述为以下几大功能模块：通讯模块(Communication Modular)、语法解析模块(Parser Modular)、布局模块(Layout Modular)、显示模块(Render Modular)、脚本语言支持模块(Script Modular)、插件模块。功能模块的划分，各个浏览器区别不大，浏览器之间区别体现在体系结构上。其中包括各个功能模块的模块化程度，以及模块之间的耦合度，还有对模块的处理的方法的不同，即功能模块是动态加载还是静态绑定。

在桌面浏览器上，浏览器发展的趋势是构件化，动态加载等。其中 Firefox、Grand-Rapid，ICEbrowser 等在这方面做了很好的工作。

为了实现好的浏览器模块化，需要解决的问题主要包括两个：

1. parser modular, Layout Modular, Render Modular 之间的数据交互的问题；
2. 模块之间的同步问题，模块之间的行为需要同步解决。

而为了解决第一个问题，目前浏览器大部分采用 DOM 机制，通过把浏览器

中的中间数据结构用 DOM 来描述，而各个模块操作中间数据结构通过 DOM 的 API 来进行。为了解决第二个问题，通用的做法是设置一个浏览器控制中心，由它来控制和协调模块之间的交互，模块之间尽量减少通讯，模块之间的通讯主要依靠浏览器控制中心来完成，

嵌入式浏览器主要应用在嵌入式设备上，在设计的时候还需要满足嵌入式领域的要求，具体来说就是应该考虑软件的尺寸，运行效率等问题。好的嵌入式浏览器体系结构应该简单，灵活，易于扩展，并且适合于二次开发。一方面尺寸应该达到一定的要求，另一方面效率也是需要考虑的一个重要问题。

3.1.2　ElaScope 体系架构

“和欣”系统是提供网络编程平台的嵌入式操作系统，结合对嵌入式浏览器技术的研究，我们认为“和欣”系统上的 ElaScope 浏览器架构应该具有如下特性：

- 简单。在嵌入式操作系统上，简单的实现有利于控制浏览器的大小，并且有助于人们理解浏览器技术，方便得进行扩展；
- 良好的扩展性。有好的扩展性才可以满足将来更复杂的扩展性；
- 好的模块划分。作为构建平台上的浏览器，可以方便的根据需求进行构件化；

浏览器的研究和实现是一个循序渐进的过程，目前“和欣”平台上已经提供了较为成熟的操作系统以及图形系统，其中构件技术，网络技术都已成型，其自滚动下载等理论体系已经相对完善，因此在满足基本功能的基础上，简单和易扩展的浏览器是一方面可以满足目前的需求，另一方面作为下一步浏览器技术的继续研究以及浏览器技术的扩展都是有好处的。

基于以上的考虑，我们设计了“和欣”系统上的浏览器框架，并且在“和欣”系统上实现了基于该框架的浏览器 ElaScope。参考图 3.1。

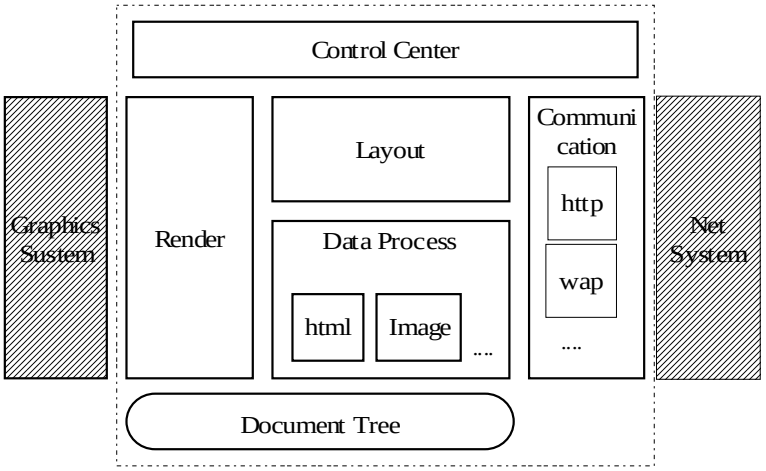

图 3.1 浏览器的框架结构图

浏览器的框架中主要包括文档对象树模块（Document Tree），数据处理模块（Data Process），布局模块（Layout），绘图模块（Render），通讯模块（网络子系统）（Communication），控制中心（Control Center）模块。这些模块包含了浏览器的基本功能模块。在体系框架中，文档对象树被抽象出来，该模块是数据处理、布局和显示模块进行数据交换的基础，通过抽象出该模块，一方面可以很好的进行浏览器内容的管理，另一方面可以很方便扩展为DOM模型。在各个模块中，网络子系统和其他模块之间耦合比较小，比较独立。[21][22][23]

该体系结构是在研究其他浏览器技术及“核心”嵌入式平台的基础上设计出来的，考虑了嵌入式应用的需求，也结合了“和欣”作为网络操作系统的特点。作为嵌入式浏览器，选择了基本的功能模块作为体系结构的基础，而且在实现中，这些基本的功能模块满足了应用的要求。框架对这些功能模块进行了很好的模块化，具有良好的扩展性，通过抽象出文档对象树，使得浏览器可以方便地扩展为 DOM 模型，进一步支持 JavaScript；在模块之间采用消息和事件注册机制来进行通讯，有效降低了模块之间的耦合，并且这种松耦合状态，使得可以很方便的添加对新的数据类型的支持；框架提供了扩展机制，有效地把 CAR 构件融合到浏览器中，使得 CAR 构件可以方便地在浏览器中支持。

3.2 Browser Engine 内部的模块通讯机制

Browser Engine 主要指浏览器的 Layout 模块和显示模块。在我们的浏览器

架构中，Browser Engine 与其它模块的通信采用接口函数和回调函数的方式，而在 Browser Engine 中通讯主要采用了对象机制和消息机制，有效减少了模块的耦合。下面首先介绍内部通讯机制的基础，基于 C 实现的对象框架，然后介绍消息机制，最后介绍通讯机制的实现。

3.2.1　基于 C 语言的对象机制

考虑到代码尺寸和效率的问题，我们选择了 C 语言来实现浏览器引擎，而由于引入了文档对象树，因此需要在 C 语言的基础上提供对象框架，这里我们参考了 GObject 的实现，利用 C 语言实现了自己的对象机制。类型机制提供了基于 C 的面向对象的框架，提供类的注册和管理机制。管理系统会在程序运行的开始启动。每个对象在创建的时候，类型系统会检查是否该类存在，如存在则根据已经注册的类信息进行对象的创建，否则，将先对未注册类的类信息进行注册，并在注册信息里添加并维护类之间的继承关系信息。类型系统也提供了 API 访问某个对象的类型信息，用来动态获取对象的类型信息。

该机制被证明是灵活的，而且提供了很好的对象支持模型，由于利用 C 来实现，有效的控制了代码的行为和尺寸，但是同时也带来了开发和维护的问题，可以考虑以后利用 c++机制进行替代和选择。

3.2.2　消息机制

消息机制提供了消息注册、关联和激活功能。消息机制主要支持布局消息和绘图消息以及与处理超链接相关的消息和对按钮点击进行处理的消息等，通过引入消息机制减少了浏览器对底层图形系统的消息或事件机制的依赖。

1 接口注册。提供 API 将类接口注册到相应的消息上。子类只需要负责注册自己的接口到相应的消息即可，由于父类会注册自己的接口到相应的消息上，同时子类继承了该接口，这样子类也会对相应的消息进行响应。同时子类可以修改继承来的接口，使其指向不同的函数，从而定制对消息的响应的行为。

2 消息关联。关联特定的对象实例和回调函数到指定的消息上。这为特定的实例响应消息提供了可能，不同于上面介绍的消息注册，接口注册提供的是某

一类的接口对消息的注册，意味着，该类的所有实例都将响应注册过得消息，而消息关联只是将某个特定的实例关联到响应的消息上。当消息被激发时，只有该实例将进行消息响应，会调用注册的回调函数。一般通过关联管理的消息是用来管理链接以及管理点击事件的消息。

3 消息的激活。通过发出消息给消息管理系统，后者将负责激活注册的函数。发消息时要求提供对象句柄及消息对应函数所需参数，部分参数在注册的时候也可以指定。被激活的函数是该对象所属类或其父类注册在指定消息上的接口函数以及该对象曾经关联在指定消息上的回调函数。

4 消息的注销。负责把特定对象从消息中移除，避免消息系统维护太多的信息，导致效率下降。

在模块内部定义了许多消息，包括 15 个为通用对象的消息，也就是所有的对象都拥有的消息。同时定义了特殊对象对应的消息。

3.2.3　内部通信过程

内部通讯主要依赖于类型系统和消息系统，在第一次调用构造某类对象的构造函数的时候，会对该类的接口部分进行初始化工作，这是因为该类的接口数据结构部分的指针是被所有实例所共享的。在对接口部分进行实例化的过程中，一方面是调用来注册该接口数据结构中的接口函数指针到相应的消息上，比如布局和显示消息等，另一方面是对相应的接口函数指针进行初始化工作。这样当实例收到某消息的时候，就会通过注册的接口函数指针找到对应的处理函数。这样，当数据分析完毕，浏览器控制中心会发出相应的消息给文档对象树上的实例，并根据前面注册到该消息上的接口函数找到对应的相应函数，进行调用，从而完成消息的响应。还有一类事件注册是注册的面向实例的响应消息函数，当实例接收到某消息是，先判断是否注册过相应的处理函数，如注册过则调用相应的处理函数，否则忽略该消息。

在前面可以看出，我们一方面抽象出了文档对象树，基于一个基于 C 语言的对象框架，而页面所有的内容都是可以抽象为对象的；在对象实例化的时候，进行了消息注册，这里消息包括了布局和绘图消息。在构造文档对象树的同时，还对一些实例进行了消息的关联，这样当浏览器控制中心只要发出相应的消息，

就可以调用不同模块中相应的函数。而这些模块中都是通过访问文档对象树来进行数据通讯的，而对于文档对象树操作的同步性是由控制中心来保证的，控制中心会保证同时只有一个模块在操作文档对象树。

3.3　ElaScope 浏览器基本工作流程

图 3.2 介绍了 ElaScope 工作的流程图。首先由用户进行地址输入，由 Browser Window 截获该消息，并向客户端窗口 Scope window 发出打开链接的消息，Scope window 接到相应的消息后会调用相应的 Browser Engine 提供的 API，进行相应的处理。最后 Browser Engine 会把网页结果内容输出到 Scope Window 中，并会把一些状态信息输出到 Browser Window，并呈现给最终用户。如果页面中有链接或者按钮，当用户点击相应的链接或者按钮时，Scope Window 会截获这些消息，并把相应的消息转发给 Browser Engine 进行相应的处理，例如提交表单操作。

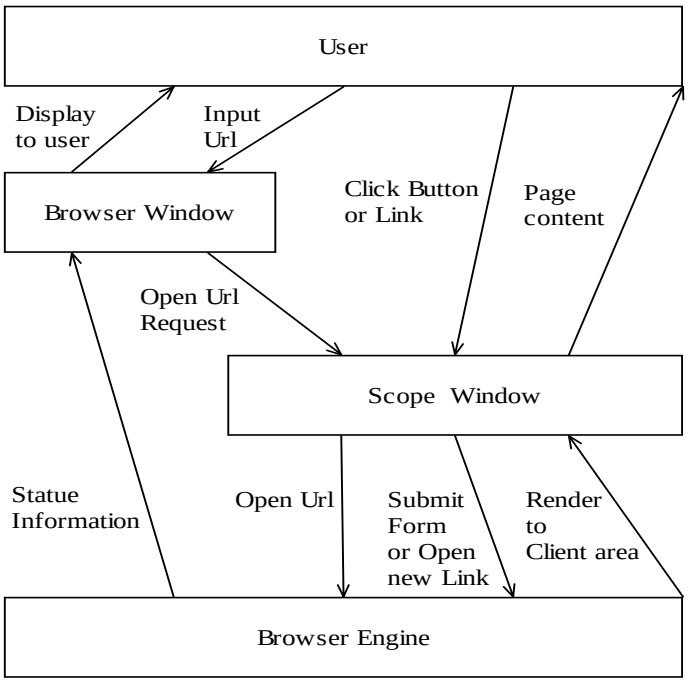

图 3.2　ElaScope 浏览器工作流程.

下面将对各个节点进行简要说明：

- Browser Window：是主窗口，也是浏览器消息的总调度处。窗口包括菜单和输入框，还有相应的前进后退按钮等。一方面负责取得用户输入，

另一方面显示状态信息，比如浏览器访问进度等等。该窗口负责接受用户所有的消息，比如鼠标键盘等消息并进行分发，用户消息也可以发送到这里来进行处理；

- Scope Window：文档显示窗口，窗口大小为 Browser window 客户端区域大小，主要对 Browser window 发来的消息进行处理，并把消息映射为 Browser Engine 可以识别的消息并进行响应，同时负责显示网页的输出内容，并控制响应用户同网页上的控件的交互；
- Browser Engine：浏览器引擎，负责对用户的输入进行解析，并取出数据，对数据进行分析并将最后结果输出到 Scope Window，同时会将一些状态信息输出到 Browser Window 上；

3.4　“和欣”系统上 ElaScope 的实现

本节简单阐述我们的浏览器设计在“和欣”操心系统上的实现：ElaScope 浏览器。

“和欣”系统上的 ElaScope 浏览器支持可以工作在手机、数字电视等嵌入式平台目前支持 HTML4.01, HTTP1.0/HTTP1.1, Jpeg, Gif, XHTML 等，对 CSS 部分支持。在 ElaScope 中，利用体系结构很好的新协议支持的特性，很好地添加了对图像的支持，目前可以通过配置文件实现浏览器是否支持图形的显示。而且各个模块之间耦合性比较小，可以比较方便的对各个模块进行独立的修改。由于基于 C 实现，使得目前 DEBUG 版本仅为 300k。由于利用了回调机制，使得其解析和显示速度在 Win2k 虚拟机上达到了甚至超过了通用浏览器。同时 ElaScope 具有很好的扩展机制，可以方便的添加对于 CAR 构件的支持。

第4章　浏览器网络子系统的设计与实现

4.1　浏览器网络子系统的设计

本节提出了 ElaScope 浏览器网络子系统的设计。该设计基于面向对象技术，参考了现有的嵌入式浏览器网络子系统设计及相关 Internet 技术，面向嵌入式领域，提供稳定、实用、功能完善的浏览器网络服务。4.1.1 小节阐述设计目的，4.1.2 小节阐述体系结构，说明网络子系统与浏览器的其它模块以及与底层“和欣”操作系统的交互关系，4.1.3 小节阐述网络子系统的对外接口，接下来的几个小节阐述网络子系统设计中的几个关键问题，包括回调机制、模块划分、控制流程、事件驱动机制以及多线程机制。

4.1.1　设计目的

浏览器网络子系统的最基本的功能是完成 URL 请求下载功能，根据用户请求的 URL，通过 HTTP 协议向服务器获取该 URL 对应的资源。网络子系统的这个基本功能可以分成以下几个方面：

a. 分析用户提出的请求
b. 生成请求数据要求的 HTTP Header
c. 通过 HTTP 协议请求数据
d. 解析返回任务的当前状态
e. 数据下载完成后设置相关标记并通知相应的浏览器的其它模块进行处理

以上是一个浏览器的网络子系统的最基本的功能。对于一个功能强大的浏览器而言，网络子系统需要有更多的功能，包括缓存管理、文件访问支持、其它协议（FTP 等协议）支持、并行访问、资源下载、安全支持等功能。

针对“和欣”浏览器的要求，“和欣”浏览器网络子系统被设计为高效、完善的浏览器网络子系统。网络子系统支持以下功能：

- ✓ 支持 HTTP/1.0、HTTP/1.1 协议；

- ✓ 支持本地文件读取；
- ✓ 支持缓存；
- ✓ 支持 Cookies；
- ✓ 支持并行网络数据读取；
- ✓ 支持 URL 资源下载；

在嵌入式浏览器的网络子系统的设计与实现中，要考虑到嵌入式系统的特殊性，使得网络子系统具有可伸缩性与容错性。可伸缩性要求整个系统有比较完整的功能，但由于嵌入式应用的差异性，使得系统能工作在高端嵌入式系统上，也能工作在低端嵌入式系统。同时要考虑到资源有限性，嵌入式应用中的资源一般都比较有限。系统可能没有大容量的存储设备。在网络子系统的构件化设计中应充分考虑这些资源限制，使得系统能够与各种应用相适应。容错性要求能处理遇到的各种异常环境，使系统工作稳定。[1][23]

4.1.2　体系结构

在 ElaScope 浏览器中，网络子系统需要与浏览器的其它模块交互协作以及底层“和欣”操作系统交互，共同完成浏览器的功能。图 4.1 描述了 ElaScope 浏览器的体系结构。

浏览器网络子系统需要与浏览器的用户界面处理模块、页面处理模块交互协作。用户界面处理模块在检测到URL请求后(可以是用户在浏览器地址栏中输入的URL请求，也可以是用户通过点击网页上的超链接提出的URL请求，这两种URL请求，在ElaScope浏览器被定义为根URL请求，即ROOT URL REQUEST，例如浏览器用户通过在地址栏中输入http://www.tsinghua.edu.cn，向网络子系统提交的URL请求就是一个根URL请求)，将向网络子系统提出URL请求，通过网络子系统获取资源；网络子系统成功获取资源以后，将设置数据解析标志,通知浏览器的页面处理模块对获得的数据进行解析。在数据解析过程中，如果需要请求其它的资源（如图片资源等），将继续向网络子系统提交URL请求(这类在数据解析过程中提出的URL请求，在ElaScope中被定义为非根URL请求，即NON ROOT URL REQUEST，例如在解析http://www.tsinghua.edu.cn页面时提交的图片资源请求，就是一个非根URL请求)；网络子系统在向服务器提交HTTP请求时，

当需要完成HTTP授权时，也需要通过界面处理模块向用户提出授权要求；最后网络子系统也将通过用户界面模块通知浏览器用户当前网络的处理状态。在网络子系统与浏览器用户界面模块的通信中，采用了消息机制。

网络子系统与操作系统进行交互中，需要调用操作系统提供的服务。ElaScope 浏览器的网络子系统需要调用“和欣”操作系统的网络接口，完成网络通信功能；调用操作系统的文件系统提供的接口，完成文件的访问；参考图 4.1。

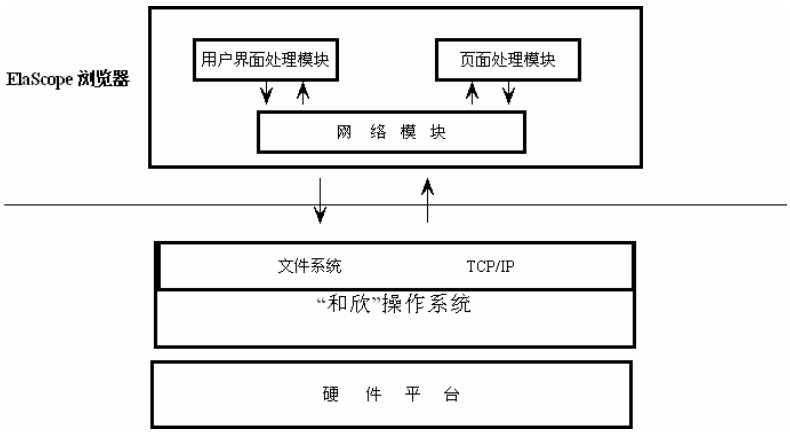

图 4.1 ElaScope 浏览器网络子系统体系结构

4.1.3　网络子系统的对外接口

ElaScope 网络子系统对浏览器的其它模块提供的服务，以接口函数的形式提供。主要提供二种类型的接口函数，一是网络子系统的初始化函数与结束函数，二是 URL 请求接口。

4.1.3.1　网络子系统的初始化函数与结束函数

ElaScope 浏览器在初始化过程中，将调用网络子系统的初始化函数，完成网络子系统的初始化过程，包括模块内部数据结构的初始化，网络子系统使用的“和欣”操作系统的网络服务的初始化工作；在浏览器关闭的过程中，将调用网络子系统的结束处理函数，完成模块的结束处理。函数说明如下：

```
void a_Cache_init(void);
```

说明：完成网络子系统的初始化工作。

```
void a_Cache_freeall(void);
```

说明：完成网络子系统的结束工作。

4.1.3.2　URL 请求接口函数

URL 请求接口完成网络子系统最重要的功能，接受用户界面处理模块与页面处理模块提出的 URL 请求。URL 请求包括获取一个 URL 资源的请求，也包括停止当前 URL 请求操作的请求。图 4.2 描述了网络子系统与其它模块的接口情况。

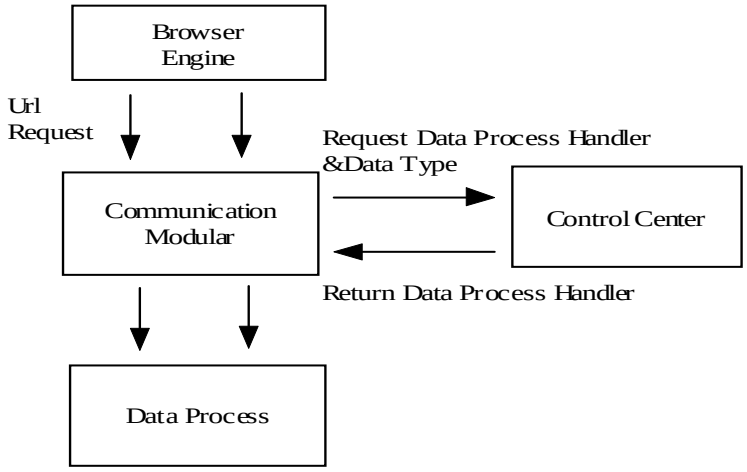

图 4.2 网络同其他模块接口情况

URL 请求被定义为三种类型，如表格 4.1。

URL 类型定义	类型说明
#define WEB_RootUrl 1	浏览器用户通过在浏览器地址栏输入或者页面中点击超链接产生的 URL 请求，被定义为根 URL 请求
#define WEB_Image 2	HTML 页面中的图片等 URL 资源请求
#define WEB_Download 4	下载请求，URL 资源将保存到文件中

表格 4.1 URL 请求类型定义

在向网络子系统提出URL请求时，需要将相关参数传递给网络子系统。在

ElaScope中，定义了ElaScopeWeb数据结构，包含一个URL请求的重要信息。ElaScopeWeb中包含了要请求的URL字符串，如http://www.tsinghua.edu.cn，URL请求类型标志（WEB_RootUrl/WEB_Image/WEB_Download）以及其它相关信息。

```
typedef struct _ElaScopeWeb ElaScopeWeb;
struct _ElaScopeWeb {
  char *url;            /* Requested URL */
  char *data;           /* Posted URL data */
  BrowserWindow *bw;    /* The requesting browser window [reference] */
  int flags;            /* Additional info, URL type */
  ElaScopeImage *Image;    /* For image urls [reference] */
  FILE *stream;            /* File stream for local saving */
  int SavedBytes;          /* Also for local saving */
};
```

完成 URL 请求的函数定义如下：

```
int    a_Cache_open_url(void *Web, CA_Callback_t Call, void *CbData);
```

说明：向网络子系统提出一个 URL 请求。输入参数 Web 为 ElaScopeWeb 指针，输入参数 Call 为相应的数据处理函数的句柄，网络子系统在数据请求处理完成以后，通过该函数句柄回调相应的处理函数，完成数据的处理过程。关于网络子系统的回调机制将在下一节中介绍。CbData 是相关绑定数据指针，函数返回唯一标志这次 URL 请求的 Key. 在 ElaScopeWeb 结构中包含的请求的 URL 字符串，URL 类型标志等信息。

```
void a_Cache_stop_client(int Key);
```

说明：停止由 Key 标志的 URL 请求的执行。

```
void a_Cache_stop_all_clients();
```

说明：停止当前网络子系统的所有的 URL 请求。

4.1.4　网络子系统的回调机制

ElaScope 的网络子系统应该是根据请求进行多线程服务的，从而是异步响应请求。网络子系统需要保存发出请求模块的信息，以便取得数据后可以返回给相应的数据处理函数。在一些情况下在发出请求的时候，数据类型是不好判断的，或者是未知的。图 4.3 给出了网络接口部分的数据流图。

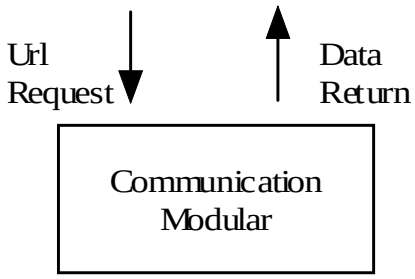

图 4.3　网络接口部分数据流图

针对上述问题，我们提出了利用回调机制进行通讯的解决方案。当网络请求处理完成以后，通过回调的形式，将请求结果返回给浏览器引擎模块。各个模块均可以通过调用接口发出网络请求，而浏览器可以处理的数据，需要在开始进行注册，以数据类型和相应的一个处理函数对的形式注册。如果发出请求的时候，了解具体的数据类型及相应的接受数据模块的入口接口，可以作为参数直接传输给网络子系统，网络子系统负责保存这些数据，当取得数据后，直接调用相应的接受数据模块的入口接口。已知数据类型的处理函数在浏览器的控制模块中注册，需要对需要处理的数据类型和相应的处理函数进行注册，注册通过调用 Mime_add_minor_type 来实现其中接口函数的声明：

```
typedef DwWidget* (*Viewer_t) (const char*, void*, CA_Callback_t*, void**);
```

Control Center 用来存储数据类型的数据结构

```
typedef struct {
    const char *Name;     /* MIME type name */
    Viewer_t Data;        /* Pointer to a function */
} MimeItem_t;
```

其中接口函数的声明，

```
typedef DwWidget* (*Viewer_t) (const char*, void*, CA_Callback_t*, void**);
```

浏览器的控制模块设置了存储数据类型的数据结构。**错误！**

而对于未知数据类型的数据，在取得相应的数据后，网络子系统可以根据协议头判断出数据类型，然后根据数据类型向控制中心请求接受数据模块的入口接口，如果是已注册的数据类型，则调用相应得函数取得相应得数据类型接口并返回给网络子系统，从而可以进行处理，如果是未知类型，则返回特定的接口处理函数，该函数将直接返回。

网络子系统通过检测从服务器返回的 HTTP Header 来判断数据类型。在 HTTP 中，MIME 类型被定义在 Content-Type header 中。最早的 HTTP 协议中，并没有附加的数据类型信息，所有传送的数据都被客户程序解释为超文本标记语言 HTML 文档，而为了支持多媒体数据类型，HTTP 协议中就使用了附加在文档之前的 MIME 数据类型信息来标识数据类型。这种机制通过抽象出两类接口有效的降低了通讯模块和浏览器其他模块的耦合性。并且增加了灵活性，对新的数据处理函数，只要提供相应的数据类型入口接口即可，只需在控制中心注册数据类型和相关的处理函数的句柄即可。当取得相应类型的数据时，网络子系统会调用相应的函数，根据数据类型查用到相应的处理函数句柄，从而利用已经注册好的函数进行数据处理。

4.1.5　模块划分与控制流程

网络子系统的内部模块划分为多个子模块，如下图。网络访问管理模块是网络对浏览器其它模块提供服务的接口模块，浏览器用户界面处理模块与页面分析模块的 URL 请求，将通过网络访问管理模块提交到浏览器网络系统。网络访问管理模块将首先检查缓存模块,判断是否请求的 URL 资源已经存在缓存中，如果已经存在，那么缓存的页面将直接返回给用户界面处理模块或者页面分析模块。如果没有存在缓存中，网络管理模块将通过创建一条并发提交控制链建立起访问处理流程。并发提交控制链将完成访问过程的事件传递，将访问事件依次传递到相应的处理模块进行处理。关于并发提交控制链将在下一节事件驱

动机制介绍。图 4.4 介绍了网络子系统的组成模块。

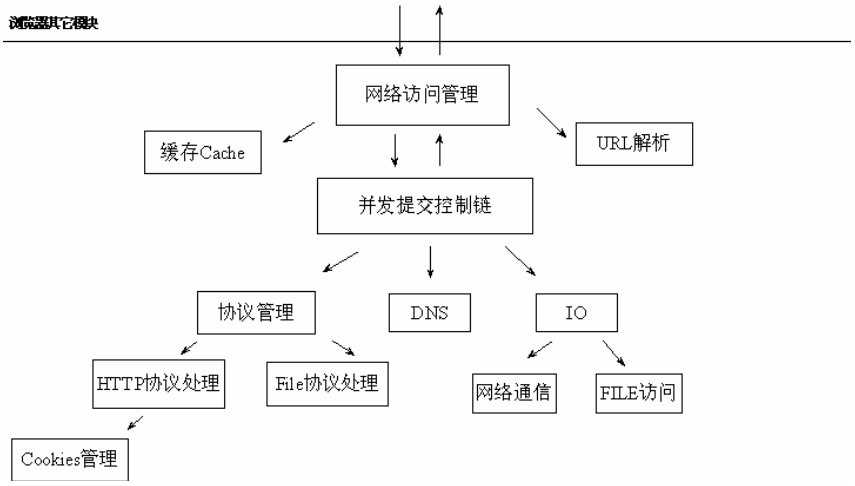

图 4.4 网络子系统的模块划分

（1）网络访问管理模块。网络子系统的对外接口都是通过该模块提供。URL 请求将通过模块接口提交给网络子系统。对于 URL 请求，网络访问管理模块首先解析 URL 字符串，判断 URL 请求是否合法。对于合法的 URL 请求，将生成内部的表示形式。如果请求的 URL 是 HTTP 协议请求，模块将首先访问缓存模块，检测请求的数据是否在缓存当中，如果在缓存当中，将直接将缓存的数据通过注册的回调函数返回给浏览器的页面处理模块。如果数据没有存在缓存当中，网络子系统将建立一条并发提交控制链，并启动处理过程。处理事件将通过并发提交控制链传递给相应的处理模块，完成整个的处理过程，并最终返回网络访问管理模块，通过回调函数将处理结果返回给浏览器的处理模块。

（2）缓存模块。Cache 通称高速缓存，它用于提高速度较快的设备与速度较慢的设备之间的传输速率。在计算机系统中，为了提高整体效率，很多地方采用了 Cache 技术。由于网络速度相对本地磁盘读取速度要慢很多，因而产生了 Web 缓存。Web 缓存通过降低服务器负载、减小网络流量、降低数据延时提高服务质量等，减小了网络建设的费用，提高了网络质量。Web 缓存主要有三类：浏览器 Cache、Proxy Cache、Surrogate。

网络子系统的缓存属于浏览器 Cache，通过在磁盘或者内存中开辟一定的空间，保存最近上网的数据，下次访问相同数据时就可以直接从本地获取，而不必每次都从网上获取。浏览器 Cache 中保存的数据是静态数据，每个数据都有一个生存期（TTL，time to live），浏览器会根据生存期来判断是否需要更新相应

的数据。

在 ElaScope 的缓存模块中，同时支持内存缓存与文件缓存，通过配置文件确定采用的缓存类型，缓存的大小等属性。

（3）并发提交控制链(CCC)。网络子系统的处理是通过事件驱动的，事件与数据在网络子系统内部子模块通过 CCC 传递。

（4）URL解析模块。模块解析URL请求字符串。浏览器的其它模块提出的URL请求将以字符串的格式传递给网络访问管理模块，如http://www.tsinghua.edu.cn，URL解析模块负责判断URL请求的合法性，对于合法的URL请求，将解析为内部的表示格式，传递给其它模块。

（5）协议管理模块。根据传递的 URL 内部表示，判断 URL 请求协议类型，如是 HTTP 协议还是其它协议。根据不同的协议调用不同的协议处理模块。

（6）HTTP 协议处理模块。HTTP 协议处理模块完成两个主要的功能，一是通过 HTTP 协议完成数据的发送与获取，另一个是 HTTP 模块事件处理，完成 HTTP 模块与其它模块的协同工作。

（7）FILE 协议管理。完成本地文件的管理功能。

（8）Cookies 管理模块。HTTP 协议是无状态协议，在对页面进行个性化处理或是为产品及服务注册时，无状态的 HTTP 连接收到了限制。为了在 HTTP 的客户端请求和服务器响应之间建立有状态的会话，在 HTTP 协议基础上引入了 Cookies 的概念。Cookies 管理模块完成 Cookies 的管理功能。

（9）DNS 模块。DNS 模块负责完成异步 DNS 协议解析，DNS 事件处理，DNS 缓存管理。

（10）IO 模块、网络通信与文件访问。提供本地文件的访问以及完成网络通信功能。

4.1.6　事件驱动机制

ElaScope 的网络子系统是一个多任务并行处理的模块。很多嵌入式浏览器的网络子系统对 URL 请求只能串行处理，不支持同时的 URL 请求处理。ElaScope 的网络可以同时启动多个 URL 请求，并行的访问服务器，提高处理的效率。

ElaScope 网络子系统由事件驱动，如图 4.5：

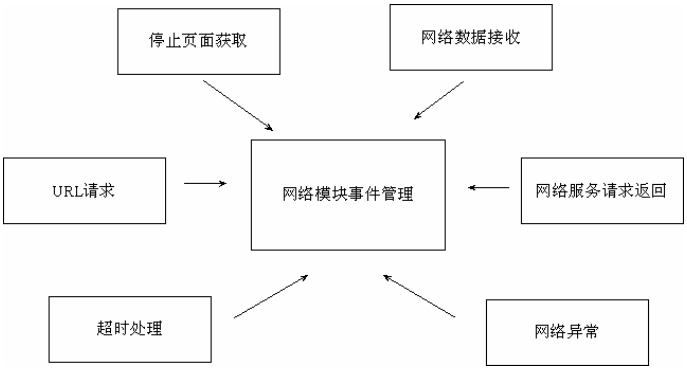

图 4.5 网络子系统事件

网络子系统的事件管理模块接收发生的事件，并激活网络子系统内部的控制模块启动事情的处理。提交给网络子系统的事情一种是来自浏览器的其它模块的 URL 请求事件，由其它模块调用网络子系统的 URL 请求接口产生，包括停止页面获取事件和 URL 请求事件。当浏览器用户执行网页访问停止操作，或在一个页面访问完成之前，输入新的 URL，请求访问新的页面，都会产生停止页面获取操作。

另一种是来自网络通信产生的事件。当服务器对 URL 请求的响应时间超过设置的时间时，会产生超时事件，激发网络事件管理模块进行超时处理。网络数据到达时，产生网络数据接收事件，由网络相应处理，将得到的数据送到事件循环中进行处理。网络访问中的异常事件，如 DNS 解析失败，服务器连接失败等网络异常情况的发生会产生网络异常事件，交给网络事件处理模块进行处理。

在 ElaScope 的网络实现中，网络子系统事件管理部分并不存在单独的管理模块，而是由多个模块共同实现的。每个子模块，如 HTTP 模块，IO 通信模块，DNS 模块，缓存模块都存在对应的事件处理逻辑，网络子系统接收到的事件，将通过网络子系统的并发提交控制链完成事件的传递过程，将通过控制链，将事件传递到对应的处理模块完成处理。例如当网络数据读取失败时，网络异常事件将传递给网络事件模块，并传递给 IO 模块的异常处理逻辑，IO 模块的异常处理完成后，将这个事件通过 CCC 链向上传递给其它模块进行处理，最后重新返回网络事件模块，将处理结果通过回调函数的方式提交给浏览器的其它模块。

4.1.7　多线程机制

网络子系统的内部为了实现并行通信，采用了多线程机制，浏览器引擎发出的多个URL请求可以被同时处理。网络子系统提供了三种线程模型，一是单线程模型，网络子系统只有一个线程，所有的URL请求将根据请求顺序被依次处理。二是固定数目的线程。网络子系统设置线程数目，一般为四个工作线程，可以同时并行处理四个URL请求。第三种方式为开线程方式。在这种方式下，并不设定线程数据，对每一个URL请求启动一个线程。针对系统的不同，可以采用不同的线程模型。[24]

网络子系统的多线程机制，借助了“和欣”操作系统提供的线程函数与相应的同步互斥机制来完成。

4.2　浏览器网络子系统的实现

本节介绍 ElaScope 浏览器网络子系统的实现。网络子系统由多个子模块组成，这里将只介绍关键模块的实现。

4.2.1　URL 解析模块

URL是浏览器中的重要概念。URL(Uniform Resource Locator)用来标记网络上的资源，给出每个资源的唯一网络位置标志。URL不仅用于HTTP协议，同样应用于FTP、News和Gopher等协议。它可以分为绝对路径和相对路径两种。在HTTP传输中，使用的接口URL是标准的URL路径。在URL中要包括：协议标志（protocol name）、主机名（host name）、路径（path name）、端口号（port name）。[25]

URL解析模块将首先检测输入的URL字符串是否合法，URL的检测算法定义在RFC文件中。例如http://www.tsinghua.edu.cn与www.tsinghua.edu.cn都是合法的URL输入。

对于合法的 URL 字符串，解析模块将生成 URL 的内部格式，记录 URL 的各个子成员部分，在内部调用中使用。URL 结构定义如下图所示。结构中的各

个部分对应着 RFC 中 URL 中定义的各个部分，具体参考 RFC 文件。

```
struct _ElaScopeUrl {
    char *url_string;
    const char *buffer;
    const char *scheme;
    const gchar *authority;
    const gchar *path;
    const gchar *query;
    const gchar *fragment;
    const gchar *hostname;
    int port;
    int flags;
    const gchar *data;                  /* POST */
    const gchar *alt;                   /* "alt" text (used by image maps) */
    int ismap_url_len;                  /* Used by server side image maps */
    int32 scrolling_position_x, scrolling_position_y;
                                        /* remember position of visited urls */
    int illegal_chars;                  /* number of illegal chars */
};
```

4.2.2　并发提交控制链（Concomitant Control Chain）

网络子系统的 CCC（Concomitant Control Chain）链接网络的各个模块，完成控制流程的传递是实现网络部分模块相互协作的关键，由 CCC 链串接 Cache，DNS, HTTP, IO 部分，完成整个处理流程。

CCC 链是一个双向的链表，定义了两个方向，前向（FWD），后向（BCK）。前向代表数据的发送工作，后向操作完成数据的接收工作。同时 CCC 中定义了五种操作：OpStart，OpSend，OpStop，OpEnd，OpAbort。分别代表启动操作，发送操作，停止操作，结束操作以及操作中发生了错误。

数据结构：

```
typedef struct    _ChainLink ChainLink;
typedef struct    _DataBuf DataBuf;
typedef void    (*ChainFunction_t)(int Op, int Branch, int Dir, ChainLink
*Info, void *Data1, void *Data2);
/* This is the main data structure for CCC nodes */
struct _ChainLink {
      void *LocalKey;

      ChainLink *FcbInfo;
      ChainFunction_t Fcb;
      int FcbBranch;

      ChainLink *BcbInfo;
      ChainFunction_t Bcb;
      int BcbBranch;
}
```

;

在 CCC 结点的定义中，将记录指向前向节点与后向节点的指针，同时将记录处理前向节点与后向节点的处理函数指针。通过这些数据，完成对当前节点的前向处理操作与后向处理操作。通过链表中的函数指针，完成操作的传递处理。因此在链表节点定义包含了事件定义、数据以及对应模块的事件处理函数（在网络子系统中，定义为 CCC 处理函数）。

完成一个完整的 HTTP 通信的 CCC 链表如下，通过该链表，将 IO 各个模块的处理模块连接起来，事件（包括相应的数据）在 CCC 链中传递，各个模块进行相应的事件处理，完成 HTTP 通信。图 4.6 给出了各个模块在 CCC 链中的位置关系，给出事件的流动方向。

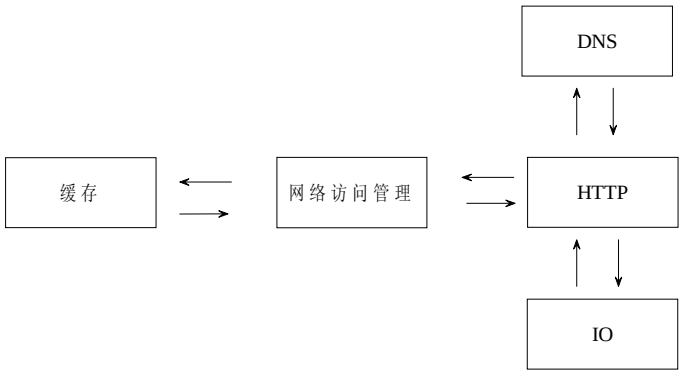

图 4.6 完成基本功能的 CCC 链

4.2.3 HTTP模块[26]

本节介绍 HTTP 模块的实现，HTTP 模块完成以下的功能：解析传递参数生成 HTTP 待发送数据；建立与服务器的连接；建立与 IO 模块的连接，启动 IO 模块完成数据的发送与接收；处理收到的服务器发送的数据；完成 HTTP 代理的管理；

■ HTTP 代理的管理

通过 HTTP 协议得到 WWW 资源有两种方式：直接和 URL 所在的主机进行连接，从 WWW 服务器上得到文件与通过代理服务器（Proxy）来获得 WWW 服务。在 HTTP 模块中定义了管理代理的相关数据结构，通过浏览器的配置文件完成 HTTP 代理的设置。在 HTTP 协议与服务器连接的过程中，如果设置了代理服务器，那么 HTTP 模块将与代理服务器建立连接，如果没有设置代理服务器，将直接与 URL 主机进行连接。

变量定义：

```
static ElaScopeUrl *HTTP_Proxy = NULL;
static char *HTTP_Proxy_Auth_base64 = NULL;
```

静态成员变量 HTTP_Proxy 来记录代理信息，HTTP_Proxy_Auth_base64 的授权管理信息。

在 HTTP 模块初始化的过程中，会通过检测环境变量与配置文件初始化代理信息。完成初始化的代码片断：

```
gchar *env_proxy = getenv("http_proxy");
if (env_proxy && strlen(env_proxy))
        HTTP_Proxy = a_Url_new(env_proxy, NULL, 0, 0, 0);
if (!HTTP_Proxy && prefs.http_proxy)
        HTTP_Proxy = a_Url_dup(prefs.http_proxy);
```

如果在配置文件中预先定义了使用代理服务器的授权信息(以 user:password 字符串格式定义，那么在初始化的过程中，还将初始化授权信息 HTTP_Proxy_Auth_base64。

```
gchar *http_proxyauth = g_strconcat(prefs.http_proxyuser, ":", str, NULL);
HTTP_Proxy_Auth_base64 = a_Misc_encode_base64(http_proxyauth);
g_free(http_proxyauth);
```

其中 a_Misc_encode_base64 函数完成字符串的 base64 格式的编码工作。

■ 解析传递参数生成 HTTP 待发送数据

函数原型：

```
char *a_Http_make_query_str(const ElaScopeUrl *url, gboolean use_proxy)
```

函数按照请求信息的格式形成待发数据，输入参数 URL 为请求的 URL 字符串经 URL 模块解析后的内部表示结构，参数 use_proxy 确实是否使用代理服务器。在函数的执行过程中，还将通过检测 HTTP 模块的设置变量，判断 Cache 的使用、HTTP 协议版本的使用。

函数的返回结果是 HTTP 请求消息字符串。HTTP 请求消息的格式如下：

```
Request = Request-Line
          *((general-header
            | request-header
            | entity-header) CRLF)
            CRLF
            [message-body]
```

下面给出了一个典型的使用 GET 方法的 HTTP 请求信息。对于 http://www.tsinghua.edu.cn的请求，生成的HTTP请求信息如下：

```
GET/HTTP/1.1
Host:www.tsinghua.edu.cn
Pragma:no-cache
Cache-control:no-cache
User-Agent:ElaScope1.0
```

请求信息向清华大学的 HTTP 服务器请求主页的一个页面（HTTP 中的 POST 方法与 GET 方法类型，POST 方法中需要提交的数据放在 message-body 中）。

■　建立与服务器的连接

HTTP 模块的以异步的方式完成与服务器的连接。在 HTTP 模块维护了一个 HTTP 处理链表，HTTP 模块将服务器的连接请求、信息的发送与接收请求提交给 IO 模块以后直接返回。在 IO 模块完成与服务器的连接、信息的发送与接收以后，会产生相应的事件，通过 Concomitant Control Chain 将事件传递给 HTTP 模块，HTTP 模块通过链表获得要处理的 HTTP 请求数据，完成事件处理。

链表节点数据结构：

```
/*
 * Http Socket Data Structure definition.
 */
typedef struct {
    int SockFD;                 /* socket id */
    const ElaScopeUrl *Url;         /* reference to original URL */
    unsigned int port;          /* need a separate port in order to support PROXY */
    gboolean use_proxy;         /* indicates whether to use proxy or not */
    void *web;                  /* reference to client's web structure. Not used now */
    GSList *addr_list;          /* Holds the DNS answer */
    GSList *addr_list_iter;     /* Points to address currently being used */
    int Err;                    /* Holds the errno of the connect() call */
    ChainLink *Info;            /* Used for CCC asynchronous operations */
}SocketData_t;
```

链表节点中记录了完成 HTTP 处理的相关数据。

HTTP 请求链表的定义（Klist 是每个链表节点有唯一主键的简单链表）：

```
/* Active sockets list. It holds pointers to SocketData_t structures. */
static Klist_t    *ValidSocks = NULL;
```

HTTP 通过解析 URL 获得目标服务器地址，通过 DNS 模块获得目标服务器的 IP 地址，然后创建 HTTP 链表节点插入链表，最后产生连接服务器的事件，通过 CCC 链传递给 IO 模块。IO 模块会异步完成连接工作，在连接完成后通知 HTTP 模块处理。

■　向服务器发送 HTTP 请求

IO 模块在完成与服务器的连接以后，会通过 CCC 将事件传递给 HTTP 模块，HTTP 模块进行相应的事件处理。处理流程参考图 4.7。

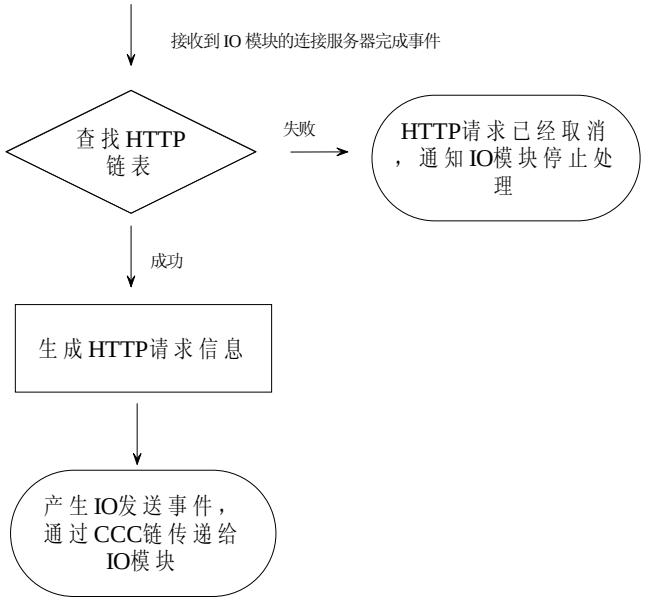

图 4.7 HTTP 请求发送的处理流程

■　接收服务器数据

IO 模块在获得数据后将产生数据到达事件并通过 CCC 将事件与相关数据传递给 HTTP 模块进行处理。HTTP 模块将解析服务器返回的 HTTP 响应消息。如果成功的获得了 HTTP 数据，将把数据传递给网络访问管理模块，由该模块将数据返回给浏览器的其它模块。

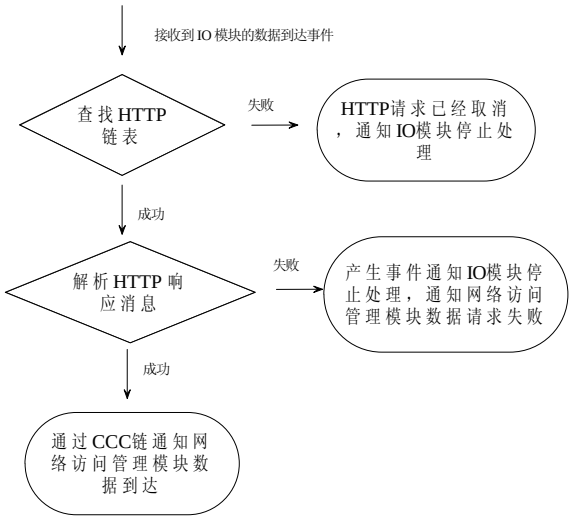

图 4.8 接收服务器数据处理流程

4.2.4　网络访问管理模块

网络访问管理模块通过多线程编程提供给浏览器引擎异步并行的 URL 请求处理服务。网络访问管理模块中通过创建工作线程来完成并行访问，在访问完成后，由回调的方式将数据传递给浏览器的其它模块。

a_Cache_open_url 是网络访问管理模块的对外接口函数。通过改接口的调用，URL 请求将进行管理模块的 URL 队列。URL 队列节点中记录需要处理的 URL，同时记录了处理完成的回调函数句柄。

URL 请求队列节点定义：

```
/*
 * Data structure for cache clients.
 */
struct _CacheClient {
    int Key;                        /* Primary Key for this client */
    const ElaScopeUrl *Url;         /* Pointer to a cache entry Url */
    unsigned char *Buf;             /* Pointer to cache-data */
    unsigned int BufSize;           /* Valid size of cache-data */
    CA_Callback_t Callback;         /* Client function */
    void *CbData;                   /* Client function data */
    void *Web;                      /* Pointer to the Web structure of our client */
};
```

管理模块在执行初始化 a_Cache_init 函数时会创建数目固定的工作线程。a_Cache_open_url 函数在将 URL 请求放入请求队列后，如果队列为空，将会 Resume 工作线程进行处理。工作线程将建立一条 CCC 链，链接各个模块，完成处理。在工作完成后，在 URL 队列中找到 URL 请求对应的节点，完成数据的回调处理。工作线程执行完成如何，如果 URL 队列有其它的 URL 请求，将继续处理请求，如果请求完成，将进入 Suspend 状态，等待 Resume。管理模块使用了“和欣”操作系统提供的线程的各种访问和操作接口方法。

4.2.5　缓存模块[27][28][29][30][33][31][32]

缓存在是浏览器中的重要技术。一般浏览器都在磁盘或者内存中开辟一定的空间，保存最近上网的数据，下次访问相同数据时就可以直接从本地获取，而不必每次都从网上获取。一些浏览器除了使用磁盘来缓存数据外，还使用了内存，以获取更快的速度。ElaScope 的缓存支持两种类型，文件缓存与内存缓存。为了提高 Cache 的查找速度，使用了哈希表来记录缓存的数据的。Cache 中哈希表的实现使用了 glib 库中的哈希表。

■　哈希函数

哈希函数通过输入的 URL 字符串，如 www.tsinghua.edu.cn 来确定哈希键值。在函数的实现中，将首先通过宏定义对输入的 ElaScopeUrl 的指针进行处理，获得 URL 字符串，然后根据 URL 字符串产生 URL 键值。

```
static guint Cache_url_hash(gconstpointer key)
{
    const char *p = URL_STR((ElaScopeUrl *)key);
    guint h = *p;

    if (h)
        for (p += 1; *p != '\0' && *p != '#'; p++)
            h = (h << 5) - h + *p;

    return h;
}
```

■　初始化函数

在 Cache 模块的初始化过程中，将会使用 glib 的函数完成哈希表的初始化。

```
void a_Cache_init(void)
{
    CacheHash = g_hash_table_new(Cache_url_hash, Cache_hash_entry_equal);
}
```

■　结束函数

如果 Cache 的缓存类型设置为内存缓存，在浏览器关闭时，Cache 模块不需要做处理，所有保留在内存中的缓存数据将丢失。对于文件类型的缓存，将完成缓存索引文件的处理，使得保留在文件中页面缓存可以在 Cache 的下次启动中恢复过来。

4.2.6　IO 模块

IO 模块是浏览器网络系统中与底层操作系统交互的模块的，IO 模块完成 IO 事件的处理，在操作系统的“和欣”操作系统的网络服务与文件服务接口的基础上，通过管理模块的多线程机制，提供异步的 IO 服务。IO 模块在完成通信或者通信失败的情况下，将通过 CCC 将事件传递给其它相应模块进行相应的处理。

第 5 章　浏览器网络子系统构件化研究

5.1　“和欣”浏览器的构件化

1980 年代以来，目标指向型软件编程技术有了很大的发展，为大规模的软件协同开发以及软件标准化、软件共享、软件运行安全机制等提供了理论基础。其发展经历了几个阶段。

以 Java，.NET，CAR 为代表的编程技术，强调构件的自描述和构件运行环境的透明性，是网络时代编程的重要技术。CAR 构件编程模型建立在面向对象技术的基础之上，是完全面向对象的，提供了动态构造部件模块（运行中可以构造部件）的机制。构件在运行时动态装入，是可换的。“和欣”构件运行平台可以依据 CAR 构件的元数据信息自动生成构件的运行环境，生成代理构件即中间件，通过系统自动生成的中间件对构件的运行状态进行干预或控制，或自动提供针对不同网络协议、输入输出设备的服务（即运行环境）。

在“和欣”浏览器的最初实现版本 ElaScope 中，由于对浏览器技术缺乏深入的研究，我们参照现有的嵌入式浏览器的设计，采用了面向对象的分析与设计方法，完成 ElaScope 浏览器的设计与实现。面向对象设计运行系统容纳具备可塑性和功能强大的对象原型和对象实例，它们可以被系统的其它部分简单的重用。但面向对象编程强调的是对象的封装，模块（对象）之间的关系在编译的时候被固定，模块之间的关系是静态的，在程序运行时不可改变模块之间的关系，就是说在运行时不能换用零件。

通过对 ElaScope 的构件化设计与实现，将实现基于 CAR 构件技术的“和欣”浏览器，将充分利用构件化设计的优点：

- 易于扩展/剪裁。嵌入式浏览器需要适应各种嵌入式设备，需要浏览器的功能易于扩展/剪裁。构件化设计的模型定义了二进制级的重用标准，其功能完全封装在内部实现中，易于组织为二进制级的链接形式，独立的添加和删除；模块使用组件库来集中管理组件，易于添加新的功能、删除已有的功能。并且基于 CAR 构件技术的应用程序，可以在系统运

行过程中，动态加载或者卸载功能模块。

- 可演进性。由于 CAR 构件模型中包含了构件服务的标准，容易实现新的构件功能而不需要修改客户程序，系统容易简单而健壮的升级，从而不断增加新功能。
- 平台无关性。构件模型一旦建立起来，随之需要提供一系列的构件来支持应用程序的功能调用，模型的二进制标准与目标平台无关。“和欣”构件运行平台不仅在“和欣”操作系统上实现，也提供了 Windows，Linux 上的实现。基于 CAR 构件技术的应用程序，可以顺利的运行在任何具有“和欣”构件运行平台的系统上。

更重要的是，在“和欣”操作系统上，基于CAR构件技术实现的“和欣”浏览器将不再只是一个信息浏览器的工具，它将是位于操作系统之上，给客户提供各种服务，成为系统的中间件平台。“和欣”浏览器给用户提供一个视窗界面，所用的服务都可以在浏览器中找到，浏览器屏蔽掉那些对客户来讲复杂的计算。用户不但可以通过浏览器获得各种服务，而且可以在浏览器上开发自己的应用程序。构件化的“和欣”浏览器将对外提供很多的接口，通过这个构件接口，客户可以开发自己的应用程序，并将它嵌入到浏览器中。[34]如下图 5.1：

图 5.1 “和欣”浏览器拓扑图

5.2　浏览器的网络子系统构件化目的

浏览器的网络子系统完成 URL 请求下载功能，缓存管理、文件访问支持、其它协议（FTP 等协议）支持、并行访问、安全支持等功能，是浏览器中相对比较独立的功能模块。

（1）实现基于 CAR 构件技术的浏览器网络子系统。ElaScope 的网络子系统基于 C 语言实现，具有尺寸小、效率高的优点。但基于 C 语言的网络子系统只能静态编译时绑定，对象链接后不可拆卸，网络子系统与浏览器将链接成一个不可分开的整体，运行时无法改变模块之间的关系。通过对网络子系统的 CAR 构件化实现，将充分利用 CAR 构件技术的优势，实现高效、灵活的浏览器网络构件库。一是有利于设计、开发、测试和优化实现某一功能的单个构件；二是采用构件实现的系统比一个不可分的一个整体实现更容易移植；三是基于构件的系统可以动态替换原有构件，有利于软件的升级。四是容易积累大量可复用的网络协议栈构件，方便以后开发的复用。

（2）为“和欣”浏览器的其它模块的构件化提供借鉴。在现有的嵌入式浏览器中，很少有基于构件技术实现的浏览器，浏览器的构件化实现可以借鉴的模型很少。通过对浏览器网络子系统的构件化、构件化设计与实现，将为浏览器其它的模块的 CAR 构件技术实现提供参考，完成“和欣”浏览器的构件化实现。

（3）构件化的浏览器网络子系统将不仅为浏览器提供服务，也将通过接口为在浏览器上开发应用程序提供服务，使得基于浏览器开发的程序能简单使用浏览器的网络功能。同时也将作为“和欣”系统的标准构件库，为所有应用程序的开发提供服务。

5.3　构件化设计

5.3.1　构件化分析

采用构件化方法实现网络子系统的关键之处在于以下三点：

1) 如何划分构件的粒度；
2) 如何设计构件的接口；
3) 提高网络通信的效率。

第一点，需要考虑嵌入式浏览器构件化的网络子系统如何划分为多个构件，确定划分构件的粒度。软件构件化方式是对具有特定功能的构件对象进行设计与实现，再将构件对象组合起来生成软件系统的开发方式。在网络子系统的构

件化实现中，首先要解决的问题是如何确定划分构件的粒度，将模块划分为多个构件。可以采用大粒度的构件化划分方式，将整个网络子系统作为一个大的构件，定义构件接口，这样实现的构件化浏览器网络将为浏览器其它模块提供一个简单的接口。大粒度的划分在构件化的实现中比较简单，但构件内部将紧密的关联在一起，不利于维护与进一步的开发。

如果采用细粒度的构件划分，那么可以采用以模块为单位的小粒度划分构件。这种情况下，每个功能独立的模块或者协议划分为一个构件。细粒度的划分具有如下的优点：

(1) 针对不同网络应用的需求，能最大限度的利用领域相关知识进行调整，提高了性能。

(2) 底层构件可以在不影响顶层使用的情况下进行修改，替换。

(3) 方便用户开发调试新的协议，方便用户添加新的协议。

但是这种将模块分成很多小块的构件也会引进一些问题。首先是这些构件的配置比较困难，如果是许多小的协议构件，系统需要为这些构件的组装提供一个框架或者工具；其次把不同的功能分别在不同的小构件中实现不利于从整体进行优化编译器进行优化时，只能在一个小的构件内进行局部优化，而不能在构件间进行整体优化；另外不同构件间的调用会损失少量的效率，如果不同的构件是创建在同一个进程内，构件间的调用是直接以指针的方式进行的，如果是创建在不同的进程内，则不可避免的需要列集/散集来传递调用和相关参数。

在浏览器的网络子系统的构件化中，要充分考虑各种情况，包括嵌入式系统的特殊要求，确定合适的构件划分粒度。

第二点，要实现构件可自由替换，首先必须设计良好的构件接口。设计接口的原则是尽量保持接口的通用性和简洁性。当有新的功能添加时，可以通过继承实现新的接口方便的加入原有的框架，并不影响原来程序的正常使用。从而实现软件的无缝升级。

第三点，一个高效的网络通信需要满足以下二个方面：（1）多个协议具有统一的使用界面，（2）具有高效的缓冲机制，避免不必要的拷贝。

为了避免大量数据的拷贝，需要设计良好的数据结构，同时可以采取“共享内存”的方法。在网络协议各层次之间尽量以指针的方式传递数据。可以参

考文献[35][36][37][38]。

5.3.2　构件划分

“和欣”浏览器的网络子系统的构件划分参考了网络协议的构件化研究、嵌入式软件构件化设计、Firefox 浏览器的构件化网络子系统。浏览器的网络子系统采用中等的划分粒度，依据大的功能模块，将网络部分划分为多个构件。基于该构件划分的浏览器网络子系统具有良好的体系架构、支持协议的扩展与协议处理构件的动态加载的特点，方便用户增加新的通信协议的支持。参考图 5.2.

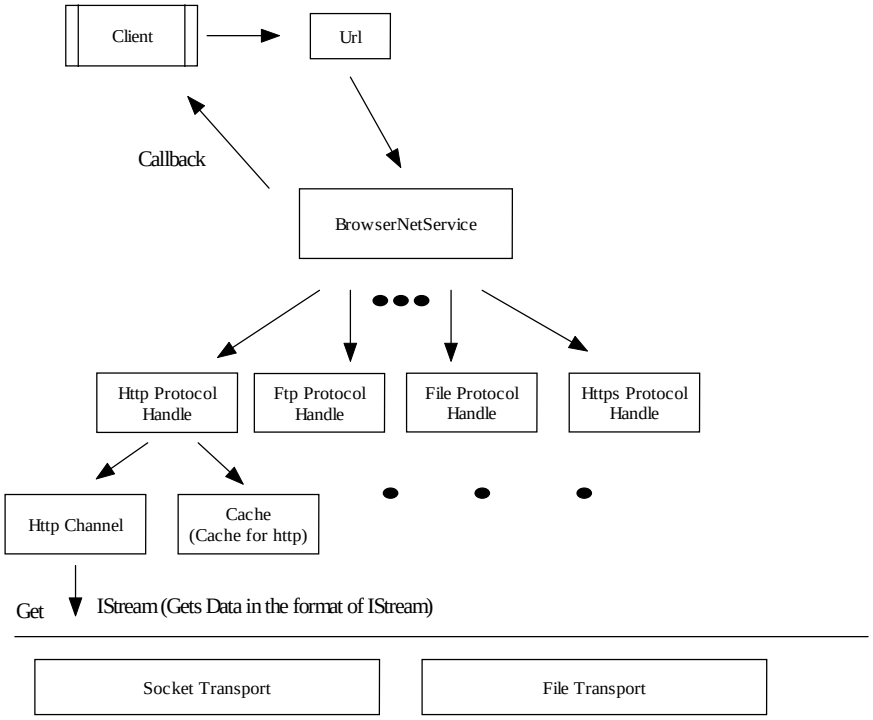

图 5.2 CAR 构件化网络子系统构件划分

构件化的网络子系统将由多个 CAR 构件组成：

- ✓ URL 解析构件。完成通用的 URI 解析工作，同时针对具体的协议类型，如 HTTP 协议，提供协议相关的解析。通过构件接口，可以访问解析后的 URL 属性，如主机名，端口，协议类型等。
- ✓ BrowserNetServices 浏览器网络服务管理构件。完成浏览器网络子系统的管理，完成不同协议处理构件的注册与加载工作，为不同的协议提供

打开 URL 的统一接口，提供回调接口注册等操作。

- ✓ HTTP 协议服务构件。完成 HTTP 解析功能与数据的 HTTP 协议异步传输功能。提供接口设置 Cache 属性、Cookies 属性，HTTP 代理属性等。
- ✓ FILE 协议服务构件。完成本地文件的读取功能，提供统一的处理接口。
- ✓ Cache 缓存管理构件。
- ✓ Transport 传输构件。提供网络层的传输抽象。

5.3.3　接口设计

本节介绍构件的接口设计。构件化的网络子系统由多个构件组成，每个构件又包括多个接口，接口中包含相关的方法。由于构件的接口很多，本节将只介绍关键构件的重要接口，对于其它的接口，将只给出构件接口的简单说明，并不介绍具体的接口。

5.3.3.1　URL 构件接口设计

URL 构件提供两种类型接口。第一种接口是协议无关接口 IUrl 接口，第二种接口是协议相关接口，包括 HTTP 相关的 IHttpUrl 接口，文件处理的 IFileUrl 接口、FTP 相关的 IFtpUrl 接口等。

在 URL 的定义中，有通用的 URL 定义，但针对不同类型的协议（URL 中定义为 Scheme），URL 的语法不同。IUrl 接口为通用的 URL 处理接口，完成通用操作，如协议检测、字符串逃逸等操作。而针对不同协议的接口完成该协议相关的 URL 的解析工作，本节将首先介绍 IUrl 接口，然后对对协议相关的接口将介绍 IHttpUrl 接口。

■ IUrl 接口介绍

IUrl 接口中的方法参考表格 5.1.

IUrl 接口方法	简介
HRESULT FromHex([in] char cDigit, [out] int * pnRet)	Return the decimal value of a hexadecimal digit.
HRESULT HexEscape([in] char cCharacter, [out]	Converts a specified character into its

EzByteBuf strEscape)	hexadecimal equivalent.
HRESULT HexUnescape([in] EzByteBuf pattern, [in] int nIndex, [out] char * pcHexUnscape)	Converts a specified hexadecimal representation of a character to the character
HRESULT IsHexdigit([in] char cCharacter, [out] BOOL * pbIsHexdigit)	Determines whether a specified character is a valid hexadecimal digit
HRESULT IsHexEncoding([in] EzByteBuf pattern, [out] int * pIndex, [out] BOOL *pbIsHexEncoding)	Determines whether a specified character is a valid hexadecimal digit.
HRESULT EscapeString([in] EzByteBuf str, [out] EzByteBuf escapeString)	Converts a string to its escaped representation.
HRESULT GetScheme([in] EzByteBuf uristring, [out] IUri_Scheme *pScheme)	Return the scheme of URL string

表 5.1 IUrl 接口方法

GetScheme 方法将返回输入字符串的协议类型，在 URL 构件中，支持的 HTTP 、FILE、FTP、HTTPS 几种 Scheme 类型，其它的类型将返回 IUri_Unknown。URL 构件中的 Scheme 类型定义如下：

```
Typedef enum IUri_Scheme {
    IUri_Http,
    IUri_File,
    IUri_Ftp,
    IUri_Https,
    IUri_Unknown };
```

■ IHttpUrl 接口

在本节我们将讲述 IHttpUrl 接口中的两个最重要的接口方法，其它的接口方法完成主机地址的检测（判断是否为本地主机）、访问 URL 字符串等操作，这里将不做介绍。

（1）语法解析方法:

```
HRESULT ParseUrl([in] EzByteBuf str)
```

该方法依据 HTTP URL 语法解析输入的 URL 字符串。Str 参数输入的 URL 字符串。返回 E_INVALIDARG 表示输入参数错误；返回 S_OK 标志方法调用成

功，输入字符串为合法的 URL，通过接口的其它方法可以获得语法解析的结果；返回 S_FALSE 标志方法调用成功，但输入的字符串为非法的 URL。

（2）获取语法解析后 URL 属性的方法

```
HRESULT GetProperty([in] IUri_Prop propertyType, [out] EzVar * pPropertyValue)
```

在调用 ParseUrl 返回 S_OK 后，标志输入的 URL 为合法的 URL，此时的 URL 字符串已经通过了语法解析，通过该方法可以获得 URL 的属性。在 URL 构件中定义了 URL 的各种属性，例如主机名、端口等信息。

5.3.3.2　Cache构件接口设计[39][40]

本节将首先讲述 Cache 构件的接口设计原则，然后讲述 Cache 构件的接口划分，由于 Cache 中包含了较多的接口方法，本节将不介绍具体的接口方法，而只对 Cache 构件的接口给出整体说明。

■ Cache 构件的接口设计原则

（1）存储策略。作为嵌入式浏览器的网络 cache 模块，充分应考虑了嵌入式设备的特点，尤其是考虑到嵌入式设备的资源有限性，如 memory constraints, bandwidth constraints。因此 cache 构件应提供接口设置支持多种管理策略，以适应不同设备的需要。存储策略定义数据的 cache 地点，可以支持的存储地点包括内存和文件。.

（2）替换算法。由于 Cache 容量的限制，当 Cache 达到最大容量以后，需要支持替换算法，将一些 Cache 内容替换。Cache 构件应提供基本的替换算法。Cache 中常用的替换算法有以下几种。

LRU：LRU（Least Recently Used）算法叫作“最近最少使用算法”，是 Cache 中较常用的算法，它是替换最近一段时间内最少被访问过的数据。该算法又被称作“最久未使用算法”，是因为它将最久没有使用的数据从 Cache 中替换掉。这是一种高效、科学的算法，可以把一些频繁调用后再不需要的数据从缓存中淘汰出去，提高缓存的利用率。

LFU：LFU（Least Frequently Used）算法替换缓存中最少被访问的对象。

LRU-K：替换缓存中最近第 K 次访问时间最远的对象。当被比较缓存中的对象还未被访问 K 次的时候，算法回退到 K-1，直到 K=1 和 LRU 一样。实际上

LRU-1 就是 LRU。LRU-K 结合了对象访问的时新性和访问频率，文献中一般的结论是 K 值越大性能越好，而在 K=2 时，性能提高比较明显，且实现代价小。

SIZE：该算法替换最大的对象。这一算法只考虑对象大小，试图保留更多的小对象来提高命中率。

GD-SIZE：按效用函数，替换效用最小的对象。每一个对象的效用 Ki 计算如下：

$$Ki = Ci / Si + L \qquad (1\text{-}1)$$

其中 Ci 是对象 i 装入缓存的代价，Si 是对象 i 的大小，L 是缓存运行的年龄因子，从 0 开始，当对象 j 被替换时，L=Kj。文献[3]中分析了的 GD-SIZE 的多个形式，当 Ci 取 1 时，算法称为 GD-SIZE(1)。在同一 L 值，即同一年龄的对象中，它优先替换大对象，具有最好的缓存命中率，但字节命中率较差。另外还有一种把网络数据包个数作为 Ci 的算法，称为 GD-SIZE(Packets)，它具有平衡的命中率与字节命中率。

（3）Cache 接口划分。为了提供方便的 Cache 访问，在 Cache 构件中，将接口分成几种类型，分别完成各自的服务类型，将提供给用户更好的 Cache 访问。

■　Cache 接口划分

Cache 对外提供三个接口：ICacheMgr，ICacheService，ICacheEntry

ICacheMgr 接口：在 ICacheMgr 接口中，提供设置与读取 cache 的管理属性，包括：CacheStoragePolicy 确定 Cache 的存储位置、Cache 大小的设置以及其它的相关参数的设置、Cache 的替换算法的设置、获得 Cache 当前属性的接口方法，通过这个接口方法，可以获得 Cache 的各种属性；

ICacheService 接口：通过 ICacheService 接口完成 CacheEntry 的创建与访问。在 Cache 中，数据以 Entry 的形式进行存储，CacheEntry 的设计在后面详细叙述。Entry 中存储的数据包括了 data 与 metadata 以及 EntryID。 从 ICacheService 接口将获得 ICacheEntry 接口，通过这个接口完成数据的读取与输入操作。

ICacheEntry 接口：通过 ICacheEntry 接口可以完成数据的读取，写入等操作。在 Cache 中，存储的单位为 Entry。在 Cache 中，将 Entry 定义为包括唯一 id，

数据，元数据的三元组，即<Entry_ID, Entry_Data, Entry_MetaData>。Entry_ID 标记一个 Entry。Entry_Data 存储主数据，Entry_MetaData 标记描述 Data 的相关元数据。在 Cache 的实现中，Entry_ID 定义为 EzByteBuf，由 cache 的调用者保证输入的 Entry_ID 的唯一性。Entry_Data 定义为 EzByteBuf，存储用户数据，对于用户数据，cache 本身不关心具体的数据格式与内容。Entry_MetaData 为<name, data>数据对，name 为 EzByteBuf 类型，data 为 EzByteBuf 类型。

例如对于 HTTP 数据的 cache，从 HTTP Server 获得数据后，用于向 Server 请求的 URL 字符串将作为 Entry_ID，而从 Server 获得的 html 或 image 等数据将作为 Entry_Data，HTTP Server 返回的 HTTP Header 将作为 Entry_MetaData 以<"Http_Response_Header", "the content of http header …">形式存储。

Cache 的访问者将通过 Entry_ID 获得相关的信息。

一个 Entry 有唯一的一个 id, 唯一的一个数据区，但可以有多个 MetaData 描述。

5.3.3.3　HTTP 构件接口设计

在构件化的 HTTP 服务中，由两个构件提供 HTTP 服务：HTTP Handler 构件与 HTTP Channel 构件。HTTP Channel 构件建立一条客户端到服务器的 HTTP 连接，完成 HTTP 协议的解析与通信工作。在 HTTP Channel 构件中，只完成最简单的数据通信功能。HTTP Handler 构件通过 HTTP Channel 构件完成 HTTP 解析与通信，同时提供接口设置 HTTP 服务属性，包括多线程并行访问设置、代理设置、Cookies 管理、Cache 管理等功能等。在 HTTP Channel 构件的基础上，HTTP Handler 构件提供了功能更强大的服务。在只需要使用最简单的 HTTP 服务的情况下，用户直接调用 HTTP Channel 构件。在需要使用功能完善的 HTTP 服务的情况下，可以使用 HTTP Handler 构件提供的服务。

5.3.3.3.1　HTTP Channel 构件接口

HTTP Channel 构件提供三个接口，参考表格 5.2:

接口名称	接口描述

IhttpWebRequest	接口提供方法，根据 URL 请求，生成 HTTP 请求信息，建立与服务器的连接，完成请求发送工作。在发送成功的情况下，通过改接口可以获得 IHttpWebResponse 接口。
IhttpWebResponse	接口提供方法获得服务器的返回信息，为返回的 HTTP 协议响应消息进行解析，在成功返回的情况下，通过该接口可以获得 INetworkStream 接口。
InetworkStream	接口提供方法完成服务器的数据的获取与发送。

表 5.2：HTTP Channel 构件接口描述表

■ IHttpWebRequest 接口中的方法

在 IHttpWebRequest 接口中包括三类方法：HTTP 协议属性设置接口方法、生成 HTTP 请求信息的接口方法、发送 HTTP 请求并获取 IHttpWebResponse 接口指针的方法。

HTTP 协议属性设置接口方法包括设置 SetAccept、SetAllowAutoRedirect、SetConnection、SetKeepAlive 等协议相关的设置接口方法。

IHttpWebRequest 中提供两个方法完成 HTTP 请求信息的生成与发送工作：

方法声明：
HRESULT CreateFromString([in] EzByteBuf str);
HRESULT CreateFromUri([in] IHttpUrl * IUriRef);

方法描述：
　　根据输入的 URL 请求字符串或输入的 IHttpUrl 接口指针，依据 HTTP 协议的设定，生成 HTTP 请求消息，并发送给目标服务器。

在请求信息发送成功以后，可以通过 GetResponse 方法获得 IHttpWebResponse 接口指针。

方法声明：
HRESULT GetResponse([out] IHttpWebResponse ** pResponseRef);
方法描述：
　　获取 IhttpWebResponse 接口指针。

■ IHttpWebResponse 接口中的方法

在 IHttpWebResponse 接口中，最重要的方法就是获得服务器返回的 HTTP 状态的方法与获得服务器数据的方法。

获得服务器状态的方法：

方法声明：
HRESULT GetStatusCode([out] HttpStatusCode *statusCode);
方法描述：
获取 HTTP Response 消息中标志的 HTTP 返回状态。状态信息保留在输入参数 statusCode 中。

获得服务器数据流的方法：

方法声明：
HRESULT GetResponseStream([out] INetworkStream ** pStreamRef);
方法描述：
当通过调用 GetStatusCode 判断出服务器成功的返回数据以后，通过改方法可以获得数据流的接口指针，通过 INetworkStream 中提供的方法完成数据的获取。

除了上面的方法以外，IHttpWebResponse 接口获取 HTTP Response 消息内容与属性的方法。

- INetworkStream 接口中的方法

INetworkStream 接口中提供了完成数据流操作的方法，包括流的创建、关闭、设置属性、读写数据等操作。

5.3.3.3.1　HTTP Handler 构件接口

HTTP Channel 构件仅仅提供 HTTP 通信的最基本功能，在 HTTP Channel 的基础，HTTP Handler 通过在内部维护多个 HTTP 连接，提供并行 HTTP 通信的功能，并且构件支持缓存、代理、Cookies、回调等工作模式。HTTP Handler 构件提供三个接口,参考表格 5.3：

接口名称	接口描述
IHttpHandlerMgr	接口完成 HTTP Handler 的管理。提供接口完成设置缓存、Cookies、代理、设置并行处理的线程数目、设置 HTTP 回调接口、设置 HTTP 协议工作模

	式。
IHttpHandler	接口提供方法完成 URL 请求的发送操作，还提供方法完成 HTTP Response 消息获取的方法。
IHttpHandlerCallBack	定义 HTTP 回调接口。

表 5.3 HTTP Handler 构件接口描述表

■ IHttpHandlerMgr 接口方法

IHttpHandlerMgr 中包含了较多的接口方法，表格给出了接口方法的简单描述。Cookies 管理方法提供完成 Cookies 属性的设定，包括设定 Cookies 模式（内存保留 Cookies 还是文件系统保留 Cookies）、设定 Cookies 路径、加载 Cookies、Cookies 启动与关闭；KeepAlive 连接模式管理方法设定是否使用 HTTP/1.1 中的 KeepAlive 连接模式；多线程管理设置工作线程数目，即最大的并行 URL 请求处理数目；Cache 管理设置 Cache 的各种属性，同时设置是否启动 Cache；代理管理方法完成 HTTP 代理的设置；回调管理完成回调接口的设定。参考表格 5.4。

接口方法类型	接口方法名称
Cookies 管理	EnableCookies 、 DisableCookies 、 RemoveAllCookies 、 SetCookiesMode、SetCookiesPath、LoadCookies
KeepAlive 连接模式管理	EnableKeepAliveMode 、 DisableKeepAliveMode 、 IsKeepAliveMode
多线程管理	SetThreadMode、GetThreadMode
Cache 管理	EnableCache、DisableCache、RemoveCache、CleanUpCache、SetCacheMode
代理管理	EnableProxy、DisableProxy、SetProxy、GetProxy
回调管理	AddEventHandler、RemoveEventHandler

表 5.4:IHttpHandlerMgr 接口方法描述

■ IHttpHandler 接口方法

IHttpHandler 接口提供方法处理 URL 请求。在 IHttpHandler 中最重要的接口方法是处理 HTTP 协议 URL 请求的方法：

方法声明：
HRESULT OpenUrl([in] EzByteBuf ebUrl, [in] EzByteBuf ebPostData,
[in] EzByteBuf ebBindData, [in] BOOL bRootUrl,
[out] UINT *pKey);
方法描述：
对于 HTTP 协议 URL 请求，URL 请求将放到 HTTP Handler 构件的请求队列。URL 请求处理完成后将通过回调注册的回调接口的方式返回。
参数描述：
ebUrl：URL 字符串
ebPostData：需要向服务器发送的数据（处理 HTTP POST 方法）
ebBindData：回调时返回给回调接口的绑定数据
bRootUrl：请求的 URL 类型
pKey：返回给接口调用者的唯一标志这次 URL 请求的主键
返回值：
E_INVALIDARG：存在输入参数错误
S_OK: 方法调用成功

■ IHttpHandlerCallBack 接口方法

在 HTTP Handler 构件中定义回调接口，HTTP Handler 在 URL 处理完成后，将回调注册的回调接口方法通知客户处理获得数据。下面给出简单的描述：

方法声明：
HRESULT EventHandler([in] HttpEvent enumEvent, [in] UINT nKey,
[in] EzByteBuf ebData,
[in] EzByteBuf ebErrString);
方面描述：
定义回调接口中必须处理回调的接口方法。HTTP Handler 回调客户注册的接口函数，通知 OpenUrl 请求的处理结果。enumEvent 中定义了各种 HTTP 情况。

IHttpHandlerMgr 接口中提供了 AddEventHandler、RemoveEventHandler 两个接口方法完成回调接口的注册与取消。

5.3.3.4　BrowserNetServices 构件接口设计

在“和欣”浏览器中，BrowserNetServices 构件为浏览器引擎提供网络服务，完成协议处理的动态加载过程。BrowserNetServices 构件提供两个接口，一个接口提供方法注册可以支持的协议以及相应的处理构件，同时注册相应的回调接口。另一个接口提供通用的 URL 请求操作。

■　IBrowserNetServiceMgr 接口方法

IBrowserNetServiceMgr 接口中定义了枚举类型的 ProtocolType，对每个类型提供了注册与取消协议处理构件的方法：

```
方法声明：
HRESULT AddProtocolHandler([in] ProtocolType enumProtocolType,
                           [in] EzStr ezstrUUNM);
HRESULT RemoveProtocolHandler([in] ProtocolType enumProtocolType);
方面描述：
    注册或取消相应协议的处理构件（HTTP 协议处理构件，FTP 协议处理构件等）
```

IBrowserNetServiceMgr 同时提供回调接口的注册与取消方法，在简单的加载协议处理构件的方法基础上，提供了进一步按照配置属性加载协议处理构件的方法，这里没有做进一步的说明。

■　IBrowserNetService 接口方法

IBrowserNetService 提供通用的 OpenUrl 接口方法。通过改方法可以完成通用的 URL 请求。该方法被调用后，BrowserNetServices 构件将检测输入的 URL 类型，然后判断是否已经启动了相应的协议处理构件，然后将请求传递给相应的处理构件。

5.4　构件化实现

在“和欣”操作系统上，我们已经实现了上面的构件化的浏览器网络系统。在构件化的实现中，内部使用了很多的 ElaScope 网络子系统的代码，相比 ElaScope 网络子系统的实现，构件化的网络子系统的实现中没有使用 CCC 链完成事件的传递与处理，而是基于构件接口的调用方式来完成事件处理。在构件

化的网络实现中，采用了回调构件接口的方法来完成数据的回调，而不是通过函数指针的直接回调的方式。成功打开一个 HTTP URL 请求的处理流程如下图：

BrowserNetServices 构件接收到 URL 请求以后，调用 URL 构件进行解析，对于 HTTP URL 请求，将把请求交给 HTTP Handler 构件处理。HTTP Handler 构件首先查找是否在缓存中，如果在缓存中，获取数据，通过回调接口返回给调用者。如果没有将完成代理、Cookies 的设置，然后根据 Handler 中的线程设置情况，将请求的 URL 放入队列当中，直到有空闲的工作线程存在，这时工作线程将使用 HTTP Channel 构件完成 HTTP URL 请求的发送过程，同时完成数据获取，得到数据后通过回调接口，将数据返回给浏览器的其它模块。参考图 5.3：

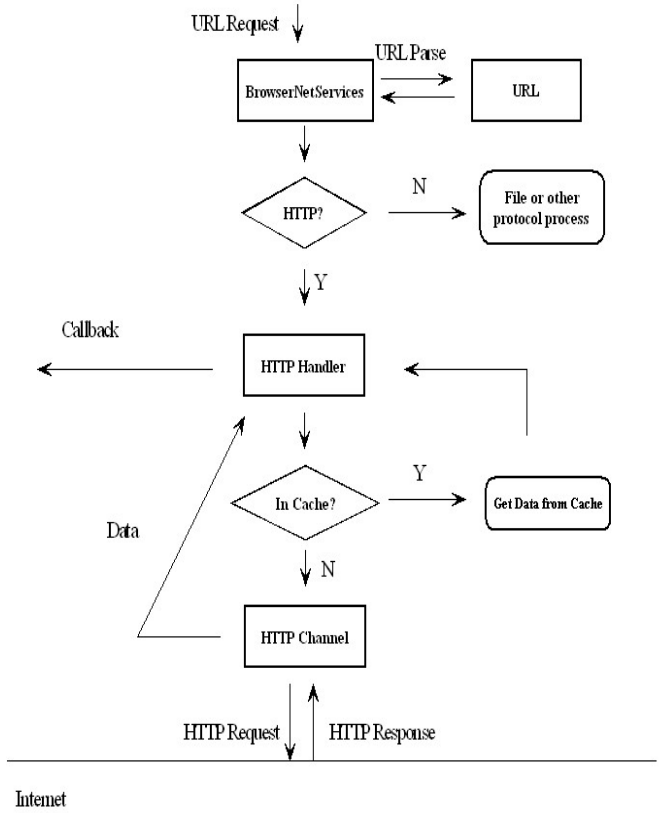

图 5.3 完成 HTTP URL 请求的控制流程

通过对网络子系统的构件化，我们实现了一个高效、灵活的浏览器网络构件库，可以为浏览器与“和欣”系统提供 WWW 服务，同时也为浏览器的整体构件化实现提供了借鉴。

第 6 章　工作总结与展望

6.1　工作总结

随着中高端嵌入式设备的发展，嵌入式浏览器在嵌入式系统中的应用越来越广泛。嵌入式设备的多样性，要求嵌入式浏览器具有高度可配置性，可定制，可剪裁。而在以普适计算，web 服务为特点的网络计算时代，嵌入式浏览器是嵌入式操作系统不可缺少的一个部分。在“和欣”操作系统上，设计与实现嵌入式浏览器，并实现“和欣”浏览器就是系统的“中间件”的设想，具有重要的研究意义。

论文提出了“和欣”操作系统上浏览器体系架构，该体系架构具有模块化好，模块之间耦合性小，易扩展等特点。该体系架构在包含基本功能模块的基础上，参考了 DOM 的机制，对页面数据进行对象化，抽象出文档对象树模块。以对象化的方式对页面管理，使得对页面的管理更加方便，逻辑性更好，同时有利于提供对 DOM 的支持。

网络子系统是浏览器中的重要组成系统，论文参考已有浏览器网络子系统的设计，并参考了相关 Internet 技术，提出了基于面向对象技术的网络子系统的设计方案。该设计方案充分考虑了嵌入式系统的特点，具有灵活、高效的特点。基于设计方案，我们已经实现了一个功能完备的嵌入式浏览器网络子系统，充分考虑了可裁减性和高效性，该系统面向嵌入式领域、稳定、实用、功能完善。

“和欣”操作系统是面向未来的网络操作系统，CAR 构件技术是“和欣”的关键技术。在浏览器网络子系统的面向对象技术的实现基础上，论文进一步研究了嵌入式浏览器的网络子系统的构件化工作，提出了采用 CAR 构件方法来实现浏览器网络子系统的设计方案，讨论了采用构件化思想实现浏览器网络子系统涉及到的问题，包括构件划分，构件接口的设计，协议构件的动态加载等。相对于已有的浏览器网络子系统的实现，基于 CAR 构件技术的浏览器网络子系统具有良好的体系架构、支持协议的扩展与协议处理构件的动态加载的特点，

方便用户增加新的通信协议的支持，同时有利于用户或者开发者积累更多可复用的构件。论文的网络子系统的构件化工作为浏览器的构件化研究提供很好的借鉴。

6.2　工作展望

在“和欣”系统上，实现基于 CAR 构件技术的浏览器，使浏览器成为“和欣”操作系统上的中间件平台，给客户提供各种服务，是一个规模很大的项目。本人在硕士期间主要参与了该项目的前期工作，为后面的工作做出了铺垫。目前我们已经在“和欣”实现了一个嵌入式浏览器 ElaScope，并完成了网络子系统的构件化工作，实现了“点击运行”的原型系统。在此基础上，还需要完成很多工作：

1. 浏览器的其它功能模块的构件化工作；
2. 脚本语言的支持，通过支持脚本语言，可以更好的实现浏览器和客户端的交互，同时可以更好的控制页面元素；
3. .浏览器和 CAR 构件的消息互通，之间协议的抽象，可以通过脚本语言操作 CAR 构件，并且 CAR 构件可以发消息给浏览器，完善浏览器的扩展机制；
4. .对浏览器的优化，可以使浏览器变得更小，更快，从而满足嵌入式平台的需要；
5. “点击下载”运行机制的研究，是网络服务编程模型的基础；

参考文献

[1] 彭莉娟. 嵌入式浏览器的研究：[硕士学位论文]. 北京：北京工业大学，2001

[2] 杨玉平 嵌入式浏览器 DeltaBrowser 的设计与实现：[硕士学位论文]. 成都：电子科技大学，2003

[3] HTTP://www.embedded-tech.com.cn

[4] HTTP://www.embed.com.cn

[5] HTTP://www.mozilla.org

[6] HTTP://www.dillo.org

[7] 郭相国. 嵌入式浏览器研究: [硕士学位论文].成都：电子科技大学计算机应用专业，2003

[8] HTTP://www.w3.org

[9] 上海构件库. Available online at HTTP://www.sstc.org.cn/.

[10] Microsoft Corporation. Distributed Component Object Model (DCOM) - Downloads, Specifications, Samples, Papers, and Resources for Microsoft DCOM. Available online at HTTP://www.microsoft.com/com/tech/DCOM.asp.

[11] Object Management Group, Inc Common Object Request Broker Architecture (CORBA/IIOP). Available online at HTTP://www.omg.org/technology/documents/formal/corba_iiop.htm

[12] Sun Microsystems, Inc. Specifications -- Enterprise JavaBeans. Available online at HTTP://java.sun.com/products/ejb/docs.html.

[13] Box, Don. Essential COM. Menlo Park, CA: Addison-Wesley Publishing Company, 1998. ISBN 0-201-63446-5.

[14] 潘爱名. COM 原理与应用. 清华大学出版社, 1999. ISBN 7-302-02268-2

[15] Major, Al. COM IDL and Interface Design. Chicago, IL: Wrox Press Inc., 1999. ISBN 1-861-00225-4.

[16] Microsoft Corporation. The Component Object Model Specification. Available online at HTTP://www.microsoft.com/com/resources/comdocs.asp

[17] Dale Rogerson（美）著，杨秀章 译. COM 技术内幕(Inside COM)，清华大学出版社

[18] 陈榕，刘艺平. 技术报告：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), 2003.

[19] 陈榕，苏翼鹏，杜永文 等. 构件自描述封装方法及运行的方法，中国专利，1514361，2004.07.21

[20] 和欣资料库，HTTP://www.koretide.com.cn

[21] 徐会建. 嵌入式浏览器 DeltaBrowser 表示层的设计与实现，[硕士学位论文].成都：电子科技大学软件与理论专业，2004

[22] 申波. 基于 XML DOM 的嵌入式浏览器研究及核心模块的设计和实现,[硕士学位论文].成都：电子科技大学计算机应用技术专业，2002

[23] 王志利. 嵌入式浏览器的研究与实现,[硕士学位论文].成都：电子科技大学计算机应用专业，2003

[24] Jim Beveridge（美）著，候捷 译. Win32 多线程程序设计，华中科技大学出版社

[25] IETF，URL 规范，IETF 网站

[26] IETF，HTTP/1.1 规范，IETF 网站

[27] Duane Wessels. Web Caching. Publisher: O'Reilly, June 2001

[28] HTTP://www.web-caching.com

[29] Brian D. Davison. A Web Caching Primer. Internet Computing, IEEE, Vol.5, No.4, July/August 2001, p38-45.

[30] S. Jin and A. Bestavros, "Sources and characteristics of Web temporal locality," In Proceedings of MASCOTS'2000.

[31] Peter K. Pearson, fast hashing of variable length text strings. Communications of the ACM, 1990,33(6), p676~678.

[32] LI Xiao-Ming, FENG Wang-Sen. Two Effective Functions on Hashing URL(In Chinese). Journal of Software. Vol.15, No.2, 2004

[33] McKenzie BJ, Harries R, Bell T. Selecting a hashing algorithm. Software Practice and Experience, 1990,20(2), p208~210.

[34] 朱剑民. 陈榕. 倪光南，“和欣”操作系统的浏览器设计模型 计算机工程与应用 2003 年 13 期

[35] ZHU Lixin, WANG Feiyue. Construction of Application Specific Embedded Operating Systems Based on Component Technique (In Chinese). Computer Engineering. Vol.30 No.3, Feb 2004

[36] 石爱国. 嵌入式组件技术研究：[硕士学位论文]. 西安：西北工业大学，2003

[37] 江峰. 构件化嵌入式系统的研究与开发：[硕士学位论文]. 杭州：浙江大学，2004

[38] Argent，Robert M. An overview of model integration for environmental applications - Components，frameworks and semantics，Environmental Modelling and Software，v 19，n 3，March，2004，p 219-234

[39] E.J. O'Neil, P.E. O'Neil and G. Weikum, "The LRU-k Page Replacement Algorithm for Database Disk Buffering", in Proc. of the 1993 ACM SIGMOD Conference, pp 297--306, 1993.

[40] P. Cao and S. Irani: Cost-Aware WWW Proxy Caching Algorithms., Proceedings of the USENIX Symposium on Internet Technologies and Systems, December 1997.

致 谢

本课题承蒙国家 863 重点软件项目资助，特致殷切谢意。

衷心感谢我的父母、弟弟对我一直以来的关心和支持。

衷心感谢导师殷人昆教授，陈榕教授对本人的精心指导。他们的言传身教将使我终生受益。感谢王小鸽，杨维康，张素琴，吴季风，陈志诚老师对本人的关心和帮助。

衷心感谢上海科泰世纪有限公司工程部“和欣”2.0 组叶忠强、周华伟、杨义斌、杨晓亮、刘亚东、杨洋等的热心指导与帮助。感谢计研 55 班全体同学曾经给予过我的帮助。感谢赵金钟、周跃平等同学对我的帮助！

声 明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____日　期：_____

个人简历、在学期间发表的学术论文与研究成果

本人简历

1997.9-2001.7 在河北大学计算机科学与技术系学习，并于 2001 年 7 月获得工学学士学位

2002 年 9 月考入清华大学计算机科学与技术系计算机科学与技术专业攻读硕士学位，导师殷人昆教授

攻读硕士学位期间发表的学术论文

[1] 李洪涛，郑任持，陈瑢，殷人昆。一种支持面向侧面的构件工程的方法 计算机应用与软件，接受待发表.

[2] 郭强，李洪涛，赵金钟，叶忠强，陈榕，周立柱. 嵌入式浏览器架构研究与实现. 计算机应用与软件，接受待发表