DLNA 技术在媒体中心中的应用研究

摘 要

随着信息技术的飞速发展，现代家庭已经逐渐演变成数字化、网络化和智能化的媒体中心。人们从家电设备、移动设备和电脑设备上获取、查看和管理越来越多的数字媒体信息。人们希望在家中的任何地方都能方便地享受丰富多彩的媒体内容，而不考虑这些媒体内容储存在什么设备上。然而，传统的网络技术由于安装、配置和管理过程非常复杂，无法满足人们的这些要求。

数字家庭充分集成了计算机技术和家电技术。目前世界上已经出现了很多数字家庭标准化组织，如 DLNA、ITU-T、ECHONet、OSGI、IHA、UOPF 和闪联等。本文首先概述了数字家庭标准化的发展现状，分析了各个标准化组织涉及的业务方向和使用的主要技术。在此基础上，选择在家电设备互连和媒体内容管理方面有显著优势的 DLNA 技术为课题展开研究。DLNA 技术以 TCP/IP 协议和 UPnP 技术为基础，通过一个无缝的、可互操作的网络将因特网、移动设备和广播网络集成起来。它主要包含物理连接、网络传输、设备发现与控制、媒体管理与控制、媒体格式和媒体传输协议六个功能组件。本文研究了 DLNA 技术实现家电设备互操作的基本原理和实现机制，重点分析了 DLNA 技术中的 UPnP 设备架构和 UPnP 音频/视频架构。UPnP 设备架构实现了设备的发现、互连和互操作；UPnP 音频/视频架构实现了媒体内容的识别、发布和管理。本文通过实现 UPnP 设备架构和 UPnP 音频/视频架构，实现了 DLNA 协议，并且详细描述了 DLNA 协议的实现过程。其中在实现 UPnP 设备架构的过程中，本文在系统运行效率、稳定性和可移植性方面做了很多分析，提出了一种基于阻塞/唤醒的单线程轮询机制。这种机制将设备发现与控制过程中的所有可能触发的事件组成一个事件链表，使用一个线程来维护。线程初始时处于阻塞状态，当有事件触发时，就唤醒线程开始对事件链表进行轮询；对于链表中的每一个结点，线程跳转到对应的事件处理入口地址去判断并处理触发的事件；在轮询处理完事件链表之后，线程再次进入阻塞状态，直到下一次事件触发或线程退出。这种机制在东软软件股份有限公司 IA 事业部研发项目组的应用中表现出了很高的效率和稳定性。在实现 UPnP 设备架构和 UPnP 音频/视频架构的基础上，本文实现了一个 DLNA 控制点系统，以此展示了 DLNA 技术在数字家庭媒体中心中的应用。

关键词：DLNA；UPnP；互操作；控制点；轮询

The Research and Application of DLNA Technology in Media Center

Abstract

As the development of information technologies, modern houses have gradually become the media centers of digital, network and intelligence. People are acquiring, viewing, and managing an increasing amount of digital media on devices in the consumer electronics, mobile device, and personal computer domains. People want to enjoy rich multimedia content in any location throughout the home, regardless of where the content is physically stored. However, conventional network technologies can not serve those requirements due to the complexity associated with their installation, configuration, and management.

Digital Home is the integration of technology of Computer and Consumer Electronics; several standard organizations of digital home have been set up in the nowadays, such as DLNA, ITU-T, ECHONet, OSGI, IHA, UOPF and IGRS. In this thesis, the development of Digital Home Standardization is summarized and the major business fields and technologies of each standard organization are discussed. Of these fields, DLNA, which is advantageous in interoperability between CE (Consumer Electronics) devices and media content management, is emphasized in this thesis. Based on TCP/IP protocols and UPnP technology, DLNA technology integrates the Internet, mobile, and broadcast networks through a seamless, interoperable network. It covers physical connection, network transports, device discovery and control, media management and control, media formats and media transport protocols. In this thesis, the basic principle and realization principle of DLNA interoperability is focused on and UPnP device architecture, which completes device discovery, connection and communication, and UPnP Audio/Video architecture, which completes media content identification, distribution and management, is emphasized. By realizing the UPnP device architecture and UPnP Audio/Video architecture, DLNA protocol is implemented and the detailed implementation process is described in the thesis. In the realization process of UPnP device architecture, great efforts are made to guarantee the efficiency, stability and portability of the system. In this thesis, a new single-thread poll mechanism with block and wake functionality is implemented. This mechanism provides an event chain, which is managed by a single thread, to manage all events which could be triggered during the device discovery and control process. The thread is initially in block state. When an event is triggered, the thread is waked and starts polling the

event chain. For each node in the chain, the thread jumps to the event handle address to deal with the occurred event. After polling all nodes in event chain, the thread completes the transaction of all occurred events. After that, it returns to block state again, until next event is triggered or the thread terminates. The application of this mechanism in the projects by R&D group of Neusoft Co. Ltd. IA division manifests its efficiency and stability. Based on the realization of UPnP device architecture and UPnP Audio/Video architecture, a DLNA control point system is established to present the application of DLNA technology in the digital home media center.

Key words: DLNA; UPnP; interoperability; control point; poll

独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：袁辉

日　　期：2006.02.23

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

（如作者和导师不同意网上交流，请在下方签名；否则视为同意。）

学位论文作者签名：　　导师签名：

签字日期：　　签字日期：

第一章 引 言

1.1 课题研究背景

随着经济全球化和社会信息化的日趋发展，信息技术日新月异，信息产业蓬勃发展；这些深深地影响和改变着人们的生产和生活方式，并且已经成为现代经济发展的重要推动力。在信息技术加快向网络化、数字化、智能化方向发展的今天，因特网、微电子、软件、移动通信以及数字视频技术领域和相关产业，将成为目前最有潜力的发展领域。

在家电领域，从数字电视开始推广，到家庭网络产品的日益成熟；从数字化产品应用到家庭医疗保健，再到数字导航系统、广播系统在汽车等家用电子领域的使用……这一切都说明消费类高科技电子产品已经开始进入千家万户。二十一世纪的现代家庭，已经逐渐演变成包含各种高科技电子产品的媒体中心，为人们的生活带来了丰富的内容和极大的便利。

目前，媒体中心中存在三大领域：

(1) 电脑互联网世界，电脑与电脑外设进行通信；

(2) 机顶盒与传统家电的广播世界；

(3) 多媒体电话、掌上电脑、笔记本电脑等类似设备的移动世界，提供无与伦比的连接性和家庭内外的自由移动性。

人们已开始在这些家电设备、移动设备和电脑设备上获取、查看和管理越来越多的数字媒体内容，希望能够在家中不同的地方通过不同的设备更加轻松便捷地欣赏这些内容，能够在家中实现这些设备的互操作，而无论这些内容存储在哪里。

这就是所谓的“数字家庭”。 数字家庭是一种理念的发展，该理念认为，家庭环境中的个人计算机、消费电子产品和移动设备应能通过有线或无线网络，无缝协调地工作，从而共享家庭环境中的数字媒体资源。

随着各种信息设备功能越来越强大以及网络技术特别是无线网络技术的飞速发展，设备间如何更加方便、智能地互连，如何更好地协同工作，已经成为人们日益关注的焦点。

制造商所面临的主要问题是，开放工业标准通常过于灵活，由不同厂商构建的产品经常不能很好地进行互操作；目前基于专有垂直实施的端到端解决方案可以较早地向市场推出产品，但对于快速涌现的新型产品的影响力又太小。

在这种背景下，国内外出现了很多由家电设备厂商、网络设备厂商等主导的数字家庭标准化组织，他们为家庭中不同类型的设备提供互连标准和规范，以实现可互操作的家庭网络。

1.2 数字家庭标准化的发展现状

目前从事数字家庭标准化的组织非常多，如国外的 DLNA、ITU-T、ECHONet、OSGI、IHA、UOPF 和国内的闪联（IGRS）等等。每个标准化组织在数字家庭中所涉及业务和使用的技术各不相同。

DLNA（Digital Living Network Alliance，数字生活网络联盟）是一个非赢利的数字家庭标准化组织，它创建于 2003 年 6 月，由 17 家创始成员组成，包括 Intel、富士通、惠普、联想、IBM、Kenwood、松下、微软、NEC、诺基亚、飞利浦、三星、Sharp、索尼、STMicroelectronics 和 Thomson 等，主要涉及家庭影音娱乐等业务应用方面。DLNA 旨在根据开放式工业标准制定一个互操作框架[1]，在数字家庭媒体中心中建立一个集中管理个人电脑、家电设备和移动电子设备的互操作网络，创造一个能够共享和发展全新数字媒体和内容服务的无缝环境，以实现跨行业的数字融合。

ITU-T（国际电联电信标准化部门）是一个电信领域的全球标准化组织[2]，成立于 1993 年，它的前身是国际电报和电话咨询委员会（CCITT）。ITU-T 研究和制订除无线电以外的所有电信领域标准，已通过的建议书有 2600 多项。ITU-T 分为 16 个研究组，研究范围涉及电信网络税费政策、电信管理网（TMN）、综合宽带电缆网络和音/视频的传输、数据通信网络、IP 网络、光传送网、多媒体业务、系统和终端、信令协议、电信软件、移动通信网络和通信设备安装、施工等各个方面。ITU-T 在家庭方面涉及的业务和应用主要是基于电信网（含 Internet）的业务和应用。

ECHONet（Energy Conservation and Homecare Network）协会于 1997 年 12 月成立[3]，由日立公司联合日本国内 152 家企业组建而成，期间得到了日本政府的支持。该组织的目标是为家电、传感器和控制器提供通信接口和协议，其研究内容包括用于 ECHONet 的通信中间件、通信接口、通信协议等，最新的标准是 ECHONet Version3.0。

OSGI（Open Service Gateway Initiative，开放式服务网关组织）成立于 1999 年 3 月[4]，由 IBM 发起，现有成员 80 多个，包括电信服务商、设备商、电脑、家电、控制系统供应商，旨在建立一个开放的服务规范[5]，制定家庭外部网络向家用电子设备提供业务所需的接口和服务标准，目前已推出了 3.0 版本。

IHA（Internet Home Alliance，互联网家庭联盟）也是成立较早的家庭网络组织[6]，于 2000 年成立，主要成员包括思科、GM、惠普、IBM、松下、SUN、惠而普及微软等公司，成员有 30 多家，还包括一些通讯设备、自动化软件开发商[7]及厨具制造商和零售商等。该联盟主要解决白色家电在家庭网络中的互连、互联网接入及相关服务，如在出现故障时能够与厂商自动接通以确定故障部位。

UOPF（Ubiquitous Open Platform Forum）是一个刚成立不久的数字家庭标准化组织[2]，于 2004 年 2 月 11 日成立，主要成员包括松下、索尼、NEC、东芝、三菱、三洋、先锋、日立等 10 家日本电子厂商和 NTT 等 4 家互联网服务商，以制订通过家庭内部网

络及互联网等实现数字家电互连的标准，让任何用户都能简易操作连接到宽带网络的数字家电。UOPF 的具体目标包括：简单设定，不需复杂的设定程序即可连接宽带网络；网络付款机制，可简易、安全地通过网络支付内容和服务等的费用；即时连接，可随时在网络上安全、简易地连接不同规范的数字家电产品。UOPF 计划在 2 年内制定相关技术规范，进行兼容性测试，积极在海内外推广发展。

国内也成立了数字家庭标准化组织——闪联[8]，闪联全称是“信息设备资源共享协同服务标准工作组”（Intelligent Grouping and Resource Sharing，简称 IGRS），于 2003 年 7 月由联想、TCL、康佳、海信、长城 5 家企业发起、7 家单位共同参与正式成立，其目标是在多种信息设备、家用电器、通讯设备之间的设备自动发现、动态组网、共享资源和相互操作方面进行标准化工作[9]。闪联适用的范围是：企业、公共场所、个人以及家庭所涉及的信息设备互联时遵循共同资源及功能服务接口标准，使设备能够有效实现资源开放及服务协同，提高设备功能互操作性，并增强不同设备间组合服务的功能。

从以上分析可以看出，各自依托自身的技术实力和自身利益整合而成的标准化组织所推出的标准在本行业有较高的适用性和成熟度，都试图在其所在领域占领技术和市场的制高点。然而家庭网络从外部网络接入到家电网络终端，乃至各类家用设备涉及行业众多，各种技术的适用范围也不尽相同，从技术上看很难有一种标准或规范适用于家庭网络所有领域，各个标准化组织所提倡的“开放性标准体系”目标也就难于实现。

正是由于上述原因，各行业在不断竞争中正逐渐强化合作，很多企业都加入了多个标准化组织并提交技术方案，以保证自身利益在各行业的体现。数字家庭正朝着一种多技术融合的方向发展。

1.3 课题主要工作与内容安排

本课题以由 Intel 主导的、在家庭设备互连和媒体内容管理方面有显著优势的 DLNA 技术为研究对象，深入分析了 DLNA 技术进行设备互连和媒体管理的原理、过程和方法，重点研究并实现了 UPnP 设备架构和 UPnP 音频/视频架构，实现了 DLNA 协议。在实现 UPnP 设备架构的过程中，提出了一种基于阻塞/唤醒的单线程轮询机制，降低了对系统资源的依赖，增强了系统可移植性，提高了系统运行效率。并以此为基础，设计实现了一个 DLNA 控制点系统。课题的具体工作涉及以下几方面内容：

第一章：引言，包括课题的研究背景、数字家庭标准化发展现状等内容。

第二章：DLNA 技术分析，主要从基本概念、核心内容和体系结构方面对 DLNA 技术的工作原理和运行机制进行分析，重点研究 DLNA 技术中的 UPnP 设备架构和 UPnP Audio/Video 架构，掌握了 DLNA 技术实现设备互连和媒体管理的原理和方法。

第三章：DLNA 控制点系统的设计，包括控制点系统的架构设计和结构设计等内容。通过分析系统初步结构设计中存在的缺点和不足，对系统结构做了改进设计，提出了一种基于阻塞/唤醒的单线程轮询机制。

第四章：DLNA 控制点系统的实现，主要在设计实现基于阻塞/唤醒的单线程轮询机制的基础上，分别实现控制点系统的 UPnP 设备架构、UPnP Audio/Video 架构和用户界面三部分，其中对 UPnP 设备架构和 UPnP Audio/Video 架构的实现，就是对 DLNA 协议的实现。

第五章：DLNA 控制点系统的互连性测试，通过对控制点系统进行测试，验证对 DLNA 协议的实现程度和实现效果。

第六章：结束语，主要包括设计心得，工作成果以及不足之处等内容。

第二章 DLNA 技术分析

2.1 DLNA 概述

数字生活网络联盟（Digital Living Network Alliance，简称 DLNA）是一个从事数字家庭标准化的组织，它提出了一个远景目标，即在数字家庭媒体中心中建立一个集中管理个人电脑、家电和移动电子设备的互操作网络，创造一个能够共享和发展全新数字媒体和内容服务的无缝环境。

在 DLNA 远景目标中，数字家庭包含了大量不同类型的设备，它们可以存储、访问和播放不同格式的数字媒体，并且具有对整个家庭网络的设备进行控制的智能。高端设备负责协调其它网络设备的运行，例如，诸如个人电脑、高级机顶盒、数字电视或网络控制板等智能平台可以管理和发布来自各种设备的丰富的数字内容，如数字录像机（DVR）、网络 DVD 播放器、多媒体手机、电视机、立体声音响或无线扬声器及显示器等。

当数字家庭里的设备能够在为用户提供的某项服务上进行透明协作时，它们之间就实现了互操作，这包括设备之间可相互通信和交换有效信息的能力。要实现互操作所需要解决的问题主要有以下几方面：

(1) 媒体中心内部设备之间的透明连接。这包括在链路层实现设备直接相连所需的网络兼容性。当采用不同的链路层技术的设备需要通信时，这些设备之间必须配置适当的链路层桥接和网络层路由。总目标是实现所有设备之间端到端的连接，能够通过家庭网络交换信息。

(2) 设备发现、配置和控制方面的统一的框架。家庭网络上的任何设备都必须能够发现网络上其它设备和服务的存在，能够识别它们的功能和相关能力，并且能配置这些设备和服务，简单控制其运行。

(3) 可互操作的媒体管理和控制框架。需要有一个覆盖整个家庭网络的媒体管理和控制框架，能够使不同厂商提供的设备之间实现媒体信息的交换和控制，还必须具备对要处理的媒体内容进行组织、浏览、搜索和选择的能力。

(4) 可互操作的媒体格式和流协议。设备之间如果要实现相互通信，就需要达成一个通用流协议，以建立媒体数据流的传输。这些设备还需要就支持的媒体格式达成一致，以确保可以共享和使用媒体。

针对以上几个实现设备互操作所需解决的主要问题，DLNA 提出了一个家庭网络设备的互操作架构[1]，它能够实现家电设备、移动设备和电脑设备之间的简单无缝地互操作，增强和丰富用户的体验；它提供了在数字家庭中搭建可互操作网络平台和设备所需的必要信息。DLNA 互操作架构充分利用了已有的开发标准和技术，并在操作上加以限的必要信息。DLNA 互操作架构充分利用了已有的开发标准和技术，并在操作上加以限

第二章 DLNA 技术分析

2.1 DLNA 概述

数字生活网络联盟（Digital Living Network Alliance，简称 DLNA）是一个从事数字家庭标准化的组织，它提出了一个远景目标，即在数字家庭媒体中心中建立一个集中管理个人电脑、家电和移动电子设备的互操作网络，创造一个能够共享和发展全新数字媒体和内容服务的无缝环境。

在 DLNA 远景目标中，数字家庭包含了大量不同类型的设备，它们可以存储、访问和播放不同格式的数字媒体，并且具有对整个家庭网络的设备进行控制的智能。高端设备负责协调其它网络设备的运行，例如，诸如个人电脑、高级机顶盒、数字电视或网络控制板等智能平台可以管理和发布来自各种设备的丰富的数字内容，如数字录像机（DVR）、网络 DVD 播放器、多媒体手机、电视机、立体声音响或无线扬声器及显示器等。

当数字家庭里的设备能够在为用户提供的某项服务上进行透明协作时，它们之间就实现了互操作，这包括设备之间可相互通信和交换有效信息的能力。要实现互操作所需要解决的问题主要有以下几方面：

(1) 媒体中心内部设备之间的透明连接。这包括在链路层实现设备直接相连所需的网络兼容性。当采用不同的链路层技术的设备需要通信时，这些设备之间必须配置适当的链路层桥接和网络层路由。总目标是实现所有设备之间端到端的连接，能够通过家庭网络交换信息。

(2) 设备发现、配置和控制方面的统一的框架。家庭网络上的任何设备都必须能够发现网络上其它设备和服务的存在，能够识别它们的功能和相关能力，并且能配置这些设备和服务，简单控制其运行。

(3) 可互操作的媒体管理和控制框架。需要有一个覆盖整个家庭网络的媒体管理和控制框架，能够使不同厂商提供的设备之间实现媒体信息的交换和控制，还必须具备对要处理的媒体内容进行组织、浏览、搜索和选择的能力。

(4) 可互操作的媒体格式和流协议。设备之间如果要实现相互通信，就需要达成一个通用流协议，以建立媒体数据流的传输。这些设备还需要就支持的媒体格式达成一致，以确保可以共享和使用媒体。

针对以上几个实现设备互操作所需解决的主要问题，DLNA 提出了一个家庭网络设备的互操作架构[1]，它能够实现家电设备、移动设备和电脑设备之间的简单无缝地互操作，增强和丰富用户的体验；它提供了在数字家庭中搭建可互操作网络平台和设备所需的必要信息。DLNA 互操作架构充分利用了已有的开发标准和技术，并在操作上加以限

制，来实现数字家庭设备之间互操作。

DLNA 互操作架构主要包括物理连接、网络传输、设备发现与控制、媒体管理、媒体格式和媒体传输六个组成部分。表 2.1 列出了 DLNA 互操作架构中的关键功能组件以及每个组件对应的技术要素。

表 2.1 DLNA 关键技术要素

Table 2.1 DLNA key technology ingredients

功能组件	技术要素
物理连接	以太网，802.11，蓝牙
网络传输	IPv4 协议簇
设备发现与控制	UPnP 设备架构 Ver1.0
媒体管理	UPnP Audio/Video 架构 Ver1.0
媒体格式	必需和可选的格式（JPEG，MPEG 等）
媒体传输	HTTP（必需）和 RTP（可选）

DLNA 技术中的关键功能组件，均基于开放标准，例如互联网协议（IP）、HTTP、UPnP™和 Wi-Fi®协议，以促进其在家电、个人电脑和移动产品制造行业的应用。

下面详细分析 DLNA 互操作架构及其功能组件。

2.2 DLNA 互操作架构

为了实现家庭网络设备之间的相互操作、相互控制，DLNA 制定了家庭网络设备的互操作架构[1]，如图 2.1 所示。

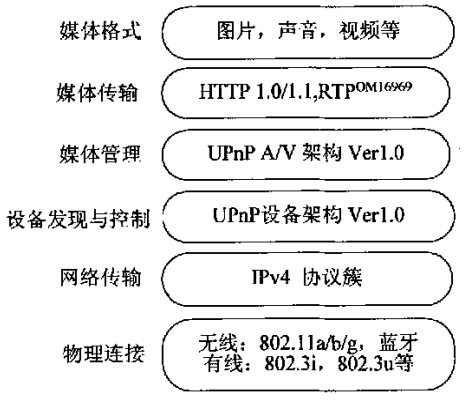

图 2.1 DLNA 互操作架构图

Fig. 2.1 DLNA interoperability architecture

DLNA 互操作架构定义了这些功能组件的运行机制，来保证家庭网络设备之间的互操作。下面分析 DLNA 互操作架构中的每一个功能组件以及各个组件对应的关键技术要素，重点分析实现设备发现与控制的 UPnP 设备架构和实现媒体管理的 UPnP Audio/Video 架构。

2.2.1 网络连接与传输

IP 协议是实现数字家庭中 DLNA 设备联网和连接的基础。IP 协议为互联网中的应用提供了网络通信支持。IP 协议基于行业标准规范之上，得到了众多设备的采用与支持。在数字家庭中使用 IP 协议优势有很多：

(1) IP 协议支持应用程序透明地运行在不同媒体之上，以实现无缝通信。例如，在媒体中心中，一个服务器设备可以通过以太网向其它房间中的电视传送媒体内容；借助 IP 协议，服务器设备和电视不会了解到媒体内容在两个独立的物理媒体中进行传送。IP 协议为 DLNA 设备之间的对等层通信提供了一个统一的框架，确保应用程序能够独立于实际传输技术之上。

(2) IP 协议可以将家中的任意设备连接到互联网。IP 协议是一种互联网协议，因此数字家庭中的任何设备都可以与其它连接到互联网的设备实现互联。

(3) IP 协议使用广泛且成本非常经济。IP 协议无处不在，规模经济和激烈竞争使得 IP 协议物理媒体的实施成本要远远低于其它技术。在家庭网络环境中，设备最常用的是以太网（802.3i 和 802.3u）和无线技术（802.11a，802.11b 和 802.11g）。在移动设备领域，比较通用的无线技术是蓝牙技术。

运行于 IP 网络上的多媒体应用受益于服务质量机制。如果没有服务质量机制，运行于不同设备上的所有应用程序具有相等的媒体传输机会，而多媒体应用（比如视频流和音频流）对声音或图像的延迟以及系统吞吐量要求比较严格。使用具有优先级的服务质量，应用程序就可以按照优先级来获得网络资源。DLNA 服务质量模型通过使用优先级，促进了媒体传输的连贯性和一致性，提高了整个 DLNA 网络的性能，增强了设备之间的互操作性。

2.2.2 设备发现与控制

通用即插即用[10]（Universal Plug and Play，简称 UPnP）是由 UPnP 论坛提出并维护的一个充分利用 TCP/IP 和 Web 技术的分布式网络体系结构。UPnP 能够在家庭、办公室和公共场所的不同类型的设备之间建立无缝的连接网络，并且进行简单设备控制和数据传输。

UPnP 设备架构不仅仅是即插即用外设模式地简单扩展，它简化了家庭和企业中智能设备的联网过程。UPnP 设备架构设计用于支持零配置、不可见的联网，以及对各种不同类型的设备进行自动地发现。这就意味着，一台设备能够动态加入一个网络，通报其功能，自动发现并了解其它设备的存在和功能。最后，设备能够顺利地自动离线，而不会造成任何其它的影响。

UPnP 不使用设备驱动程序，取而代之的是使用通用协议。UPnP 充分利用了包括 IP、TCP、UDP、HTTP 和 XML 在内的互联网组件。UPnP 数据包规范基于 IP 协议，以 XML

来表达，通过 HTTP 进行传输。此外，当成本、技术或传统因素等阻止与 UPnP 连接的媒体或设备运行 IP 协议时，UPnP 还可通过桥接方式支持运行非 IP 协议的媒体。

图 2.2 列出了 UPnP 设备架构中定义的控制点和设备之间的通信协议栈[11]。在设备发现与控制的不同阶段，采用不同的协议。

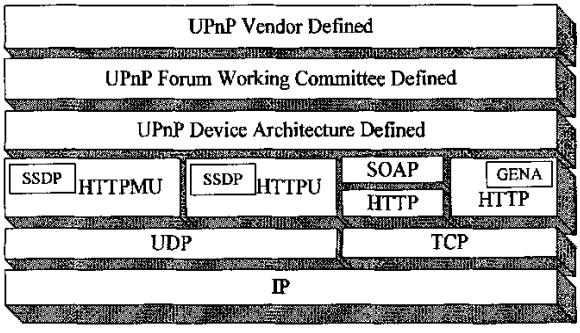

图 2.2 UPnP 协议栈

Fig. 2.2 UPnP protocol stack

要发送的 UPnP 数据消息经过处理后，通过采用简单服务发现协议（Simple Service Discovery Protocol，简称 SSDP）、通用事件通知架构（General Event Notification Architecture，简称 GENA）和简单对象访问协议（Simple Object Access Protocol，简称 SOAP）来进行格式化，然后消息通过运行于 UDP 协议上的 HTTP 多播或单播，或者运行于 TCP 协议上的标准 HTTP 进行传输，最终所有消息均通过 IP 协议进行传输。

UPnP 网络不依赖于任合媒体。UPnP 设备可以在任何操作系统上采用任何编程语言来实现。UPnP 设备架构是与操作系统和编程语言无关的。UPnP 并未针对运行于 UPnP 设备上的应用而指明或限制 API 的设计。UPnP 通过使用浏览器和传统应用程序控制来使厂商能够控制设备的用户界面并实现交互。

UPnP 设备架构定义了媒体中心中的两种通用的设备：控制点（Control Point）和被控制设备（以下简称设备）。设备的职能类似于一个服务器，它根据自己的功能向媒体中心提供一系列服务，每个服务定义了一些动作和状态变量。控制点通过调用设备提供的服务来获得设备的状态和控制设备的运行。

UPnP 设备和控制点都可以在各种不同的平台上实现，包括 PC 机或嵌入式系统。设备和控制点也可以在同一个设备上，并且多个设备或控制点可以相互嵌套。

设备发现与控制的过程中，一个 UPnP 设备（设备或控制点）的运行和交互过程如下：

(1) 寻址：设备通过寻址来获得一个网络地址。

(2) 发现：设备向控制点宣告自己加入网络；控制点发现自己感兴趣的设备。

(3) 描述：控制点获得设备的功能、属性等信息的详细描述。

(4) 控制：控制点向设备发送控制消息；设备向控制点返回控制的响应消息。

(5) 事件：控制点监听设备状态变化；设备状态发生变化时通知控制点。

(6) 展示：控制点显示设备的一个用户界面。

其中控制、事件和展示过程相互独立，相互补充。UPnP 设备发现与控制的主要过程如图 2.3 所示。

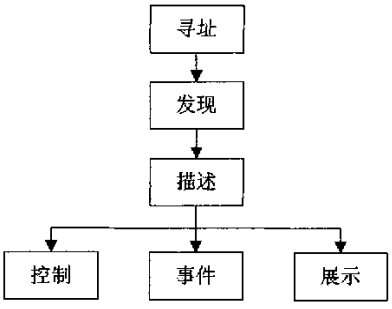

图 2.3 UPnP 设备发现与控制过程

Fig. 2.3 UPnP device discovery and control process

下面详细叙述设备发现与控制中每个步骤中的详细机制。

2.2.2.1 寻址

寻址是 UPnP 设备架构的基础。每一个设备（无论是控制点还是被控设备）第一次加入家庭网络时，都必须首先获得一个 IP 地址。

每一个设备都必须配有动态主机配置协议（DHCP）客户端，并在设备首次与网络连接时搜索 DHCP 服务器。如果有 DHCP 服务器可以使用，即网络处于管理状态，则设备必须采用 DHCP 服务器分配给它的 IP 地址。如果没有 DHCP 服务器可用，即网络处于未管理状态，则设备必须利用 Auto-IP 协议来获取一个地址。

Auto-IP 协议负责管理一台设备如何从一组保留地址中智能地选出一个未使用的 IP 地址，以及如何能够在处于管理和未管理状态的网络间轻松切换。

使用 Auto-IP 协议自动配置 IP 地址的设备必须进行定期检查，以确定是否有 DHCP 服务器存在。一旦发现有 DHCP 服务器存在，则设备进行动态地址分配，释放原先自动配置的地址，使用新的地址加入网络。

如果设备在与 DHCP 服务器通信过程中获得了一个域名（例如通过一台 DNS 服务器或通过 DNS 转发），则设备应当在后来的网络互操作中使用该名称；否则就采用其 IP 地址。

2.2.2.2 发现

设备获得了网络地址，就称该设备加入了网络。发现是 UPnP 设备架构中的第一步。

当设备加入网络之后，UPnP 发现协议允许该设备向网络中的控制点宣告其服务。同样，当一个控制点加入网络之后，UPnP 发现协议允许该控制点搜索网络上感兴趣的设备。两种情况下的基本交换信息均为一种发现消息，发现消息中包含关于该设备或其服务之一的少量基础性的细节信息，例如设备或服务类型、标识符和指向更详细信息的

一个指针。

当一个新的设备加入网络之后，它会多播发现消息来宣告设备本身及其嵌入式设备和服务。所有感兴趣的控制点必须监听标准的多播地址，以发现新的可用设备和服务。宣告消息中包括直到宣告过期的持续时间。如果设备仍然可用，则应该定期将宣告消息重新发送（带有新的持续时间）。如果设备要退出网络，则设备应当明确撤消其宣告。如果设备本身不能撤消，则宣告将自动过期。

当一个新的控制点加入网络之后，它会多播发现消息来搜索感兴趣的设备和服务。所有设备必须监听这些消息的标准多播地址，同时必须在其嵌入式设备或服务与控制点的发现消息中的搜索标准相符合时做出响应。

控制点发现感兴趣的设备可能是由于设备发送了宣告自己的发现消息，也可能由于设备对一条搜索设备的发现消息做出了响应。在任何一种情况下，如果控制点对一个设备感兴趣并想进一步了解关于该设备的更多信息，则控制点必须利用发现消息中的信息来发送一条描述询问消息。在描述部分将对此做详细说明。

图 2.4 详细描述了设备的发现过程。

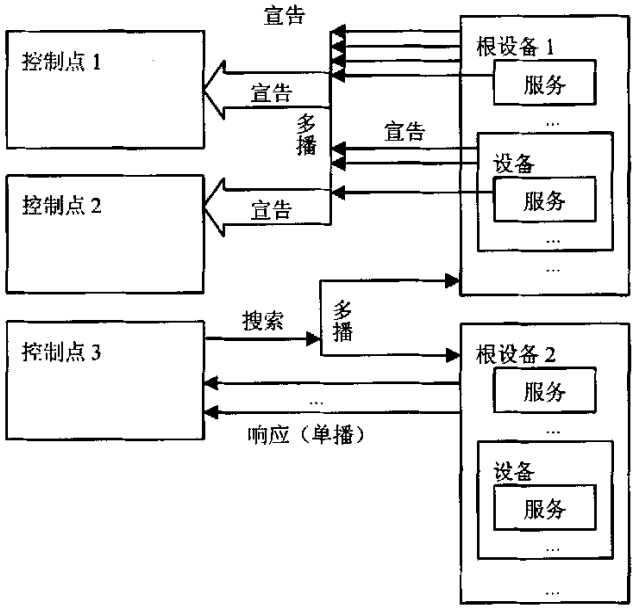

图 2.4 设备发现示意图

Fig. 2.4 Device discovery illustration

当设备离线后，它应当多播大量的发现消息来撤消其早期宣告，明确地宣布其嵌入式设备和服务不再可用。

在设备发现过程中，设备的宣告消息、控制点的搜索消息以及设备的响应消息，都是由简单服务发现协议（SSDP）定义的。其中，设备宣告消息和控制点搜索消息都是由运行于 UDP 协议上的 HTTP 多播来传输的，设备的响应消息是由运行于 UDP 协议上的 HTTP 单播来传输的。

在设备发现过程中，需要选择一个适当的宣告有效时间，使得能够在最小化网络流量和最大化设备状态及时更新之间求得一个平衡。相对较短的有效时间可以保证控制点在牺牲额外网络流量的情况下获得设备的当前状态；相对较长的有效时间会损失设备状态的刷新速度，但是可以大大减少网络流量。一般来讲，宣告有效时间的设定原则是：希望短期内成为网络组成部分的设备有效时间短；希望成为网络长期成员的设备有效时间长。

2.2.2.3 描述

在控制点发现了一个设备之后，控制点仍然对设备知之甚少。控制点只能了解发现消息中的相关信息，如设备（或服务）的类型、设备的全球唯一标识符和设备 UPnP 描述的 URL 地址等。为了能让控制点更多地了解设备及其功能，进而与设备进行交互，控制点必须从发现消息中得到设备描述的 URL，并通过此 URL 取得设备描述。

对于一个设备的 UPnP 描述一般分成两个逻辑部分：设备描述（描述所包含的物理与逻辑设备）以及一个或多个服务描述（描述设备对外提供的能力）。UPnP 设备描述包括特定厂商、制造商信息、模块名称和编号、序列号、制造商名称、特定厂商网站 URL 等。对于设备中的每种服务，设备描述都包含服务类型、名称、服务描述 URL、控制 URL 以及事件 URL。设备描述还包含所有嵌入式设备描述与对应的 URL 地址集。控制点在获得 UPnP 设备的设备描述之后，就可以使用服务描述 URL 获得 UPnP 设备的服务描述。在控制、事件和展示部分，将详细介绍如何使用控制 URL、事件 URL 和展示 URL。

在家庭网络中，一个物理设备可以包含多个逻辑设备。多个逻辑设备既可以被看作是一个包含嵌入式设备的根设备，也可以被看作是多个根设备（可能没有嵌入式设备）。前者是一个有关根设备的 UPnP 设备描述，其中包含所有嵌入式设备描述。后者是多个 UPnP 设备描述，每个根设备对应一种描述。

UPnP 服务描述包括一系列命令或动作、服务的响应以及每种动作的参数。服务描述还包括一系列变量。这些变量描述了服务运行时的状态，其中包括数据类型、取值范围和事件特性的描述。

UPnP 设备描述和服务描述由 UPnP 厂商编写，采用 XML 句法，并且遵循标准 UPnP 设备模板和服务模版。UPnP 设备模板和服务模板由 UPnP 论坛工作委员会生成，来源于 UPnP 模板语言。UPnP 模板语言来源于 XML 标准结构。

取得 UPnP 设备描述和服务描述的过程非常简单。控制点首先从发现消息中得到设备描述的 URL，通过这个 URL 就可以获得 UPnP 设备的设备描述。然后，控制点可以通过设备描述中提供的 URL 取得一个或多个服务描述。这是一个基于 HTTP 的简单流程。

图 2.5 详细描述了设备的描述过程。

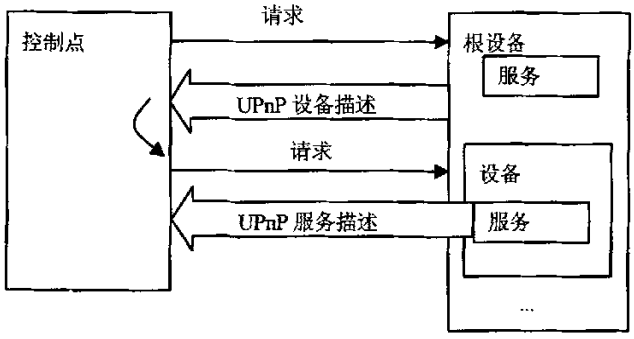

图 2.5 设备的描述过程

Fig. 2.5 Device description illustration

2.2.2.4 控制

在获得 UPnP 设备的设备描述和服务描述之后，控制点可以向这些服务发出控制动作请求，同时控制点也可以查询服务的状态变量值。发出控制动作请求实质上是一种远程过程调用。控制点将控制动作请求发送到设备服务，在动作执行完毕后，设备服务返回执行的结果。

为了控制一个设备，控制点向该设备的服务发出一个控制动作，这是通过控制点向 UPnP 设备的服务控制 URL 发送一个对应的控制消息来实现的。设备服务接收到控制消息后会根据控制动作的内容执行相应的动作，提供对应的服务，在执行完请求的动作之后返回运行结果。动作的执行可能会引起 UPnP 设备服务运行状态的变化，使服务的状态变量发生变化，这种变化将以事件通知的形式发布到所有相关的控制点。在事件部分将详细介绍事件通知过程。

图 2.6 详细描述了设备的控制过程。

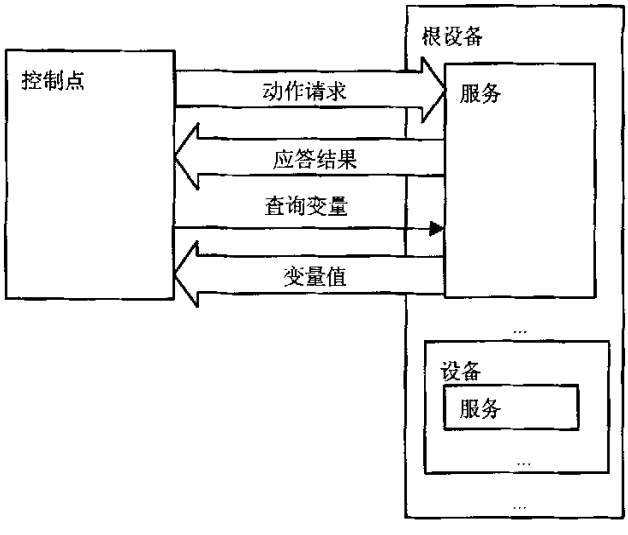

图 2.6 设备的控制过程

Fig. 2.6 Device control illustration

控制点可以查询服务的状态变量值以获得状态变量的当前值。与发出一个动作的过

程相似，控制点向服务的控制 URL 发送一个适当的查询消息，服务则返回相应的变量值。每个服务必须保持状态表的一致性，以便控制点能够查询并接收到有意义的值。每个查询消息只能查询一个状态变量，控制点必须发送多个查询消息来查询多个状态变量。

只要设备和服务的发现宣告没有过期，控制点就会假定该设备及其服务可用。如果设备取消其宣告，或者设备和服务的发现宣告过期，则控制点认为该设备及其服务不再可用。

设备控制过程中的控制动作请求消息和设备服务的应答消息的格式都是由简单对象访问协议（SOAP）来定义，并使用 HTTP、TCP/IP 协议进行传输。

2.2.2.5 **事件**

UPnP 设备的服务描述包括服务提供的动作列表和运行时描述服务状态的变量列表。在设备运行的过程中，如果设备服务的一个或多个状态变量发生了变化，服务将会在这些变量发生变化时发布更新。控制点可以订阅设备服务以获得这些信息。

要订阅设备服务的事件通知，控制点首先从该设备的设备描述中获得要订阅服务的服务标识符和事件 URL，然后向此 URL 发送一个包含事件通知消息交付 URL 的订阅消息。订阅消息中的事件通知消息交付 URL 指明了控制点接收设备服务事件通知消息的地址。订阅消息是一个接收所有事件消息的请求。UPnP 设备架构不提供在变量基础上订阅事件消息的机制。控制点将接收来自设备服务的所有事件通知消息。

如果设备服务接受控制点的事件订阅，它将向控制点返回唯一的订阅标识符和订阅持续时间，并立即发送一个初始化事件通知消息，事件通知消息以 XML 描述，包含服务的所有状态变量的名称和值，使得控制点可以初始化设备服务状态模型。

在设备服务状态发生变化时，就会向订阅该服务的控制点事件通知消息交付 URL 发送事件通知消息。每个事件消息都标有事件编号，以便进行错误检查。当设备服务发送初始化事件消息时，事件编号被初始化为 0。对于以后每次发送事件通知消息，设备服务会增加事件编号的数值。控制点接收到事件通知消息后，首先判断该事件通知消息的编号是不是前一个编号的增量，如果是，则说明接收正确；如果不是，则说明控制点在接收设备服务事件通知消息的过程中丢失了数据包，需要先退出网络，然后重新登录。

为了保持订阅，控制点必须在事件订阅过期之前通过发送续订消息进行续订。续订消息被发送到与订阅消息相同的 URL，但续订消息不包含事件消息的交付 URL，而是包含设备服务为控制点分配的唯一订阅标识符。

如果订阅过期，控制点的唯一订阅标识符将失效，设备服务将停止向控制点发送事件通知消息，并将过期的控制点从其订阅者列表中清除。如果控制点试图发送订阅消息之外的任何消息，设备服务将会以该订阅标识符无效为由而拒收此消息。

当控制点不再需要了解某一设备服务的运行状态时，应该撤消其订阅。撤销订阅可以减少网络负载。如果一个控制点突然从网络上退出，它可能无法发送撤销消息，则该控制点的订阅最终将会过期而自动取消。

控制点应该时刻注意来自设备和服务的宣告消息。如果设备和服务发出了离线宣告，则控制点认为其事件订阅已经被取消。

图 2.7 详细描述了设备的事件订阅与事件通知的全过程。

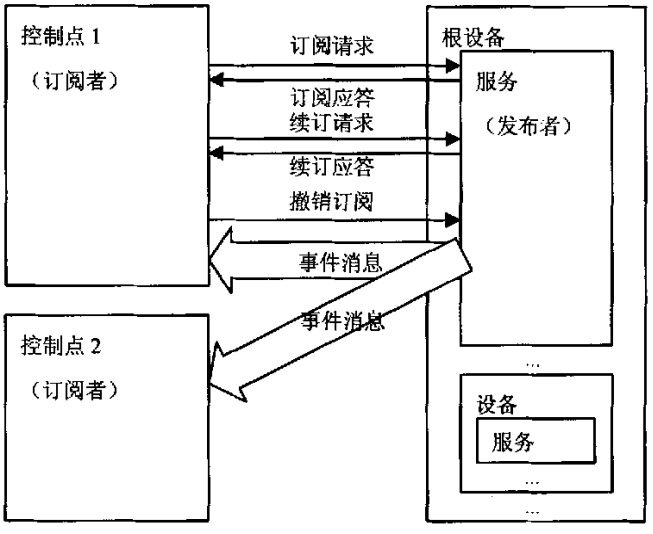

图 2.7 设备的事件订阅与事件通知过程

Fig. 2.7 Device event subscribe and event notify illustration

设备事件过程中的所有消息都是由通用事件通知架构（GENA）定义的，并使用 HTTP、TCP/IP 协议进行传输。

2.2.2.6 展示

如果 UPnP 设备拥有进行展示的 URL，那么控制点就可以通过此 URL 取得一个页面，在浏览器中加载该页面，并根据页面功能，支持用户控制设备和浏览设备状态。展示 URL 包含在 UPnP 设备的设备描述中。每一项完成的程度取决于展示页面和设备的具体功能。

图 2.8 描述了设备的展示过程。

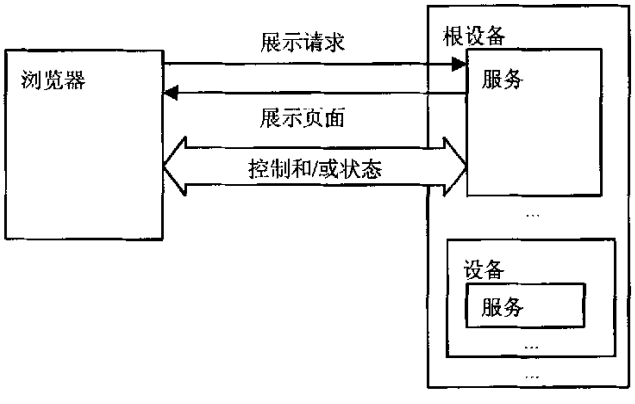

图 2.8 设备的展示过程

Fig. 2.8 Device presentation illustration

综上所述，UPnP 设备架构主要通过寻址、发现、描述、控制、事件和展示六个步

骤来实现网络设备的发现与控制过程。UPnP 设备架构支持 DLNA 家庭网络中的设备自动配置网络属性，发现网络中其它设备的存在与功能特性，并以一种连续统一的方式控制与这些设备进行交互，很好地满足了用户希望简化家庭设备联网的需求。UPnP 设备架构是 DLNA 数字家庭中互操作网络所要求的主要标准。

2.2.3 媒体管理

DLNA 互操作架构采用 UPnP Audio/Video 架构[12]来管理家庭网络上各式各样的媒体内容信息。UPnP Audio/Video 架构支持设备与应用在家庭网络中识别、管理和发布媒体内容，很好地满足了 DLNA 互操作架构中管理与控制媒体的需要。

UPnP Audio/Video 架构定义了 UPnP 媒体设备与相关的控制点应用之间的交互模式。UPnP 媒体设备包括电视、录像机、CD/DVD 播放机、机顶盒、音响系统、静态照相机、电子相框和计算机等。UPnP Audio/Video 架构可以支持设备之间采用任意格式和通过任意传输协议传送娱乐内容。

UPnP Audio/Video 架构对 UPnP 设备的类型做了进一步的划分，它定义了三种逻辑设备：媒体服务器，媒体渲染器和控制点。在家庭网络中每种设备都提供不同的服务，供其它 UPnP 设备使用。媒体服务器上存储娱乐内容，并可以将这些内容通过家庭网络传送到其它 UPnP 设备上。媒体渲染器可以接收家庭网络上的娱乐内容并在本地播放。控制点用来协调媒体服务器和媒体渲染器的运行，来实现对媒体服务器和媒体渲染器的控制。

UPnP Audio/Video 架构如图 2.9 所示。

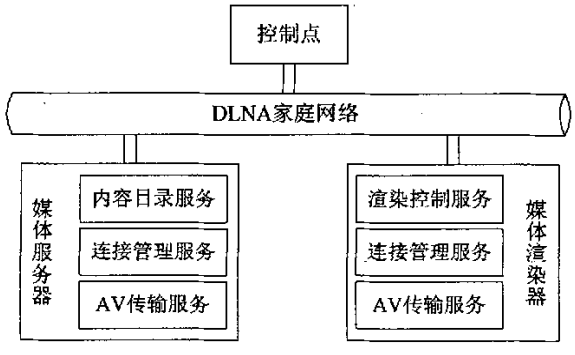

图 2.9 UPnP Audio/Video 架构

Fig. 2.9 UPnP Audio/Video architecture

2.2.3.1 媒体服务器

媒体服务器上储存娱乐内容（如电影，音乐，图片等），它可以通过家庭网络将这些娱乐内容传输到其它设备上进行播放。媒体服务器主要提供三个服务：

(1) 内容目录服务：列出了可用的媒体内容属性，包括名字，作者，创建日期，文件大小等等；还提供各个媒体文件所支持的传输协议和数据格式。

(2) 连接管理服务：协调并选择合适的传输协议和媒体格式，实现媒体内容从媒体

服务器向媒体渲染器的传输。

(3) AV 传输服务：控制媒体内容的传输流程（播放、停止、暂停和搜索等）。

其中内容目录服务和连接管理服务是媒体服务器必须提供的；AV 传输服务是可选的，它取决于媒体内容传输时所采用的传输协议。当媒体内容是采用的面向 push 的传输协议（如 IEEE-1394）时，媒体服务器必须提供 AV 传输服务。

2.2.3.2 媒体渲染器

媒体渲染器具有渲染媒体内容的能力，它能够从家庭网络上的媒体服务器接收媒体内容并进行渲染。媒体渲染器主要提供三个服务：

(1) 渲染控制服务：控制媒体内容的渲染形式（如音量大小、静音开关、亮度和对比度等等）。

(2) 连接管理服务：协调并选择合适的传输协议和媒体格式，实现媒体内容从媒体服务器到媒体渲染器的传输。

(3) AV 传输服务：控制媒体内容的传输流程（播放、停止、暂停和搜索等等）。

其中渲染控制服务和连接管理服务是媒体渲染器必须提供的，AV 传输服务是可选的，它取决于媒体内容传输时所采用的传输协议。当媒体内容是采用的面向 pull 的传输协议（如 HTTP-GET）时，媒体渲染器必须提供 AV 传输服务。

2.2.3.3 控制点

控制点通过调用媒体服务器和媒体渲染器提供的服务，来协调并控制媒体内容从媒体服务器向媒体渲染器的传输，管理媒体内容的渲染流程和渲染形式。这一操作的详细过程如下：

(1) 控制点上线后，首先使用简单服务发现协议（SSDP）定位网络上所有的媒体服务器和媒体渲染器。控制点多播一条 SSDP 设备发现消息，来搜索网络上所有符合 UPnP 媒体服务器规范和 UPnP 媒体渲染器规范的设备。所有符合条件的设备必须对控制点的发现消息做出应答，并在应答消息中包含该设备的设备描述 URL 地址。

(2) 控制点发现网络上的媒体服务器和媒体渲染器之后，分别获得各个设备的设备描述并进行解析，以确定各个设备的实际功能（如提供哪些服务，包含哪些动作等）。如果控制点对某一个设备感兴趣，那么控制点将继续与该设备进行交互。

(3) 对于发现的每一个媒体服务器，控制点都会调用其内容目录服务，获得媒体服务器提供的可用的媒体内容，并将从各个媒体服务器收集到的所有可用媒体内容集中在一起，提供给用户使用，而不考虑各个媒体内容是由哪个媒体服务器提供的。

(4) 当用户指定要渲染的媒体内容时，控制点首先调用媒体服务器的内容目录服务来获得该媒体内容支持的传输协议和数据格式，然后调用各个媒体渲染器的连接管理服务，获得各个媒体渲染器支持的传输协议和文件格式，最后控制点通过比较这些传输协议和文件格式，来选择一个能够渲染指定媒体内容的媒体渲染器。

(5) 当要使用的传输协议和文件格式确定之后，控制点调用媒体服务器和媒体渲染

器的连接管理服务，使用选定的传输协议和文件格式建立和配置内部网络连接和媒体流传输子系统。

(6) 当媒体服务器和媒体渲染器配置好之后，根据采用的媒体内容传输协议，媒体服务器或者媒体渲染器会向控制点返回一个 AV 传输服务的实例。控制点使用这个实例来指定要从媒体服务器向媒体渲染器传输的媒体内容。

(7) 当用户在选定的媒体内容上执行了操作（如播放）时，就调用了 AV 传输服务中相应的动作，媒体内容开始传输并渲染。用户也可以选择其它操作如停止、暂停等等。

(8) 在渲染媒体内容的过程中，用户可以通过控制点调节媒体内容的渲染方式。控制点是通过调用媒体渲染器的渲染控制服务来实现对渲染方式的控制的。

控制点通过调用媒体服务器和媒体渲染器提供的各种服务，来实现媒体的控制和管理。在媒体渲染过程中，控制点不参与媒体内容从媒体服务器到媒体渲染器的传输。

综上所述，UPnP Audio/Video 架构很好地满足了家庭网络对媒体内容发布、管理和控制的需求，是 DLNA 数字家庭媒体管理与控制的主要标准。

2.2.4 媒体格式

媒体格式描述了在家庭网络中传输和渲染的媒体内容是如何被编码和格式化的。DLNA 媒体格式模型用于为网络互操作提供一个基准，并分别为不同类型的设备定义了三种媒体（图像、音频和视频）必需的媒体格式集合和可选的媒体格式集合。表 2.2 规定了各种媒体的必需格式和可选格式。

表 2.2 DLNA 媒体格式

Table 2.2 DLNA media format

媒体类型	必需格式集	可选格式集
图像	JPEG	PNG，GIF，TIFF
音频	LPCM	AAC、AC-3、ATRAC3plus、MP3、WMA9
视频	MPEG2	MPEG1、MPEG4、WMV9

为了支持家庭网络上不同种类设备之间的互操作，媒体互操作单元对不同种类设备所必需的媒体格式之间执行了基本的转化。并且，DLNA 媒体模型制定了从可选格式向必需格式转化时的规则，从而保证了媒体可以在所有设备上使用。

2.2.5 媒体传输

媒体传输定义了媒体内容是如何在家庭网络中传输的。在家庭网络中存储媒体的媒体服务器设备和渲染媒体的媒体渲染器设备，都必须支持 HTTP 协议，将其作为媒体传输的根本机制。此外，实时传输协议（Real Time Protocol，简称 RTP）也可以被用做媒体传输，该协议是可选的。

2.3 本章小结

本章对 DLNA 技术做了详细地分析。DLNA 是数字生活网络联盟的简称，该联盟为了能够建立一个集中管理个人电脑、家电设备和移动设备的无缝互操作网络，提出了一个家庭网络设备的互操作架构。

DLNA 互操作架构由物理连接、网络传输、设备发现与控制、媒体管理、媒体格式和媒体传输六大功能组件构成。本章系统介绍了 DLNA 互操作架构的各个功能组件以及每个功能组件使用的关键技术要素，重点介绍了其中的 UPnP 设备架构和 UPnP Audio/Video 架构。

UPnP 设备架构和 UPnP Audio/Video 架构是 DLNA 互操作架构实现设备发现与控制和媒体管理的重要基础。本章详细分析了 UPnP 设备架构通过寻址、发现、描述、控制、事件和展示来实现设备发现与控制的原理和机制，详细分析了 UPnP Audio/Video 架构中控制点通过调用媒体服务器和媒体渲染器提供的服务来实现媒体管理的过程，从而理清了 DLNA 技术实现家电设备互连和媒体内容管理的原理和机制。

第三章 DLNA 控制点系统的设计

3.1 设计目标

要设计一个支持 DLNA 协议的控制点系统，主要有两方面功能需要考虑：

(1) 在设备发现与控制方面。控制点系统启动后无须配置即可加入网络，能够自动发现网络上的设备，能迅速发现加入网络的新设备和退出网络的旧设备，能够了解各个设备的类型、所提供的服务，能够根据需要与设备通信，通过调用设备提供的服务控制设备的运行，能够随时掌握网络中设备的运行状态。这些目标可以通过实现 UPnP 设备架构来完成。

(2) 在媒体内容管理方面。控制点能够获得网络上媒体服务器提供的媒体内容信息，能够了解媒体渲染器的播放能力，能够了解媒体服务器和媒体渲染器支持的传输协议和文件格式，能够协调使得媒体内容从媒体服务器传输到媒体渲染器上进行渲染，并且能够控制媒体的渲染流程和渲染方式。这些目标可以通过实现 UPnP Audio/Video 架构来完成。

此外，在 DLNA 控制点系统的整个设计过程中，还有以下几个问题考虑：

(1) 系统可移植性

UPnP 设备架构和 UPnP Audio/Video 架构与编程语言和底层操作系统无关，在任何系统平台上使用任何语言都可以实现。因此，在设计 DLNA 控制点系统的过程中，应该充分利用 DLNA 技术的这些特性，以增强 DLNA 控制点系统的可移植性。

(2) 系统稳定性和运行效率

DLNA 控制点系统运行在家庭网络设备上，是家庭网络中所有设备的控制中心，因此控制点系统对于稳定性具有很高的要求。同时作为一种家电设备，控制点必须具备高效的数据处理能力和足够快的运行效率。

另外，由于控制点系统运行在操作系统之上，而目前的绝大多数操作系统都支持 DHCP、ARP 和 Auto-IP 协议。因此在设计控制点系统的过程中，不对设备寻址进行设计，以减轻控制点系统的复杂度。对设备展示也将不进行设计，取而代之的是设计一个专门的用户界面，实现与用户的交互。

3.2 系统设计思想

3.2.1 UPnP 设备架构的实现思想

在 TCP/IP 网络应用中，通信的两个进程之间相互作用的主要模式是客户/服务器模式[13]，即客户向服务器发出服务请求，服务器接收到请求后，提供相应的服务。客户/

服务器模式的建立基于以下两点：

(1) 网络中软/硬件资源、运算能力和信息不均等，需要共享，从而造就拥有众多资源的主机提供服务，资源较少的客户请求服务这一非对等作用。

(2) 网络中进程的通信完全是异步的，相互通信的进程之间既不存在父子关系，又不共享内存缓冲区，因此需要一种机制为希望通信的进程间建立联系，为二者的数据交换提供同步，这就是基于客户/服务器模式的 TCP/IP。

客户/服务器模式的通信过程如图 3.1 所示。

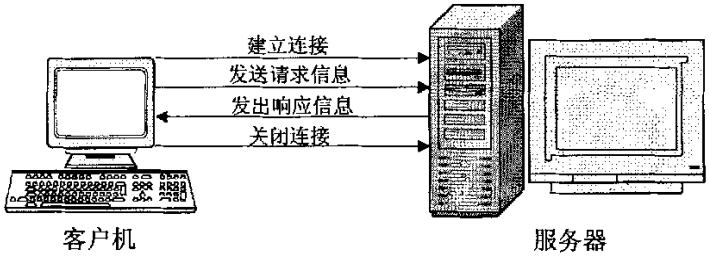

图 3.1 客户/服务器模式通信过程

Fig. 3.1 The communication of client-server model

在 DLNA 网络上，控制点与设备之间的通信符合上面分析的情形。家庭网络中，各个设备所拥有的资源和提供的服务都各不相同；设备功能有很大的差异，有的设备提供各种媒体内容信息，有的设备提供播放这些媒体内容的能力。控制点和其它设备的运行完全是异步的，它们之间无法共享内存，所以可以采用客户/服务器模式来实现控制点与 DLNA 设备之间的相互通信。

在控制点与设备进行通信的过程中，“客户”与“服务器”是一个相对的概念，控制点是作为客户端还是作为服务器，取决于它在设备发现与控制过程中所要实现的具体功能。分析 UPnP 设备架构中的设备发现与控制过程，有以下两种不同的设备通信方式：

(1) 在控制点控制 DLNA 设备的过程中，控制点根据用户需要向设备发送控制消息，调用设备提供的服务来实现对其的控制。在这一过程中，控制点和该设备构成了客户/服务器通信模型，控制点作为客户端首先向作为服务器的设备发送连接请求，在连接建立之后，控制点向设备发送控制请求消息，设备收到后执行相应的处理操作，并在处理完成后向控制点返回响应消息。

(2) 在 DLNA 设备的运行过程中，当设备状态发生变化时，就向控制点发送事件通知。通过这种方式，控制点可以随时掌握设备的运行状态。在设备向控制点发送事件通知的过程中，设备和控制点构成了客户/服务器通信模型，设备作为客户端向作为服务器的控制点发送连接请求，在连接建立之后，设备向控制点发送事件通知消息，其中包含了设备的最新运行状态，控制点收到事件通知后即了解到设备当前的运行状态，并向设备返回响应消息。

从上面的分析可以得出，控制点系统中的 UPnP 设备架构的实现，既需要包含一个客户端，来向其它设备发送控制消息以控制设备的运行，又需要包含一个服务器，来接收其它设备的事件通知消息以掌握设备的运行状态。

家庭网络中，控制点系统与其它 DLNA 设备之间的客户/服务器通信模型如图 3.2 所示。

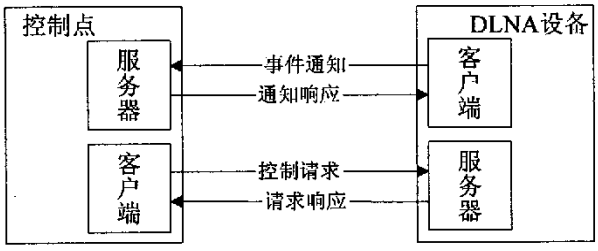

图 3.2 DLNA 网络中客户/服务器通信模型

Fig. 3.2 The communication of client-server model in DLNA network

3.2.2 UPnP Audio/Video 架构的实现思想

UPnP Audio/Video 架构建立在 UPnP 设备架构之上，充分利用了 UPnP 设备架构提供的设备和服务发现机制以及控制设备运行的功能，负责家庭网络上媒体内容的识别、发布和管理。UPnP Audio/Video 架构的实现相对简单。

UPnP Audio/Video 架构将 DLNA 网络上的设备分为：媒体服务器、媒体渲染器和控制点。媒体服务器上储存并管理一些媒体内容，这些媒体内容可以通过 DLNA 网络进行传输。媒体服务器主要提供内容目录服务、连接管理服务和 AV 传输服务，其中内容目录服务和连接管理服务是媒体服务器必须提供的，内容目录服务可以列出媒体服务器包含的媒体内容信息，连接管理服务协调媒体内容向其它设备的传输。AV 传输服务是可选的，它取决于媒体内容传输时所采用的传输协议。当媒体内容是采用的面向 push 的传输协议时，媒体服务器需要提供 AV 传输服务，来管理媒体内容的传输过程。

媒体服务器[22]的结构如图 3.3 所示。

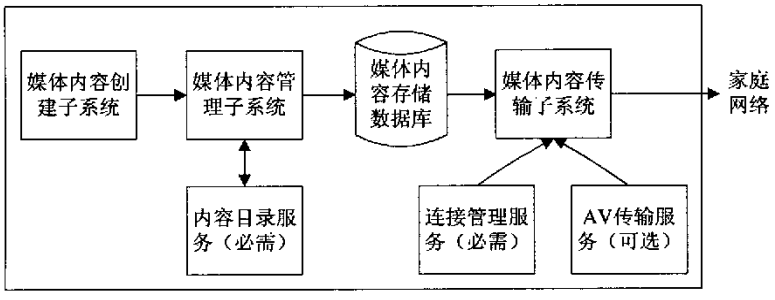

图 3.3 媒体服务器结构图

Fig. 3.3 Media server chart

媒体渲染器提供媒体的渲染能力，如播放、收听等，并且媒体渲染器支持渲染从家

庭网络上获得的媒体内容。媒体渲染器主要提供渲染控制服务、连接管理服务和 AV 传输服务，其中渲染控制服务和连接管理服务是媒体渲染器必须提供的，渲染控制服务用来控制媒体内容的渲染形式，连接管理服务用来协调从其它设备接收媒体内容传输。当媒体内容是采用的面向 pull 的传输协议时，媒体渲染器需要提供 AV 传输服务。

媒体渲染器[23]的结构如图 3.4 所示。

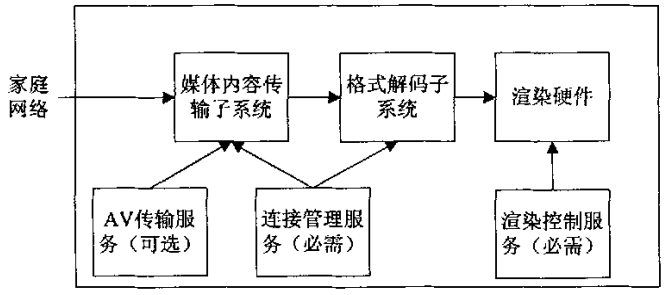

图 3.4 媒体渲染器结构图

Fig. 3.4 Media renderer chart

控制点的主要任务是通过调用媒体服务器的内容目录服务来获得可用的媒体内容信息，通过调用媒体服务器和媒体渲染器的连接管理服务，来协商选择合适的传输协议和媒体格式进行媒体内容传输，通过调用 AV 传输服务来控制媒体渲染流程，通过调用媒体渲染器的渲染控制服务来控制渲染方式。

UPnP Audio/Video 架构的实现的主要思想，就是分别实现媒体服务器和媒体渲染器所需提供的服务，实现控制点对这些服务的调用。

本课题只考虑控制点系统中 UPnP Audio/Video 架构的实现，因此 UPnP Audio/Video 架构实现的主要思想是通过 UPnP 设备架构提供的各种机制，调用媒体服务器和媒体渲染器提供的服务，维护网络上的媒体信息，管理网络上的媒体渲染器，实现媒体渲染，同时为上层提供接口。

综合上面的分析，控制点系统中 UPnP Audio/Video 架构模型如图 3.5 所示。

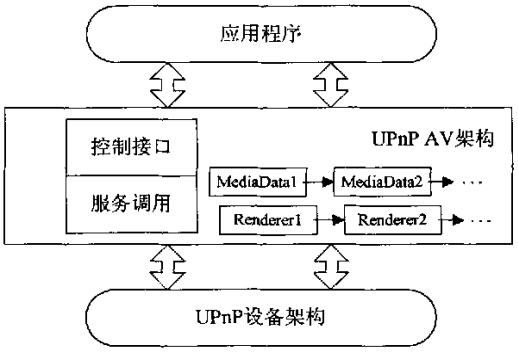

图 3.5 UPnP Audio/Video 架构模型

Fig. 3.5 UPnP Audio/Video architecture model

3.3 系统架构设计

根据上面的分析，控制点系统的架构设计如图 3.6 所示。

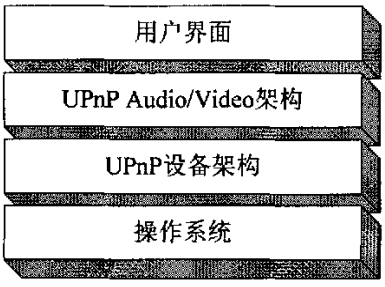

图 3.6 控制点系统架构图

Fig. 3.6 Control point system architecture

在控制点系统架构中，用户界面负责向用户提供控制界面，同时根据用户操作调用下层接口。UPnP Audio/Video 架构负责管理媒体内容和渲染器，通过对下层的调用来向上层提供服务。UPnP 设备架构是整个系统架构中最重要的部分，负责设备的发现与控制，同时向上层提供接口。

控制点系统架构的这种设计形式分别使用了 UPnP 设备架构和 UPnP Audio/Video 架构，因此继承了它们的优点，另外还具有如下特点：

(1) 支持 DLNA 协议，能够与其它 DLNA 设备兼容。

(2) 层次清晰，功能明确，易于实现。

(3) 可扩展性好，未来对某一层功能的扩展不会影响整个系统架构。

(4) 较好的平台无关性，易于在不同平台之间移植。

(5) 易于调试，易于维护。

3.4 系统结构设计

系统架构设计只是对控制点系统在层次上进行划分，只是从宏观上对系统的设计。下面对系统的架构进行细化，得出控制点系统的详细结构。

3.4.1 UPnP 设备架构结构的初步设计

首先对 UPnP 设备架构采用模块化设计方法进行初步设计。根据 UPnP 协议中的设备运行过程，同时考虑到本课题的设计目标，将 UPnP 设备架构划分为四个功能模块：设备发现模块、设备描述解析模块、设备控制模块和设备事件模块。下面分别分析和设计 UPnP 设备架构中的各个模块。

(1) 设备发现模块

设备发现主要包含以下几个过程：搜索网络上的可用设备和服务；监听网络上设备和服务的上线宣告消息、更新宣告消息和撤销宣告消息；更新和维护可用设备和服务的

信息。

设备发现过程的流程图如图 3.7 所示。

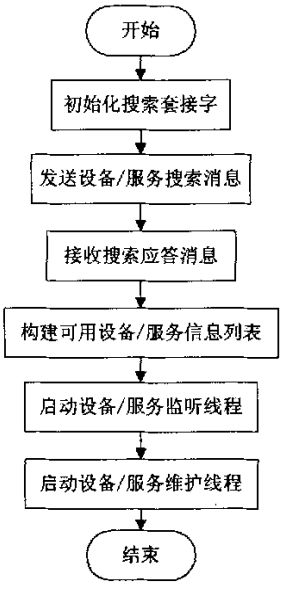

图 3.7 设备发现流程图

Fig. 3.7 Flow chart of device discovery

设备发现过程中的搜索消息、应答消息以及设备/服务的宣告消息都是由简单服务发现协议（Simple Service Discovery Protocol，简称 SSDP）来定义的。在设备发现过程要组装与解析消息。

设备/服务监听线程主要负责监听网络上可用设备和服务的更新宣告或离线宣告，以及新加入网络的设备和服务的上线宣告；当接收到这些宣告消息时，更新可用设备和服务信息列表。设备/服务监听过程的流程图如图 3.8 所示。

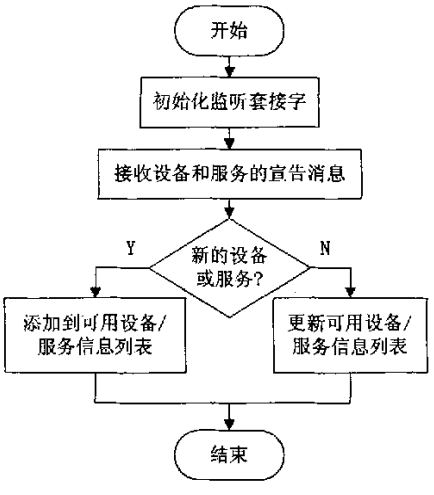

图 3.8 设备/服务监听流程图

Fig. 3.8 Flow chart of device/service monitor

设备/服务维护线程主要负责维护网络上可用设备和服务信息列表，定期检查是否有

设备或服务过期，保证列表中的设备和服务都可用。可用设备/服务维护过程的流程图如图 3.9 所示。

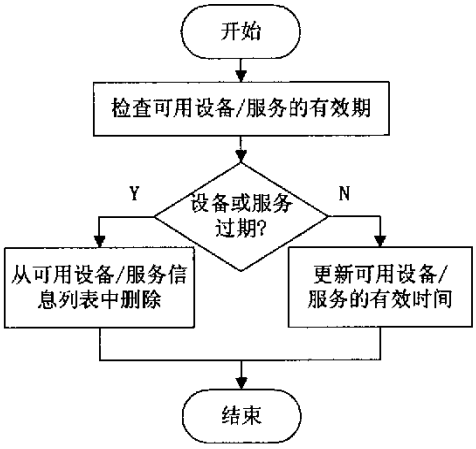

图 3.9 可用设备/服务维护流程图

Fig. 3.9 Flow chart of managing useable device/service

(2) 设备描述解析模块

设备描述主要包含以下几个过程：与设备建立连接；获得设备的设备描述和服务描述；对设备描述和服务描述进行解析。

设备描述过程的流程图如图 3.10 所示。

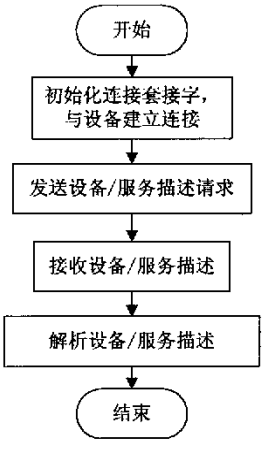

图 3.10 设备描述流程图

Fig. 3.10 Flow chart of device description

设备描述详细描述了一个 UPnP 设备的基本属性，包括设备类型、设备信息、提供的服务等等。服务描述详细描述了一个设备服务提供的动作、每个动作对应的参数和服务运行的状态变量等等。控制点通过解析设备的设备描述和服务描述来获得设备及其服务的属性和功能等详细信息。设备描述和服务描述都是以 XML 规范[17]来书写的，需要对描述文档进行解析。为了解析 XML 文档，需要设计一个 XML 文档分析器，它将 XML 文档分解为一种树状数据结构，文档中的每个字段都与树中的各个结点一一对应。控制

点获得设备的设备描述和服务描述之后，首先将它们解析成对应的 XML 树，然后将树中各个结点的属性值添加到可用设备和服务链表的各个结点中去。

一个 UPnP 设备描述文档经过解析之后建立的 XML 树状结构如图 3.11 所示。

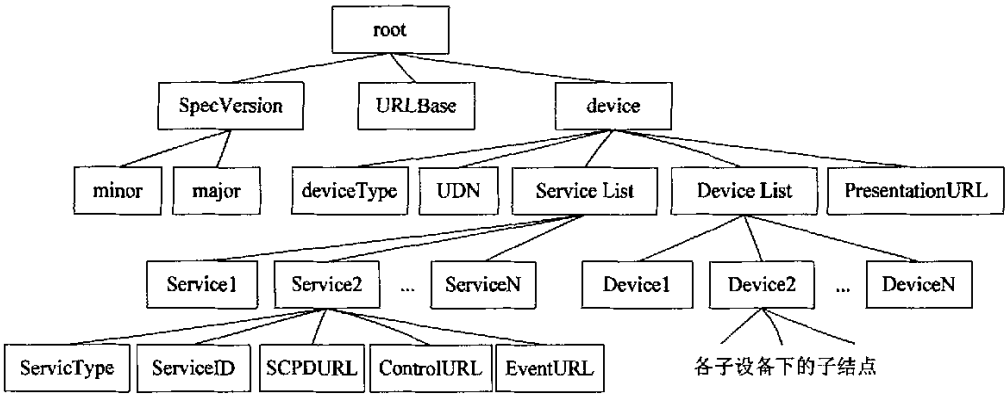

图 3.11 设备描述的 XML 树

Fig. 3.11 XML tree of device description

服务描述文档经过解析之后的 XML 树状结构与设备描述的相似，在此就不赘述。

(3) 设备控制模块

设备控制主要包含以下几个过程：组装控制请求数据包；向设备发送控制请求；接收控制请求的应答消息；解析控制应答消息结果。设备控制过程是一种远程过程调用，使用了简单对象访问协议（Simple Object Access Protocol，简称 SOAP[18]），该协议使用 XML 规范来描述控制请求和应答的格式，使用 HTTP 协议进行传输。

设备控制过程的流程图如图 3.12 所示。

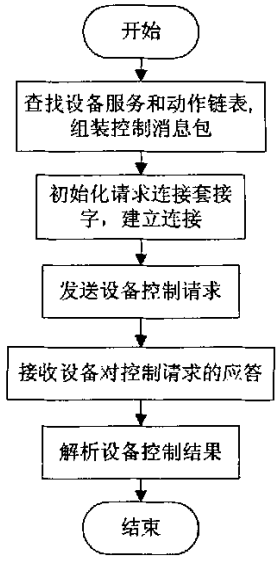

图 3.12 设备控制流程图

Fig. 3.12 Flow chart of device control

在设备描述解析模块中，设备描述文档经过解析之后，将设备提供的服务组成一个

设备服务链表，链表中的每个结点代表设备的一种服务，包含了服务控制 URL 地址，控制点是通过向服务控制 URL 地址发送控制消息来实现对设备服务的调用的。服务描述文档经过解析之后，将服务提供的动作组成一个服务动作链表，该链表中的每个结点代表服务的一种动作，包含了动作名称、参数列表。控制点在控制设备运行时，首先查询服务的动作名称和参数列表，按照相应的格式组装控制消息，从而实现调用设备服务执行相应的操作。

在设备控制过程中，控制点首先查询设备服务链表和服务动作链表，获得服务控制地址和要调用的动作的名称和参数列表，按格式组装好要发送的控制消息，然后向设备的服务控制 URL 地址发送控制请求消息，在消息中包含了要调用的服务的动作名称和参数。当设备接收到控制点的控制消息后，按照控制消息中包含的动作名称和参数，查找控制请求消息对应的动作的入口地址，然后开始执行动作请求，执行完毕后向控制点返回执行的结果，完成设备控制过程。

(4) 设备事件模块

设备事件主要有以下几个过程：订阅设备的服务；接收设备的事件通知；验证事件通知消息的正确性；维护设备和服务的状态信息；保持控制点对设备服务的订阅；根据需要撤销对设备服务的订阅。

设备事件过程的流程图如图 3.13 所示。

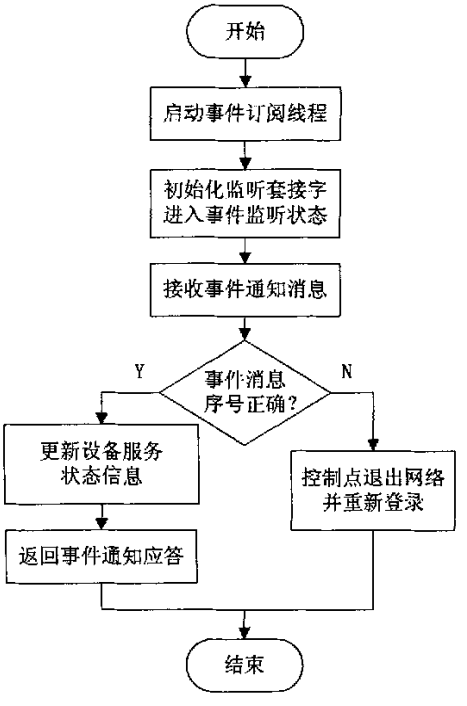

图 3.13 设备事件流程图

Fig. 3.13 Flow chart of device event

控制点接收到事件通知消息后，首先与期望收到的事件通知序号做比较，相同则说

明控制点运行过程中未出现丢包，才可以接受事件通知。如果发现事件通知序号不相同，则说明运行过程中有错误发生，控制点将先退出网络，然后重新登录。

事件订阅线程主要负责订阅控制点感兴趣的设备服务，并维护一个设备服务订阅列表，通过定期发送更新订阅消息来保持控制点对设备服务的订阅。事件订阅过程的流程图如图 3.14 所示。

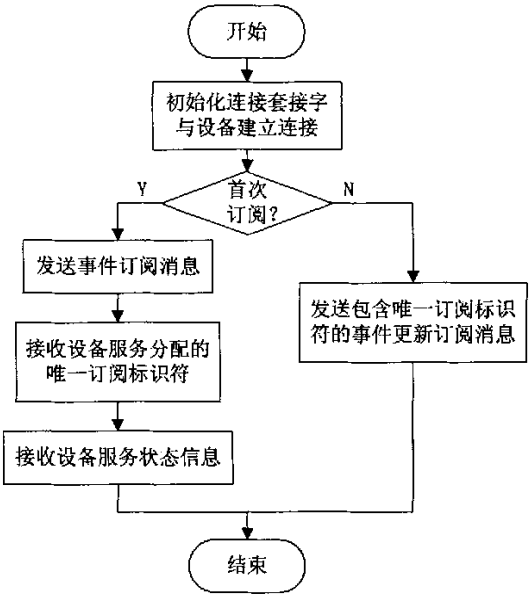

图 3.14 事件订阅流程图

Fig. 3.14 Flow chart of event subscription

控制点订阅一个设备服务的事件时，首先查询设备服务链表，找到该服务所对应的事件订阅 URL 地址，然后向此地址发送事件订阅消息，在订阅消息中主要包含控制点接收该服务事件通知的网络地址，设备在运行状态发生变化时，就是向这个控制点提供的这个网络地址发送事件通知消息通知控制点的。另外订阅消息中还要包含此次事件订阅的有效期。

控制点订阅了某一个设备服务后，该服务会为控制点分配一个唯一订阅标识符。控制点将其订阅的设备服务组成一个订阅服务链表，链表的每个结点都包含对应的唯一订阅标识符、该服务事件订阅的过期时间和期望收到的下一个事件通知消息的序号。

在控制点更新事件订阅时，只需提供该服务对应的唯一订阅标识符。控制点必须在事件订阅过期时间之前更新服务订阅。

设备事件过程中的订阅消息、事件通知消息以及撤销订阅消息都是由通用事件通知架构协议（General Event Notification Architecture，简称 GENA）来定义的。在设备事件过程要组装与解析消息。

上面对 UPnP 设备架构的设备发现、描述、控制和事件四个过程做了详细剖析，综合分析了设备运行的各个阶段所做的处理工作，以及各个过程之间相互通信相互协作的

机制。综合上面的分析过程，可以得出 UPnP 设备架构结构的初步设计如图 3.15 所示。

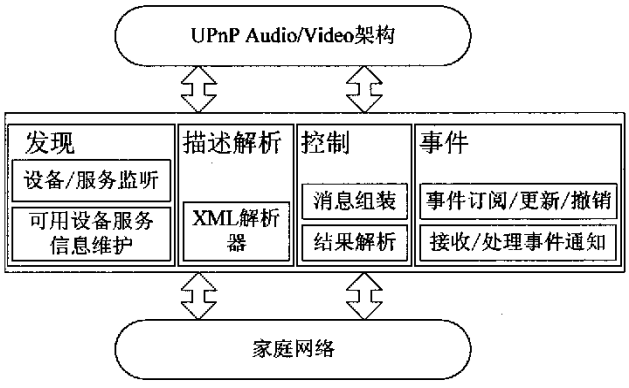

图 3.15 UPnP 设备架构的结构图

Fig. 3.15 UPnP device architecture chart

这样设计 UPnP 设备架构的结构有很多优点，如模块结构清晰，功能明确，便于模块化程序实现等；但还存在一些不可忽视的缺陷。

首先，在功能分配方面，各个模块之间有重叠的部分：

(1) 网络通信处理。在设备发现模块中，控制点使用发现消息搜索网络上的设备，监听设备和服务的宣告消息并处理。在设备控制模块中，控制点向设备发送控制消息，并处理设备对控制消息的应答。在设备事件模块中，控制点先要向设备服务发送事件订阅消息，接收设备服务返回的唯一订阅标识符；设备运行状态发生变化时，控制点会接收到设备服务的事件通知，并做出响应。

(2) 通信消息的组装和解析。在设备发现与控制过程中，各个模块都采用不同格式的消息。在设备发现模块中包含设备搜索消息，搜索应答消息，设备和服务宣告消息；在设备控制模块中包含设备控制消息和控制应答消息；在设备事件模块中包含事件订阅消息，更新订阅消息，撤销订阅消息和事件通知消息。在控制点运行的各个阶段，都需要对这些消息进行对应的组装与解析。另外，在设备描述解析模块中，还要对设备描述和服务描述进行解析，这些数据的处理和消息的处理在很多情况下都是相同的。

其次，在系统资源使用方面，这种设计结构需要创建六个线程：

(1) 设备和服务监听线程；

(2) 可用设备和服务维护线程；

(3) 设备描述和服务描述解析线程；

(4) 设备控制线程；

(5) 事件订阅和维护线程；

(6) 事件通知监听线程。

这些线程在控制点启动到控制点退出网络的整个过程中都是要运行的。在控制点的运行过程中，设备和服务监听线程始终监听网络，发现有新设备加入后首先更新可用设

备和服务列表，然后立即通知可用设备和服务维护线程、设备描述和服务描述解析线程以及事件订阅线程进行处理；控制线程需要始终运行来向上层提供服务；事件通知监听线程需要始终运行来监听设备的事件通知，以保证控制点随时掌握设备运行状态的变化。使用过多的线程会耗费系统资源，影响系统稳定性和运行效率。同时，这些线程之间的同步、通信和互斥也会降低系统的可移植性。

从上面的分析可以发现，目前设计的结构具有如下缺点：

(1) 模块间功能冗余度比较高；

(2) 系统资源依赖性高；

(3) 系统实现复杂度高；

(4) 系统移植性差；

(5) 容易引起系统稳定性差和运行效率低等问题。

可见，需要对目前设计的 UPnP 设备架构的结构进行进一步改进，以解决上面叙述的这些缺点。

3.4.2 UPnP 设备架构结构的改进设计

针对上面分析的结果，现在来对 UPnP 设备架构的结构进行改进，改进的目标主要包括两方面：减小模块间的冗余度，提高模块的通用性；降低对系统资源的依赖，提高系统性能。

3.4.2.1 针对提高模块通用性降低冗余度的设计

首先分析 UPnP 架构的初步结构图和设备发现和控制的各个流程图，可以发现：

(1) 初步结构图中的各个模块都包含网络连接的建立和网络数据通信的处理，很容易想到将这些功能从各个模块中提取出来，划分为一个套接字处理模块，专门负责控制点与设备之间的网络通信。

(2) 初步结构图中的各个模块都需要对各自的数据包进行组装和解析，只是各个模块的数据包定义格式不同，因此进行组装和解析时的规则不同而已。另外，UPnP 设备和服务描述都是按照 XML 规范来书写的，所以也需要对这些描述文档进行解析。根据这些特点，可以单独设计一个数据包组装与解析模块，来负责在设备发现与控制过程中各个阶段数据的封装与分析。

(3) 分析设备控制过程、事件订阅、事件更新订阅和事件撤销订阅过程。这些过程开始时，控制点都要先以“客户端”的身份向设备（“服务器”）发送连接请求，在连接建立之后以请求、应答的方式与设备进行通信。在这些过程中，控制点与设备之间的通信方式可以抽象为一个客户端与服务器之间的通信方式。所以设计一个 HTTP 客户端模块，负责这些过程中控制点与其它 DLNA 设备的通信。

(4) 分析控制点接收事件通知过程。与(3)中叙述的过程类似的是，在接收事件通知过程中，控制点与设备的通信方式也可以抽象为一个客户端与服务器之间的通信方式，

只是在这一过程中，控制点扮演了“服务器”的角色，设备扮演了“客户端”的角色。因此相应的，设计一个 HTTP 服务器模块，负责控制点接收事件通知过程中的通信。

(5) 分析(3)和(4)中叙述的过程可以发现，在那些过程中控制点与设备之间的通信都是基于连接的，即开始通信之前必须先建立一条通信链路。在设备发现过程中，根据 UPnP 协议的规定，设备搜索消息、设备/服务宣告消息等是通过运行于 UDP 协议之上的 HTTP 多播和 HTTP 单播来传送的，即设备发现过程中的通信是基于无连接的。所以需要单独设计一个设备和服务发现模块，负责设备发现过程中的通信。

(6) 在设备发现与控制的各个阶段都有一些数据需要维护和管理。包括：网络上可用设备和服务链表的创建和维护，设备描述和服务描述文档的解析，设备控制消息的发送和响应消息的处理，订阅的设备服务链表的创建和维护等等。因此划分一个控制点数据管理模块，专门处理这些数据信息。

经过上面的分析，将 UPnP 设备架构的结构重新划分为六个功能模块：套接字处理模块、数据包组装与解析模块、设备和服务发现模块、HTTP 服务器模块、HTTP 客户端模块和控制点数据管理模块。这种模块划分方式，将模块间冗余度减小到了最低，大大提高了模块的通用性。

3.4.2.2 针对降低系统资源依赖性的设计

下面针对如何降低对系统资源的依赖进行分析。通过设计一种基于阻塞/唤醒的单线程轮询机制，来管理设备的发现和控制过程，减小对系统资源的依赖。

首先分析设备发现与控制过程中的事件触发机制和数据流程。在整个过程中，控制点的可能触发的事件可以归纳为四种情形：

(1) 由控制点发起。此时控制点发现有设备加入或退出网络。数据流程主要是，当设备加入网络时，控制点监听到设备的上线宣告消息，然后根据上线宣告消息中的设备地址依次获得设备的设备描述和服务描述，经过解析之后将结果填充到可用设备和服务链表的对应的结点中去，同时控制点向感兴趣的设备服务发送事件订阅消息；当设备退出网络时，控制点监听到设备的离线宣告消息，将退出的设备及其提供的服务从可用设备和服务链表中删除。

(2) 上层调用。此时控制点根据用户的操作去访问设备。数据流程主要是，控制点查询可用设备链表和服务链表，根据实际调用情形将数据经过组装等一些处理之后经过网络发送给设备，然后从设备接收响应消息，经过解析等处理之后再将数据结果返回给上层。根据实际情况，控制点与设备可能会进行多次的通信。

(3) 由设备发起。此时设备向控制点发送数据。数据流程主要是控制点监听到设备的连接请求，与设备建立连接之后接收设备发送来的数据，先对其进行解析，然后根据具体情形进行处理，并向设备返回响应消息。由设备发起的事件触发情形主要是在设备的事件阶段，设备以事件通知的形式通知控制点其状态的变化。

(4) 时间中断。此时控制点对引起过期的事件进行处理。数据流程主要包括：对于

过期的设备或服务，控制点将其从可用设备或服务链表中删除；对于即将过期的事件订阅，控制点根据需要发送更新订阅消息。另外，在控制点与设备通信过程中建立的连接，在通信完毕后，这些连接空闲一段时间后需要关闭，以节省系统开销和网络资源。因此连接的关闭也会引起事件触发。

在上述可能触发的四种事件情形中，前三种事件归根结底都是有关网络套接字通信的处理，后一种是有关计时器的处理。因此可以设计一种机制，让控制点系统启动之后，首先经过一系列初始化，然后进入阻塞状态，当有上面描述的某一种事件触发时，就唤醒控制点开始相应的处理，处理完成后控制点再次进入阻塞状态，直到有其它事件触发或者控制点系统退出。

通过上面的分析，对控制点的事件触发机制进行抽象，本人设计了一种基于阻塞/唤醒的单线程轮询机制[14]。

(1) 设计思想

在线程中创建一个事件链表，链表中的各个结点表示控制点设备发现与控制过程中可能触发的不同事件，每个结点都包含该结点对应的事件触发处理入口地址指针。

线程开始运行后，控制点首先将其初始化，然后进入阻塞状态；当链表中的某一个结点有事件触发时，线程被唤醒开始轮询事件链表；对于事件链表中的每一个结点，线程跳转到该结点对应的事件触发处理入口地址去执行，判断该事件是否触发并做相应处理；执行完毕后线程继续轮询链表的下一个结点；当线程轮询完整个事件链表之后，就完成了所有的处理，再次进入阻塞状态，直到有下一个事件触发或者线程结束。

基于阻塞/唤醒的单线程轮询机制的流程图如图 3.16 所示。

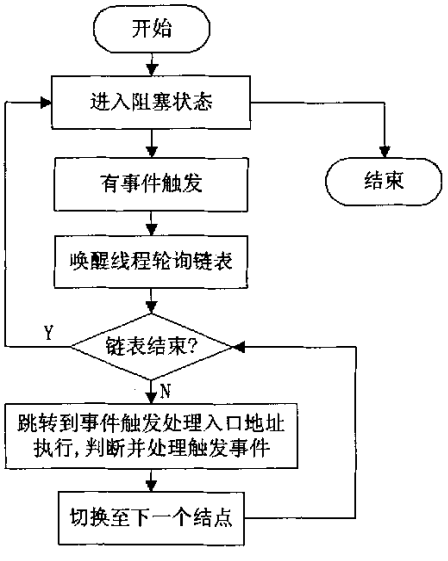

图 3.16 基于阻塞/唤醒的单线程轮询机制流程图

Fig. 3.16 Flow chart of a single-threaded poll mechanism with block/wake

(2) 优点

这种基于阻塞/唤醒的单线程轮询机制有很多优点：

(a) 首先，大大降低了对系统资源的依赖性，仅仅使用一个线程就可以完成设备发现与控制过程中所有触发事件的处理。

(b) 其次，系统使用线程数的减少，避免了程序设计时由于线程之间同步、通信和互斥带来的问题，减小了系统在不同操作系统平台之间移植时的工作量，增强了系统的可移植性。

(c) 同时，这种“有则处理，无则等待”的运作方式，减少了系统资源开销和状态切换，提高了系统的运行效率。

综合前面的分析，最终得出改进后的 UPnP 设备架构的结构图如图 3.17 所示。

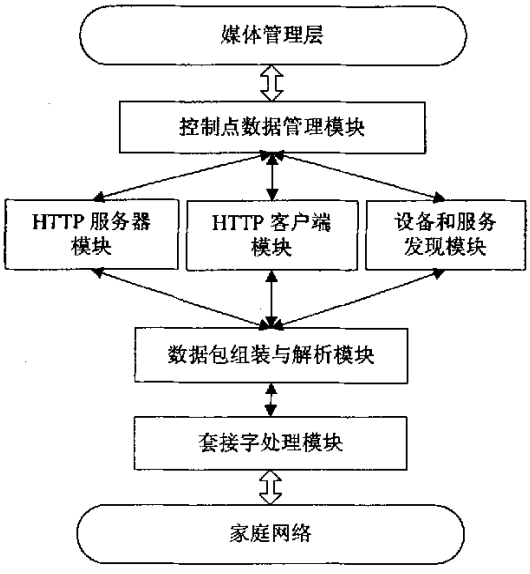

图 3.17 UPnP 设备架构的结构图（改进后）

Fig. 3.17 UPnP device architecture chart (improved)

改进后的 UPnP 设备架构的结构具有如下优点：

(1) 模块间高内聚，低耦合；

(2) 系统资源依赖性低；

(3) 系统稳定性好，运行效率高；

(4) 系统可移植性好。

3.4.3 UPnP Audio/Video 架构的结构设计

下面设计 UPnP Audio/Video 架构的结构。在媒体管理过程中，控制点调用媒体服务器和媒体渲染器提供的服务，协调并管理媒体服务器上的内容传输到媒体渲染器上进行播放，控制点不参与媒体内容的传输。

媒体管理过程如图 3.18 所示。

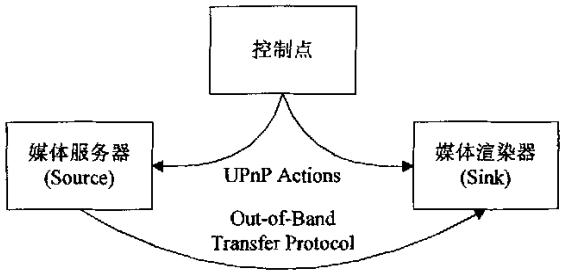

图 3.18 媒体管理过程示意图

Fig. 3.18 Illustration of media management

分析媒体管理过程，将 UPnP Audio/Video 架构中控制点按照其功能划分成两个模块：

(1) 控制点需要调用设备服务并处理设备返回的信息，所以设计一个服务调用处理模块来负责对设备服务的调用处理。

(2) 控制点需要维护网络上可用媒体内容信息和媒体渲染器信息，掌握设备的状态变化，因此相应地设计一个媒体设备管理模块。

UPnP Audio/Video 架构的结构图如图 3.19 所示。

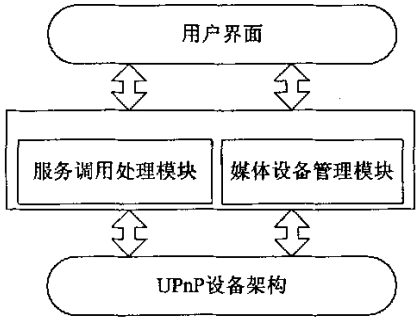

图 3.19 UPnP Audio/Video 架构结构图

Fig. 3.19 UPnP Audio/Video architecture chart

3.4.4 用户界面的结构设计

用户界面是控制点系统用来与用户交互的桥梁。本课题主要是通过实现 UPnP 设备架构和 UPnP Audio/Video 架构来实现 DLNA 协议，为了验证实现效果以及检验 DLNA 技术在实际中的应用，设计一个控制点系统的演示操作界面。

在用户界面的设计过程中，基于系统可移植性的考虑，将其设计为基于控制台的操作界面。将用户界面部分划分为两个模块：控制台管理模块和主控模块。

(1) 控制台管理模块提供控制点系统与用户的交互接口，根据用户的输入操作，调用底层提供的接口，管理控制点与其它设备的通信。

(2) 主控模块管理网络上的可用设备和服务，掌握各个设备的运行状态，管理控制点系统的运行过程。

经过上面的分析，设计用户界面的结构如图 3.20 所示。

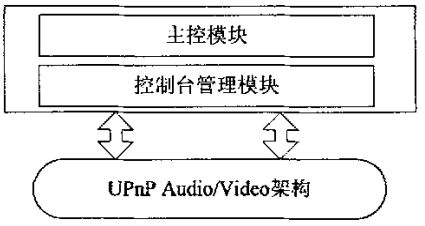

图 3.20 用户界面结构图

Fig. 3.20 User interface chart

综合前面对控制点系统架构中各个部分的系统分析过程，可以得出控制点系统的结构图如图 3.21 所示。

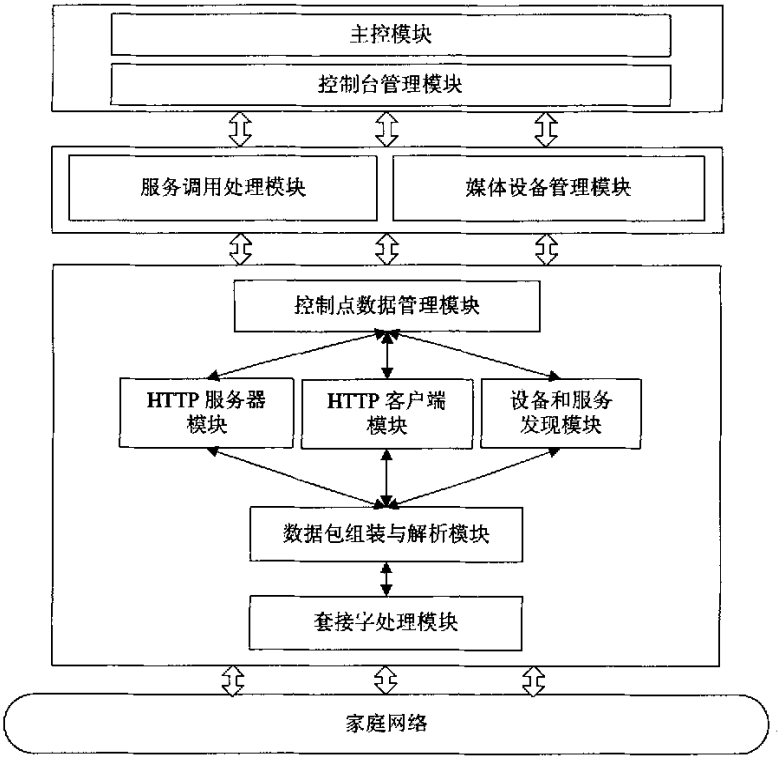

图 3.21 控制点系统结构图

Fig. 3.21 Control point system chart

3.5 本章小结

本章详细分析了控制点系统的设计过程。

首先明确了控制点系统的设计目标，确定在设计过程中需要考虑系统可移植性以及系统的稳定性和运行效率等要素。然后提出了系统的设计思想，分析了 UPnP 设备架构和 UPnP Audio/Video 架构的主要实现思想，确定了控制点系统与其它设备的主要通信机制是一种基于客户/服务器的通信模式。在明确系统设计思想的基础上，设计了控制点的系统架构，将控制点系统分为 UPnP 设备架构，UPnP Audio/Video 架构和用户界面三个

部分，并对系统架构的各个部分进一步细化，得到了控制点系统的模块结构。

在对 UPnP 设备架构的结构进行设计时，首先按照设备的发现和控制过程，对 UPnP 设备架构初步划分为四个模块：设备发现模块、设备描述解析模块、设备控制模块和设备事件模块，并详细分析了各个模块的运行机制。然后对这种初步设计结构做了评估，发现其中存在功能冗余以及对系统资源依赖性比较高等缺点。

针对初步结构设计中存在的缺点，对 UPnP 设备架构的结构做了改进设计。首先分析设备发现与控制的各个阶段的运行过程，提取各个过程的操作实质，将 UPnP 设备架构重新划分为六个模块：套接字处理模块、数据包组装与解析模块、设备发现和服务模块、HTTP 服务器模块、HTTP 客户端模块和控制点数据管理模块，这种划分方式大大提高了模块的内聚性，降低了模块之间的耦合性。然后综合分析控制点系统运行过程中的事件触发机制，设计了一种基于阻塞/唤醒的单线程轮询机制，使用一个线程来管理设备的运行，降低了控制点对系统资源的依赖性，同时增强了系统可移植性，提高了系统稳定性和运行效率。

第四章 DLNA 控制点系统的实现

4.1 基于阻塞/唤醒的单线程轮询机制的实现

在控制点系统的结构设计过程中，为了降低对系统资源的依赖性，本人设计了一种基于阻塞/唤醒的单线程轮询机制，使得控制点在设备发现与控制过程中的所有操作都在一个线程中完成。下面来分析这种机制的实现。

4.1.1 线程阻塞/唤醒的原理

在网络通信过程中，套接字在执行输入/输出操作时有两种模式[15]：阻塞模式和非阻塞模式。在阻塞模式下，在输入/输出操作完成前，执行操作的套接字处理函数（例如 send 和 recv）会一直等候下去，不会立即返回程序（将控制权交还给程序）；而在非阻塞模式下，套接字处理函数无论如何都会立即返回。

对于阻塞套接字，它的缺点是应用程序很难同时通过多个建立连接的套接字通信，其扩展性较差。对于非阻塞套接字，它的缺点是调用套接字处理函数获得成功返回之前，会频繁地返回“调用失败”的错误。

在这些情况下，引入了 select——多路复用技术，它使那些想避免在套接字调用过程中被阻塞的应用程序，采取一种有序的方式同时进行对多个套接字的管理。多路复用技术的本质是使用 select 函数实现对套接字输入/输出进行管理。

select 函数的目的是防止应用程序在套接字处于阻塞模式下时，在一次输入/输出调用过程中被迫进入阻塞状态；同时防止套接字处于非阻塞模式下时，产生调用失败错误。利用 select 函数，可以判断能否从一个套接字读出数据，或者能否向一个套接字写入数据。除非满足事先用参数规定的条件，否则 select 函数会在进行输入/输出操作时阻塞。select 的函数原型如下：

```
int select(
  int nfds,
  fd_set* readfds,
  fd_set* writefds,
  fd_set* exceptfds,
  const struct timeval* timeout
)
```

其中，参数 nfds 只是为了保持与早期的 Berkeley 套接字应用程序兼容。fd_set 数据类型代表一系列特定套接字的集合，readfds 参数用于检查可读性，writefds 参数用于检查可写性，exceptfds 参数用于例外数据。例如，假定想判断一个套接字是否可读，可以

在 select 函数调用前先将这个套接字添加到 readfds 集合，等待 select 函数调用返回之后，再判断这个套接字是否仍为 readfds 集合的一部分。若是，则表明该套接字上有数据可读，否则表明该套接字上没有数据可读。在这三个参数中（readfds，writefds 和 exceptfds）至少有一个不能为空值。

最后一个参数 timeout 是一个指向 timeval 结构的指针，用于设定 select 函数调用最多等待输入/输出操作执行多长的时间。在 timeout 设定的等待时间内，select 函数调用会一直阻塞，直到有符合指定条件的套接字出现后才返回；如果 timeout 是一个空指针，则 select 函数调用会无限期地阻塞下去，直到出现符合条件的套接字；如果等待时间设置为 0，则 select 函数调用会立即返回。select 函数成功返回之后，各个类型套接字集合（readfds，writefds 和 exceptfds）中包含对应的未完成输入/输出操作的套接字句柄，并向应用程序返回所有套接字句柄总数。若超过 timeout 设定的时间，则返回 0。如果 select 函数调用失败，则返回 SOCKET_ERROR。

用 select 函数对套接字进行监视之前，应用程序必须先根据需要将套接字句柄分配到各个 fd_set 集合（readfds，writefds 和 exceptfds）中去，再调用 select 函数，在其成功返回之后便可知道一个套接字上是否正在发生上述的输入/输出活动。表 4.1 中列出了对套接字集合进行处理和检查的宏操作。

表 4.1 操作套接字集合的宏

Table 4.1 Macros to operate fd_set

宏名称	功能
FD_ZERO(*set)	将 set 初始化为空集合
FD_SET(s, *set)	将套接字 s 加入集合 set
FD_ISSET(s, *set)	检查套接字 s 是否是集合 set 的一名成员；若是，则返回 TRUE；否则返回 FALSE
FD_CLR(s, *set)	从集合 set 中删除套接字 s

线程的阻塞/唤醒充分利用了多路复用技术的这种“满足事先用参数规定的条件则唤醒，否则就一直阻塞”的特性。线程运行时，首先对设备发现与控制过程中使用的套接字进行初始化，将它们设置为非阻塞方式，根据需要将它们分配到各个套接字集合中，并设置好等待时间，然后线程调用 select 函数进入阻塞状态；当有满足条件的套接字出现或者时间到期（有事件触发）时，就唤醒线程开始进行处理；对事件处理完毕之后，线程重新将这些套接字分配到各个集合中，设置好等待时间，然后调用 select 函数再次进入阻塞状态，直到下次被唤醒或者线程结束。

4.1.2 事件链表的数据结构

结合线程阻塞/唤醒的实现原理，下面来设计事件链表的头结点结构：

(1) 在线程轮询事件链表的过程中，为了使线程能够正常退出，需要设置一个线程结束标识。线程每次开始轮询之前，都要根据线程结束标识来判断是否要退出线程。

(2) 线程进入阻塞状态之后，有两种方式可以将其唤醒：一种是有符合条件的套接字出现；另一种是 select 函数调用的等待时间到期。因此可以通过设置 select 函数的等待时间，来管理线程的阻塞。如果将 select 函数的等待时间设置为 0，则线程跳过阻塞状态，直接去处理事件。利用这种机制，设计一个线程唤醒标识。在线程处理触发事件的过程中，如果有数据尚未接收或发送完毕，或者需要通知其它模块进行相应处理，则将线程唤醒标识置位；这样线程结束此次轮询事件链表处理触发事件之后，会在下一次 select 调用时立即返回，直接开始下一次轮询事件链表，进行相应的事件处理。

(3) 对于事件链表上的每一个结点都对应一个事件触发对象，事件触发对象中包含对应的事件触发处理的入口地址指针。因此设计一个触发事件的对象成员。

(4) 事件链表的下一个结点地址。

经过上面的分析，将事件链表的头结点结构设计如下：

```
struct EventChain
{
    int TerminateFlag;
    int InvokeFlag;
    void *Object;
    void *Next;
}
```

下面来分析控制点系统运行过程中都有哪些类型的事件触发，从而可以分析出事件链表结点的 Object 成员都可能有哪些种类型。

(1) 发生最频繁的事件是网络数据收发，即网络套接字上有数据读取或者有数据写入。产生网络数据接收或发送事件的对象包括：控制点以客户端的身份向设备发送连接请求和进行网络数据收发时使用的套接字，控制点以服务器的身份接受设备连接请求和进行数据收发时使用的套接字，控制点发现设备和服务过程中使用的套接字。

当网络套接字上有数据读取或者写入时，select 函数调用立即返回，线程立即被唤醒开始轮询事件链表，进行相应的数据处理。如果数据处理过程没有完成，就将线程唤醒标识置位，那么线程在完成本次事件链表轮询后，会跳过阻塞状态，直接开始下一次事件轮询，继续执行未完成的数据处理。

(2) 另一种事件触发类型是计时器触发，当某一个计时器的时间到期时，会触发一种事件。产生计时器触发事件的对象主要包括：到期的可用设备或服务，到期的服务事件订阅。另外，控制点与设备之间的连接在完成数据收发之后，根据实际情况可能需要立即断开或者等待一段时间后断开，此时也可以把该连接加入到计时器触发对象中，只需将连接断开计时器的有效期设置为零（表示立即执行）或所需等待的时间即可。

如何才能实现在计时器时间到期时立即唤醒线程进行事件处理呢？前面分析过，通过设置 select 调用的等待时间，可以管理线程的阻塞。这里再次利用这种机制。在轮询事件链表处理计时器事件触发对象时，首先更新计时器的有效期，然后将这个有效期作为下一次调用 select 函数时的等待时间，也就是对线程下一次处于阻塞状态的时间做了限定，保证在计时器到期之前线程一定能够处理计时器触发对象。

4.1.3 事件链表的操作

对事件链表的常用操作如表 4.2 所示。

表 4.2 事件链表的常用操作

Table 4.2 Event chain usual operations

接口名称	功能
CreateChain()	创建一个事件链表
AddToChain(Chain, object)	向事件链表中添加事件触发对象
StartChain(Chain)	开始轮询事件链表
UnBlockChain(Chain)	将线程唤醒标识置位
StopChain(Chain)	结束事件链表

4.1.4 轮询过程中的事件触发处理接口

在上面分析过，事件链表中的各个结点对应的事件触发对象的类型各不相同，因此在事件触发时所需要做的事件处理也不相同。在设计事件触发处理接口时，需要采用面向对象设计中“多态”的思想，首先设计好统一的接口，然后在实现各个事件触发对象时决定要对事件进行哪些具体的处理操作，详细实现这些统一的事件触发处理接口。这样，在事件链表轮询的过程中，控制点就会依据事件触发处理接口对应的实际入口地址去执行当前事件触发对象对事件的处理。

经过前面的分析，设计三个事件触发处理接口，将其定义如下：

```
void (*PreSelect)(void* object, fd_set *readset, fd_set *writeset, fd_set *errorset, int* blocktime);
void (*PostSelect)(void* object, fd_set *readset, fd_set *writeset, fd_set *errorset);
void (*Destroy)(void* object);
```

对于不同的事件触发对象，在实现这些处理接口时所做的事件处理操作可能不一样，但接口的规范必须参照上面来设计，并且不是所有的处理接口都必须实现，根据实际情况选择所需的实现即可。

当触发事件的对象是套接字时，PreSelect 接口用来初始化套接字，并将其分配到对应套接字集合中去；PostSelect 用来检查相应套接字集合中是否有套接字，有的话就做相应处理；Destroy 接口负责在线程结束时删除事件触发对象，释放申请的资源。

当触发事件的对象是计时器时，PreSelect 接口用来检查并更新计时器的有效期，并将这个有效期返回给线程，作为线程下一次的阻塞时间，从而保证计时器到期时会立即唤醒线程进行处理。对于 PostSelect 接口计时器事件触发对象可以不作实现。

综合上面的分析过程，可以得出轮询线程的执行过程如图 4.1 所示。

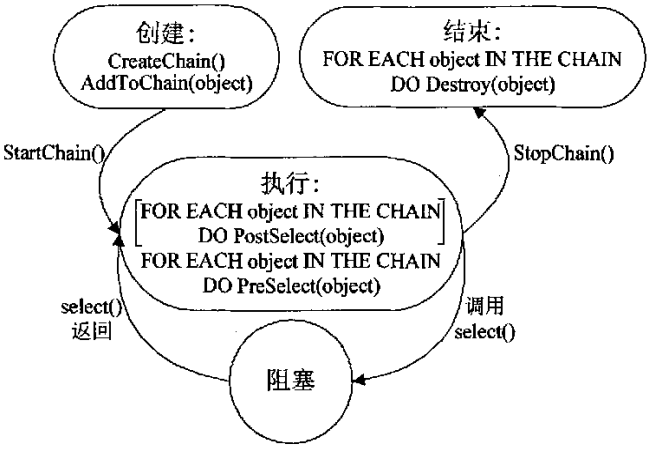

图 4.1 轮询线程运行过程

Fig. 4.1 Poll thread running process

线程启动之后，首先创建事件链表，向事件链表中添加的控制点的各个事件触发对象，设置好每个对象对应的事件触发处理接口。然后线程第一次轮询事件链表，执行每一个事件触发对象的 PreSelect 接口，对各个对象进行设置，例如将套接字分配到所需的套接字集合，更新计时器有效期等等。在对各个对象设置完毕之后，线程调用 select 函数，进入阻塞状态。当事件链表中有事件触发时，select 调用返回，线程被唤醒开始轮询链表，执行每一个事件触发对象的 PostSelect 接口，判断各个事件触发对象是否有事件发生并进行处理。处理完毕后线程又一次轮询事件链表，执行事件触发对象的 PreSelect 接口，重新设置各个对象，然后再次调用 select 进入阻塞状态，一直到下一次事件触发或者线程结束。线程结束之前最后一次轮询事件链表，执行每一个事件触发对象的 Destroy 接口，进行资源回收工作。

4.2 UPnP 设备架构的实现

4.2.1 UPnP 设备和服务

UPnP 设备和服务是 UPnP 设备架构中最基本的两个元素，是设备发现与控制过程中控制点处理的主要对象。首先来设计 UPnP 设备和服务的结构。

根据 UPnP 设备架构的定义，UPnP 设备的属性主要包括设备名称、设备类型、标识号、设备厂商信息、设备描述的地址、设备所包含的服务列表等。UPnP 服务的属性主要包括服务名称、服务描述地址、控制请求地址和事件订阅地址以及服务包含的一系列动作信息和服务的状态变量信息等。

通过上面的分析，分别设计四个结构体如下：

(1) struct UPnPDevice：表示 UPnP 设备结构；

(2) struct UPnPService：表示 UPnP 服务结构；

(3) struct UPnPAction：表示服务的动作结构；

(4) struct UPnPStateVariable：表示服务的状态变量结构。

这四种结构体之间的关系如图 4.2 所示。

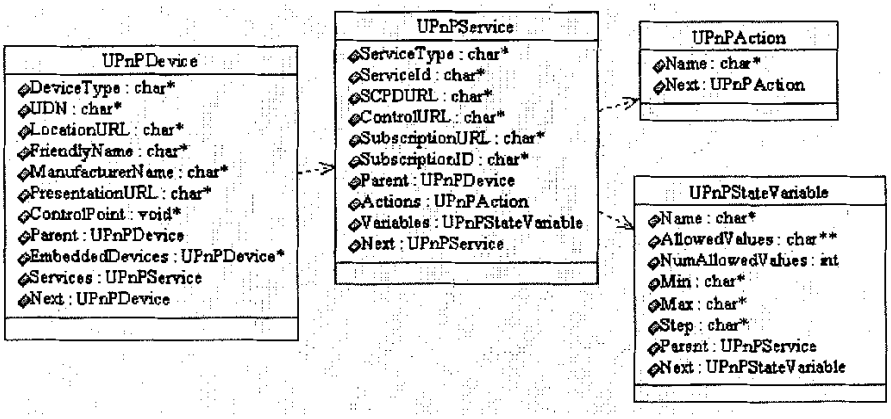

图 4.2 UPnP 设备与服务结构体关系图

Fig. 4.2 UPnP device structure and service structure relation

控制点管理和维护一个可用设备和服务链表，当发现新设备之后，首先解析设备描述和服务描述，然后将结果填充到设备链表和服务链表的各个结点中去。当有设备退出网络后，控制点将从可用设备链表和服务链表中删除该设备及其包含的服务。控制点对可用设备和服务的管理将在下面各个模块的实现中详细说明。

4.2.2 计时器

4.2.2.1 计时器的数据结构

在前面的分析中可以知道，在设备发现与控制过程中，有四种情形需要使用计时器进行时间控制：

(1) 监控可用设备和服务的有效期，当有设备或服务过期时，需要将其删除；

(2) 监控服务事件订阅的有效期，当事件订阅到期时，需要更新订阅；

(3) 控制点与设备之间建立的连接空闲一段时间后，需要将其关闭；

(4) 欲建立的连接重试一定次数后仍不成功，需要将其结束。

计时器要对各个对象的有效时间进行监控。当发现某个对象要过期时，就相应地执行处理工作，如删除该对象等等。另外，计时器是事件触发对象的一种，所以计时器在设计事件触发处理接口时必须符合规范。

在设计定时器的过程中，如果为每一个设备（或服务）都设计一个单独的计时器来监控，那么将大大增加事件链表的规模。可以设计将这些设备（或服务）组成一个链表，

使用一个计时器来监控，这样可以大大减小事件链表的规模。

通过上面的分析，设计一个计时器结构体 LifeTime，在结构体 LifeTime 中包含一个它要监控的对象链表，同时结构体 LifeTime 作为事件链表中的一种事件触发对象，还需要实现符合规范的事件触发处理接口。将需要监控的对象设计为结构体 LifeTimeMonitorData，其中主要包含对象名称、有效时间和到期时所需执行的处理等等。结构体 LifeTime 和结构体 LifeTimeMonitorData 的关系如图 4.3 所示。

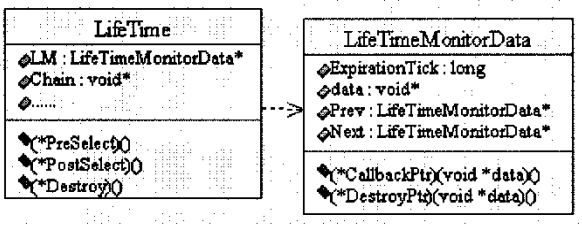

图 4.3 计时器的数据结构

Fig. 4.3 Data structure of time monitor

4.2.2.2 计时器的接口

对计时器的接口设计如表 4.3 所示。

表 4.3 计时器的接口

Table 4.3 Interfaces of time monitor

接口名称	功能
CreateLifeTime()	创建一个计时器实例
LifeTime_Add()	向计时器的监控对象链表中增加一个对象
LifeTime_Check()	检查监控对象是否到期，并做相应处理
LifeTime_Remove()	将一个监控对象从链表中删除
ILibLifeTime_Destroy()	删除一个计时器实例

在初始化的过程中，计时器首先创建监控对象链表 LM，将每一个需要监控的对象加入到该链表中，同时设置好监控对象的有效时间以及过期时进行处理的入口地址指针。

在线程轮询事件链表的过程中，当事件触发对象是计时器时，线程遍历计时器的监控对象链表 LM，查看并更新监控对象的有效期，如果到期，则线程跳转到该监控对象的到期事件处理地址（CallbackPtr）执行处理，并将该对象从监控对象链表中删除（DestroyPtr）；最后计时器将监控对象中有效期最小的值作为线程下一次的阻塞时间，以保证有监控对象到期时立即唤醒线程进行处理。

4.2.3 套接字处理

控制点与设备之间通过家庭网络进行通信，套接字处理负责在网络通信过程中连接的建立和数据的收发，是控制点系统用来接收和发送网络数据的最小逻辑单元。下面来

分析套接字处理的实现过程。

在套接字处理控制点与设备的通信时，主要有三个过程：创建和初始化套接字，建立连接；从套接字读取数据；向套接字写入数据。可以设计一个套接字处理结构体 AsyncSocketModule 来处理这些过程。

首先，很容易想到在 AsyncSocketModule 结构体中设计一个套接字类型的成员变量 internalSocket。由于在 AsyncSocketModule 结构体的使用过程中有两种方式：做为客户端主动向设备发起连接；做为服务器端被动接受来自设备的连接。因此在 internalSocket 套接字的初始化过程中就需要做不同的处理。

其次，在 AsyncSocketModule 结构体中设计一个数据缓冲区，用来接收从套接字读取的数据。由于套接字数据可能需要分几次才能接收完毕，所以套接字在只接收到部分数据时，应该能够允许上层应用程序来读取已接收到的数据。为了满足这个需求，将这个数据接收缓冲区设计成一个先入先出队列的结构。

还要设计一个数据缓冲区，用来缓存应用程序要向外发送的数据。考虑到应用程序可能会不断将数据交给套接字去发送，因此将这个数据发送缓冲区设计成一个链表的结构，每当应用程序有数据要发送时，就将其组装成一个待发送数据结点，插入到数据发送链表中。

另外为了表示一个套接字对象是否已经被使用，设置一个套接字使用状态标识符。在初始化套接字对象时将状态标识符置为“空闲”状态。控制点只有在套接字对象处于空闲状态时才可以使用它。

综合上面的分析过程，可以得出 AsyncSocketModule 结构体的数据结构如图 4.4 所示。

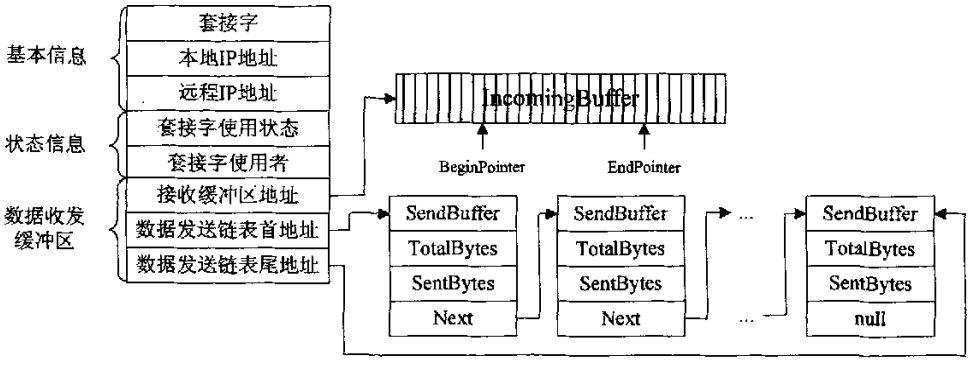

图 4.4 套接字处理的数据结构

Fig. 4.4 Data structure of socket control

控制点使用套接字发送数据时，首先查看数据发送链表，如果链表中有待发送的数据，则只需要将此次要发送的数据插入到数据发送链表尾部即可；如果数据发送链表为空，则首先将要发送的数据插入链表，然后开始使用套接字输入/输出函数开始数据发送。

控制点使用套接字接收数据时，首先查看数据接收缓冲区，如果缓冲区中包含已经被处理过的数据，则先把这些“无效数据”删除，将有效数据（由 BeginPointer 和 EndPointer 标识）移动到缓冲区首部。然后开始使用套接字输入/输出函数开始接收数据，并对数据进行处理。

由于套接字处理是事件链表中的一种事件触发对象，将其事件触发处理接口实现如下：

(1) 在 PreSelect 接口中，套接字处理对象将其 internalSocket 套接字分配到对应的读写套接字集合中。

(2) 在 PostSelect 接口中，套接字处理对象检查 internalSocket 套接字上是否有数据可读或可写，如果有则进行相应的处理。如果套接字对象需要继续处理操作，比如 internalSocket 套接字上有数据未发送完毕，则将事件链表的线程唤醒标识置位，让线程结束此次轮询事件链表之后，跳过阻塞状态，直接进入下一次事件链表的轮询，继续进行数据处理。

(3) 在 Destroy 接口中，主要操作是关闭套接字处理对象的 internalSocket 套接字，释放套接字的读写缓冲区等。

4.2.4 数据包组装与解析

在控制点与设备交互的过程中，不同的时期使用不同格式的数据包进行通信。数据包组装与解析就是负责在收到一个数据包后根据其格式将其解析成原始数据，在发送数据之前按照格式将数据组装成标准的数据包。另外，数据包组装与解析还负责对 UPnP 设备描述和服务描述（XML 文档）的解析。下面来分析数据包组装与解析的实现过程。

在设控制点系统的运行过程中，需要使用数据包组装与解析的情形包括：

(1) 在设备发现过程中，控制点发送的设备或服务搜索消息的组装；设备或服务对控制点搜索消息的应答消息的解析；设备或服务宣告消息的解析。这些消息的格式是由简单服务发现协议定义的。

(2) 在设备描述过程中，对设备描述和服务描述的解析。这些描述都是以 XML 协议的规范来定义的，因此也就是对 XML 文档的解析。

(3) 在设备控制过程中，发送控制消息时对控制消息的组装；收到设备对控制消息的应答时对应答消息的解析。这些消息的格式是由简单对象访问协议定义的。

(4) 在设备事件过程中，控制点对设备服务的事件订阅消息的组装；设备服务对事件订阅消息的应答消息的解析；设备服务的事件通知消息的解析。这些消息的格式都是由通用事件通知架构定义的。

在上面的这些过程中，数据包的接收和发送都是通过 HTTP 协议来传送的。因此首先设计 HTTP 包头的数据结构，定义 packetheader 结构来表示一个数据包头，该结构中包含数据包的版本、数据体长度等信息；为了提高灵活性，在 packetheader 结构中设计

一个链表来包含包头的各个域的值。数据包头的数据结构如图 4.5 所示。

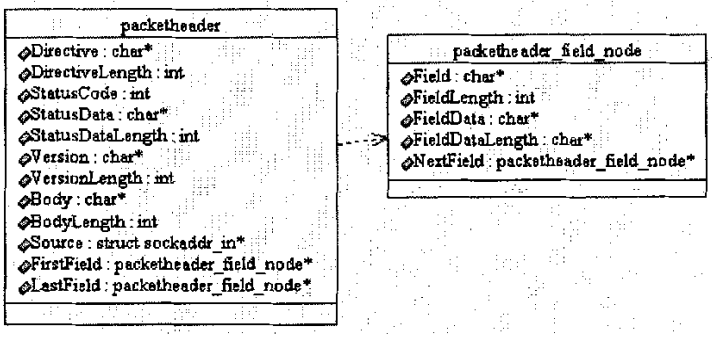

图 4.5 数据包头的数据结构

Fig. 4.5 Data structure of packet header

在数据包接收过程中，控制点先将收到的数据包头解析成一个 packetheader 结构，数据包头的各个域都填充到该结构的链表中去，然后提供给上层应用程序使用。在数据发送过程，控制点先创建一个空的 packetheader 结构，根据实际情况填充好结构中的各个域，然后将这个组装好的 packetheader 结构转换成对应的数据块，最后使用套接字处理将数据块发送出去。

由于 SOAP 协议使用 XML 规范来描述，因此设备控制消息和对应的应答消息也符合 XML 规范。UPnP 设备描述和服务描述也都是符合 XML 规范的文档。所以数据包组装与解析还要提供对 XML 文档的解析。在上一章中已经分析过，对于 XML 文档的解析，是设计一个 XML 分析器，将一个符合 XML 规范的消息包转化成一个 XML 树状结构。对 XML 树的结点 XMLNode 结构以及每个结点对应属性 XMLAttribute 结构的设计如图 4.6 所示。

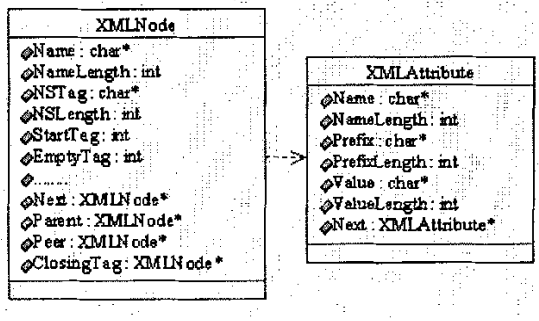

图 4.6 XML 结点及其属性的数据结构

Fig. 4.6 Data structure of XML node and attribute

在控制点与设备通信时传递的数据中，包含有很多特殊的符号，如控制消息中的“<”、“>”、“:”，HTTP 消息包中的“/r/n”、“,”、“/”、“-”和空格等等。控制点接收到一段数据之后，根据需要将数据分割为以特殊符号为边界的子串，然后将结果提供给其它模块使用。字符串解析中使用的数据结构如图 4.7 所示。

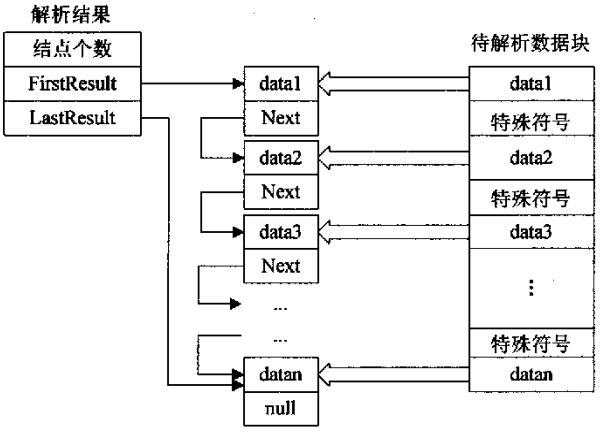

图 4.7 字符串解析结果的数据结构

Fig. 4.7 Data structure of string parse result

在对数据包的解析过程中，控制点首先使用字符串解析接口将数据包的包头从数据包中分离出来，然后根据数据包头的格式定义来将其解析成一个包含域链表的 packetheader 结构体。控制点可以查询该结构体的域链表来获得数据包的信息。

数据包组装过程与数据包解析过程相反。控制点首先创建一个空的 packetheader 结构体，根据实际需要填充结构体的域链表，然后将结构体转化为对应的数据包头数据块，再加上数据包体一起组成一个完整的数据包。

对 XML 文档的解析过程，实际上也是字符串的处理过程。首先使用字符串解析接口将 XML 文档分割为各个小部分，然后将分割结果经过判断填充到 XML 树的各个结点中去。对于设备描述和服务描述，控制点先将它们解析成对应的 XML 树，再使用 XML 树的各个结点来填充可用设备和服务链表。

4.2.5 设备和服务发现

设备和服务发现是家庭网络里 DLNA 设备实现互操作的基础。控制点通过设备和服务发现获得网络上可用的设备和服务，并对其进行管理和控制。下面分析设备和服务发现的实现过程。

在上一章分析过，设备和服务发现过程与设备控制过程和设备事件过程不同，它使用简单服务发现协议定义的运行于 UDP 协议之上的数据包进行处理。设备和服务发现分为两种方式：控制点发出设备和服务搜索消息，等待接收应答；控制点监听设备或服务的宣告消息。

由于设备和服务发现过程中的消息处理比较单一，并且使用基于无连接的套接字进行数据通信，因此不采用套接字处理对象来处理设备和服务的发现过程，而是重新设计一种新的结构来专门负责。

根据上面的分析，设计一个 SSDPSocketModule 结构来管理设备和服务发现过程。在 SSDPSocketModule 结构中，首先设计一个套接字用来监听设备和服务的宣告消息，

设计一个套接字用来发送设备和服务的搜索消息并接收响应，还要设计一个搜索关键字，来标识控制点要搜索的设备或服务的类型；

SSDPSocketModule 结构的数据结构如图 4.8 所示。

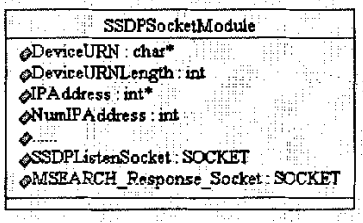

图 4.8 设备和服务发现的数据结构

Fig. 4.8 Data structure of device and service discovery

设备和服务发现是事件处理链表中的一种事件触发对象，将其事件触发处理接口实现如下：

(1) 在 PreSelect 接口中，设备和服务发现对象将其监听套接字和搜索套接字分配到读套接字集合中。

(2) 在 PostSelect 接口中，设备和服务发现对象分别判断监听套接字和搜索套接字上是否有数据可读，如果监听套接字上有数据可读则说明接收到了设备或服务的宣告消息，如果搜索套接字上有数据可读则说明收到了搜索消息的应答。

(3) 在 Destroy 接口中，设备和服务发现对象分别关闭监听套接字和搜索套接字，释放申请的资源。

4.2.6 HTTP 客户端

控制点系统使用 HTTP 客户端来向 DLNA 设备申请建立连接，发送请求消息，接收并处理设备的应答。控制点主要有以下过程需要使用 HTTP 客户端与设备进行通信：

(1) 设备控制过程中，控制点向设备发送控制请求消息；

(2) 设备事件过程中，控制点向设备发送事件订阅、更新订阅和取消订阅消息。

下面来分析 HTTP 客户端的实现过程。

首先设计一个 WebClientManager 结构来管理 HTTP 客户端的运行。在 WebClientManager 结构里，维护一组 AsyncSocketModule 结构的套接字处理对象来负责网络数据的接收与发送。控制点作为客户端与设备通信时，要先向设备发送请求消息，所以设计一个 WebRequest 结构来描述一个请求消息，其中包含请求的数据内容、目标设备的 IP 地址和控制点的一些信息。

在控制点与设备通信过程中，对于同一个设备，控制点只需使用一个套接字处理对象与设备建立一条连接；而对于不同的设备，控制点需要使用不同的套接字处理对象分别建立不同的连接。在每一条连接上，会发送不同的 WebRequest 结构的请求，并接收

对请求的应答。控制点需要有专门的机制来管理这些连接。所以设计一个WebClientDataObject结构，每个WebClientDataObject对象表示控制点与一个设备的连接，该结构包含连接使用的套接字处理对象、请求消息队列、连接状态标识符等信息。

控制点与设备的每一个连接都有四种状态：

(1) 等待建立连接，尚未建立；

(2) 已经建立的连接且其请求消息队列中包含请求消息；

(3) 已经建立连接且其请求消息队列为空，处于空闲状态；

(4) 连接已经断开。

因此在 WebClientManager 结构中设计一个连接等待建立队列，用来保存等待建立的连接；设计一个链表，保存所有已经建立的连接；设计一个链表，用来保存处于空闲状态的连接。

当一个连接处理完数据之后就进入空闲状态，为了节省系统资源，提高系统吞吐量，需要在连接空闲一定时间后将其关闭，释放其占用的套接字处理对象。当一个连接在建立过程中重试了一定次数之后，如果仍没有连接成功，也应该关闭该连接，以释放系统资源。为了管理连接的空闲状态和重试次数，设计一个计数器，来监控空闲连接和重试超时连接。

综合上面的分析，可以得出 HTTP 客户端的数据结构如图 4.9 所示。

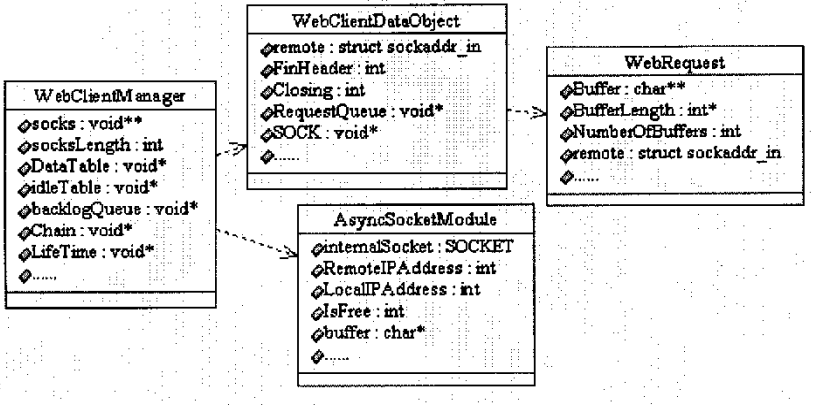

图 4.9 HTTP 客户端的数据结构

Fig. 4.9 Data structure of HTTP client

控制点向一个设备发送请求消息时，首先查找 HTTP 客户端的已建立连接链表，判断控制点与该设备是否已经建立连接：

(1) 如果连接尚未建立，则初始化一条新连接，将请求消息添加到新连接的请求消息队列，然后将这条尚未建立的新连接加入连接等待建立队列，同时更新已建立连接链表。

(2) 如果连接已经建立，那么控制点判断该连接的请求消息队列是否为空。如果请

求消息队列不为空，则说明该连接上有消息未发送完，只需要将待发送的请求消息加入请求队列即可。如果请求对列为空，则说明该连接处于空闲状态，首先将待发送的请求消息插入到连接的请求对列，然后将该连接从空闲连接链表中删除，开始使用该连接向设备发送请求。

对于 HTTP 客户端的计时器，其监控对象链表中包含两种类型的结点：

(1) 空闲连接。当一条连接的请求消息队列为空时，该连接就进入空闲状态，将其加入计时器的监控对象链表，并设定好过期时间。如果空闲连接到期时仍未进行数据收发，则断开该连接，释放系统资源。

(2) 重试连接。当一条连接初次建立时未成功，则将其加入计时器的监控对象链表，设定好等待重试的时间，当到期时就重新尝试建立连接；如果超过重试次数之后仍未成功，则终止该连接。

HTTP 客户端是事件链表中的一种事件触发对象，因此将其事件触发处理接口实现如下：

(1) 在 PreSelect 接口中，HTTP 客户端查看连接等待建立队列，如果有等待建立的连接，则选择一个空闲的套接字处理对象建立连接并开始发送请求消息。

(2) 在 Destroy 接口中，HTTP 客户端断开所有建立的连接，清空请求消息队列，释放占用的套接字处理对象，关闭套接字。

4.2.7 HTTP 服务器

HTTP 服务器运行时监听网络上的消息，当发现有客户端的连接请求时就接受并建立连接，收发数据并进行处理，最后向客户端返回应答消息。

控制点在设备事件阶段需要使用 HTTP 服务器与设备进行通信。在设备事件阶段，控制点监听到设备的连接请求，连接建立后控制点接收设备的事件消息通知，并做相应的处理，最后控制点向设备返回应答。

下面来分析 HTTP 服务器的实现过程。

首先设计一个 WebServer_StateModule 结构来管理 HTTP 服务器的运行。HTTP 服务器需要监听设备的连接请求，发现连接请求时需要建立连接和处理数据。因此在 WebServer_StateModule 结构中设计一个 AsyncServerSocketModule 结构成员，用来管理 HTTP 服务器的监听套接字 ListenSocket，维护一组套接字处理对象。当监听套接字 ListenSocket 发现连接请求时，就从 AsyncServerSocketModule 维护的套接字处理对象中选择一个空闲的来进行处理。当 AsyncServerSocketModule 维护的套接字处理对象都已经被使用时，控制点将不再接受设备的连接请求，一直到有新的套接字处理对象空闲的时候，控制点才继续监听并接受设备的连接。

为了管理控制点与设备之间的会话，设计一个 WebServer_Session 结构，会话中包含建立该会话使用的套接字处理对象，并创建一个 WebClientDataObject 结构的成员来管理

会话过程中数据的收发与处理。WebClientDataObject 的数据结构和运行机制已经在上一节中分析过。

当控制点接收到设备的连接请求时，控制点首先使用 AsyncServerSocketModule 维护的套接字处理对象与设备建立连接，然后创建一个会话，管理与设备之间的通信过程。

在会话创建之后，如果没有数据通信，则会话进入空闲状态，会话在空闲一定时间之后需要将其关闭，释放会话占用的套接字资源。在控制点向设备应答完毕之后，根据具体情形需要立即关闭或等待一段时间后关闭连接套接字。为了管理这些过程，在 HTTP 服务器中设计一个计时器。

综合上面的分析，设计 HTTP 服务器的数据结构如图 4.10 所示。

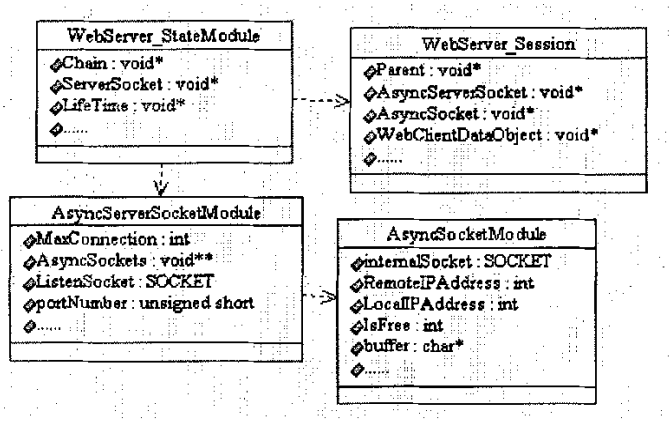

图 4.10 HTTP 服务器的数据结构

Fig. 4.10 Data structure of HTTP server

对于 AsyncServerSocketModule 结构，它负责监听设备的连接请求，使用套接字处理对象与设备通信，因此它是事件链表中的一种事件触发对象，将其事件触发处理接口实现如下：

(1) 在 PreSelect 接口中，将它的 ListenSocket 套接字初始化为非阻塞方式，并设置为监听状态，最后将套接字分配到读套接字集合中。

(2) 在 PostSelect 接口中，首先判断它的 ListenSocket 套接字是否在读套接字集合中，如果是则说明 HTTP 服务器监听到了设备的连接请求，就从它维护的套接字处理对象中选择一个空闲的套接字对象与设备建立连接，开始数据通信。

(3) 在 Destroy 接口中，关闭 ListenSocket 监听套接字，释放系统资源。

对于 HTTP 服务器的计时器，其监控对象链表中的结点类型只有 WebServer_Session 会话一种。当控制点与设备建立会话之后，如果会话上没有数据处理，就将其加入计时器的监控对象链表，将有效期设置为允许空闲的时间。当控制点向设备应答完毕后，根据使用的 http 协议版本的不同，控制点需要立即关闭套接字，或者等待一段时间之后再关闭套接字，所以将对应的会话加入计时器的监控对象链表，将有效期设置为 0（表示

立即关闭）或者等待的时间。当监控对象链表中的会话对象到期时，HTTP 服务器就将结束会话，关闭会话占用的套接字，释放系统资源。

HTTP 服务器是事件链表中的一种事件触发对象。在 HTTP 服务器的 Destroy 接口中，控制点释放 HTTP 服务器运行时占用的资源。具体套接字资源的释放会在 AsyncServerSocketModule 结构和套接字处理对象的 Destroy 处理过程中执行。

4.2.8 控制点数据管理

控制点数据管理模块是 UPnP 设备架构的中枢，它通过套接字处理模块、数据包组装与解析模块、设备和服务发现模块、HTTP 客户端模块和 HTTP 服务器模块的支持，维护 DLNA 网络上的可用设备和服务，控制和管理 DLNA 设备的运行。

下面分析控制点数据管理的实现过程。首先设计一个 UPNP_CP 结构，用来管理控制点系统的运行。对 UPNP_CP 结构的成员设计如下：

(1) 为了发现家庭网络上的设备和服务，定义一个 SSDPSocketModule 结构类型的成员，将发现的可用设备和服务组成一个链表，在 UPNP_CP 结构中包含此链表的首地址。

(2) 在设备控制过程，以及事件订阅、事件更新订阅和事件撤销订阅过程中，控制点以客户端的身份与设备进行通信。为了管理这些过程，定义一个 WebClientManager 结构类型的成员。

(3) 在设备事件过程中，设备向控制点发送事件通知，控制点以服务器的身份与设备进行通信。为了管理这个过程，定义一个 WebServer_StateModule 结构类型的成员。

(4) 可用设备和服务链表中的设备和服务都有有效期，当过期后该设备及其服务都被认为不可用。控制点对设备服务的事件订阅也是有有效期的。因此设计一个计时器，来监控这些对象的有效期。

综合上面的分析设计过程，控制点的结构体关系图如图 4.11 所示。

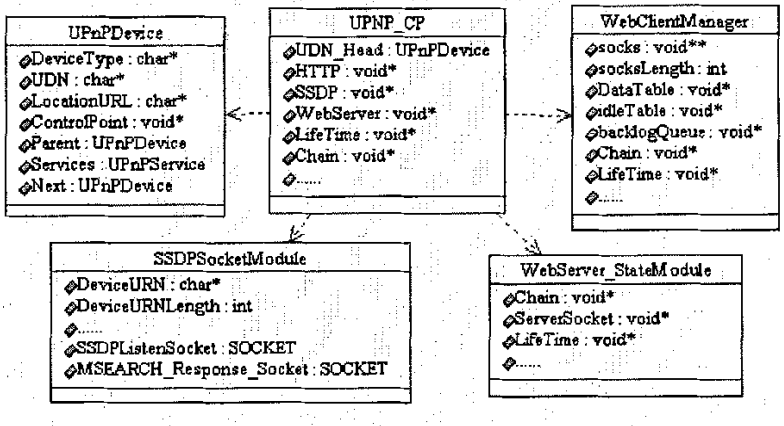

图 4.11 控制点结构体关系图

Fig. 4.11 Control point structure relation

控制点系统启动后，首先使用 SSDPSocketModule 搜索网络上的设备和服务，将搜

索结果填充到可用设备和服务链表，并监听设备和服务的宣告消息，当发现设备加入或退出网络，或者设备和服务更新其有效期时，控制点就更新可用设备和服务链表。

在设备描述阶段，控制点使用 WebClientManager 以客户端的身份向设备发送请求，获得设备的设备描述和服务描述。控制点首先使用数据包组装与解析模块中提供的接口，将设备描述和服务描述解析成对应的 XML 树，然后将 XML 树中的每个结点属性添加到可用设备和服务链表对应的结点中。每当控制点发现有新设备上线时，都将重复上述过程。

在设备控制阶段，控制点首先查询设备服务的动作链表，获得控制消息的方法名和参数，然后使用数据包组装与解析模块提供的接口按格式组装控制消息，接着使用 WebClientManager 以客户端的身份向设备发送控制请求，并接收到设备的响应消息之后，使用数据薄组装与解析模块的接口进行解析，将结果提交给上层。

在设备事件阶段，控制点首先查询可用设备链表，获得设备服务的事件订阅地址，然后使用数据包组装与解析模块提供的接口组装好事件订阅消息，接着使用 WebClientManager 以客户端的身份向设备进行事件订阅，并接收设备服务为控制点分配的事件订阅标识符。控制点订阅事件之后，使用 WebServer_StateModule 以服务器的身份监听设备的事件通知消息。

对于控制点的计时器，其监控对象链表中包含两种结点：

(1) 可用设备。当控制点发现一个可用设备时，就将该设备加入控制点计时器的监控对象链表，将有效期设置为设备宣告消息中指定的生命周期。当控制点收到一个可用设备的更新宣告时，就更新监控对象链表中该设备的有效期。如果在设备过期之前控制点没有收到该设备的更新宣告消息，则控制点认为该设备及其服务不可用，将其从可用设备和服务链表中删除。

(2) 设备服务的事件订阅。当控制点订阅了一个设备服务之后，就将被订阅的服务加入计时器的监控对象链表，将有效期设置为控制点事件订阅的有效期。当到期时，控制点就会更新设备服务的事件订阅，并且更新监控对象链表中该服务结点事件订阅的有效期。

控制点 UPNP_CP 是事件链表中的一种事件触发对象，对其事件触发处理接口实现如下：

(1) 在 PreSelect 接口中，控制点检查其 IP 地址是否发生变化。如果发生变化，则控制点首先清空可用设备和服务链表，然后使用 SSDPSocketModule 重新搜索可用设备和服务，重新订阅设备服务的事件，更新计时器的监控对象链表。

(2) 在 Destroy 接口中，控制点删除可用设备和服务链表中的各个结点，释放链表占用的系统资源。对于 UPNP_CP 结构中的 SSDPSocketModule 成员、WebClientManager 成员和 WebServer_StateModule 成员等的资源释放，分别在它们各自的 Destroy 接口中实现。

4.3 UPnP Audio/Video 架构的实现

根据 UPnP Audio/Video 架构的规定，将家庭网络上的设备分为媒体服务器、媒体渲染器和控制点。根据媒体内容传输时采用的协议，决定是由媒体服务器还是媒体渲染器提供 AV 传输服务。当采用的面向 push 的传输协议时，由媒体服务器提供 AV 传输服务；当采用的面向 pull 的传输协议时，由媒体渲染器提供 AV 传输服务。

本课题在实现 UPnP Audio/Video 架构时，采用 HTTP-GET 作为媒体内容的传输协议，因此由媒体渲染器提供 AV 传输服务。下面来分析 UPnP Audio/Video 架构的实现过程。

4.3.1 媒体设备管理

在 UPnP Audio/Video 架构中，控制点的主要职能是对网络上的媒体内容和渲染设备进行管理。控制点协调媒体服务器和媒体渲染器，使得媒体内容能够从媒体服务器传送到媒体渲染器进行渲染。

首先设计媒体内容对象的结构，媒体内容的属性主要包括名称、创建者、文件大小等基本信息，还包括该媒体内容的网络地址、所支持的文件格式和传输协议等必要信息。控制点通过调用 UPnP 设备架构提供的服务，发现家庭网络上的可用媒体内容，将这些媒体内容信息组成一个链表，提供给用户使用，而不论这些媒体内容是否来自同一个媒体服务器。

通过上面的分析，设计一个 MediaObject 结构，来表示一个媒体内容对象，其对应的数据结构如图 4.12 所示。

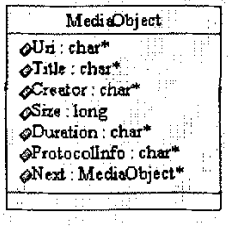

图 4.12 媒体内容对象的数据结构

Fig. 4.12 Data structure of media object

控制点系统启动后，首先发现网络上的媒体服务器，通过 UPnP 设备架构向媒体服务器发送控制消息，调用媒体服务器的内容目录服务，获取每个服务器提供的可用媒体信息，将收集到的媒体内容组织成一个可用媒体内容链表，并提供给用户使用。

控制点运行过程中，每当发现新的可用媒体服务器时，控制点都会浏览该媒体服务器提供的媒体信息，更新可用媒体内容链表。每当有媒体服务器退出网络时，控制点将其提供的媒体内容从可用媒体内容链表中删除。

下面来设计媒体渲染器的结构。因为媒体渲染器是一种 UPnP 设备，所以媒体渲染

器包含 UPnP 设备的所有属性，另外，媒体渲染器还需要包含所支持的文件格式和传输协议、待播放的媒体列表信息以及媒体渲染器建立的连接。对于一个媒体渲染器建立的连接，包含媒体内容的信息、连接使用的协议、传输状态和渲染状态等信息。对媒体渲染器对象的数据结构设计如图 4.13 所示。

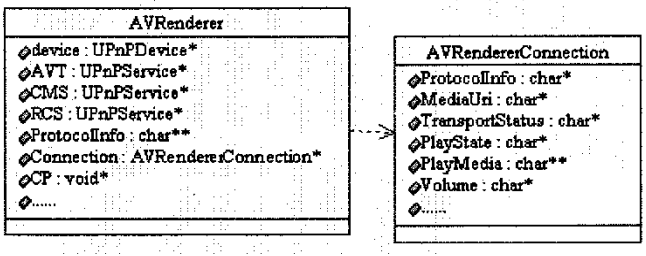

图 4.13 媒体渲染器对象的数据结构

Fig. 4.13 Data structure of media renderer

控制点启动后，通过 UPnP 设备架构发现网络上可用的媒体渲染器，获得每个渲染器支持的文件格式和传输协议，将媒体渲染器组织成一个可用渲染器设备链表，并提供给用户使用。每当发现有媒体渲染器加入网络或退出网络时，控制点都会更新可用渲染器链表。

在媒体渲染过程中，控制点首先根据用户的选择分别遍历可用媒体内容链表和可用渲染器链表，找到用户要使用的媒体渲染器和要播放的媒体内容；然后判断该媒体渲染器所支持的文件格式和传输协议是否能够满足媒体内容的要求；如果满足条件，则控制点创建一个新的连接，用来管理媒体内容从媒体服务器向媒体渲染器的传输过程。如果选定的媒体渲染器不支持要播放的媒体内容，则给出错误信息。

4.3.2 服务调用处理

UPnP Audio/Video 架构建立在 UPnP 设备架构之上。控制点对可用媒体内容和媒体渲染器的管理过程，以及控制点调用媒体服务器和媒体渲染器提供的服务来实现媒体内容的渲染过程，最终都是通过 UPnP 设备架构提供的机制来实现的。控制点通过设备发现来定位家庭网络上的媒体服务器和媒体渲染器；通过向设备发送控制消息，调用设备提供的服务来管理媒体服务器和媒体渲染器；通过设备事件消息，来掌握媒体渲染器的运行状态。

服务调用处理负责根据需要调用媒体服务器和媒体渲染器的设备服务，接收并处理它们返回的信息。控制点调用媒体服务器或媒体渲染器的某一种服务时，首先指定要调用的设备服务的方法名，设置对应的参数，然后通过使用 UPnP 设备架构提供的接口向目标设备发送控制消息；媒体服务器或媒体渲染器接收到控制消息之后执行相应的操作，并向控制点返回执行结果。控制点就是通过这种服务调用及返回的方式来控制和管理媒体服务器和媒体渲染器，进行媒体管理。

分析媒体管理过程，控制点需要调用并处理的服务主要有以下几种：

(1) 内容目录服务

内容目录服务[24]由媒体服务器提供，它列出了媒体服务器上可用的媒体内容信息以及各个媒体内容所支持的传输协议和数据格式。控制点调用内容目录服务之后，获得媒体服务器的返回结果，然后将每一个媒体对象的信息从结果中解析出来，添加到可用媒体内容链表中。每当控制点发现有新的媒体服务器加入网络后，都会调用该媒体服务器的内容目录服务，获得新的可用媒体内容。

(2) 连接管理服务

媒体服务器和媒体渲染器都提供连接管理服务[25]，它提供了设备本身所支持的媒体传输协议和媒体格式。控制点在创建一条新的 AVRendererConnection 连接之前，首先分别调用要建立连接的媒体服务器和媒体渲染器的连接管理服务，判断是否存在共同支持的传输协议和媒体格式可以实现媒体内容的渲染。如果存在，控制点就创建一条新的连接，并使用选定的传输协议和媒体格式来初始化，准备开始传输过程。如果不存在，则给出提示信息。

(3) AV 传输服务

AV 传输服务[26]由媒体渲染器提供。AV 传输服务中包含一系列方法，如播放、停止、暂停等，用来控制媒体内容在媒体渲染器上的渲染流程。控制点根据要渲染的媒体内容创建好 AVRendererConnection 连接，开始媒体渲染之后，就通过调用媒体渲染器的 AV 传输服务来控制媒体内容的渲染过程；根据用户的操作，控制点调用 AV 传输服务中对应的方法，控制媒体内容的渲染，满足用户的需要。

(4) 渲染控制服务

渲染控制服务[27]由媒体渲染器提供。渲染控制服务中包含一些方法，如静音开关、音量调节、亮度控制等，用来控制媒体内容在媒体渲染器上的渲染形式。在媒体内容从媒体服务器传输到媒体渲染器上进行播放的过程中，控制点通过调用媒体渲染器的渲染控制服务来控制媒体内容的渲染形式。

控制点上线后，首先定位网络上的媒体服务器和媒体渲染器设备，这是通过 UPnP 设备架构中的设备发现和设备描述来完成的。然后控制点向媒体服务器发送控制消息，调用内容目录服务，列出媒体服务器上可用的媒体内容。在进行媒体渲染时，控制点先查询媒体服务器和媒体渲染器，为待渲染的媒体内容选择共同支持的传输协议和数据格式，然后使用选定的协议和格式来配置媒体服务器和媒体渲染器，最后使用媒体渲染器提供的 AV 传输服务来管理媒体内容的传输过程。在进行媒体传输的同时，还可以使用媒体渲染器的渲染控制服务来调节渲染特性。

在媒体渲染过程中，控制点、媒体服务器和媒体渲染器之间的交互示意图如图 4.14 所示。

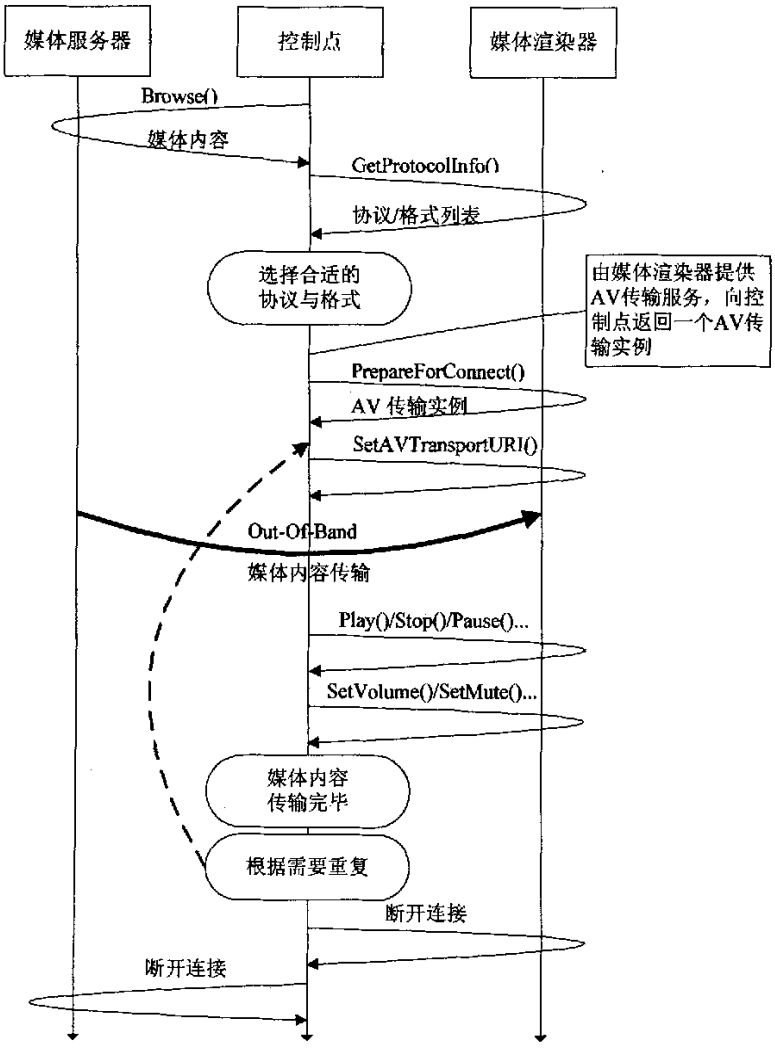

图 4.14 控制点、媒体服务器和媒体渲染器交互示意图

Fig. 4.14 Illustration of interactions among control point, media server and media renderer

在 UPnP Audio/Video 架构中，媒体服务器必须提供内容目录服务和连接管理服务；媒体渲染器必须提供渲染控制服务，连接管理服务和 AV 传输服务。控制点只负责通过调用这些服务来实现媒体内容从媒体服务器传输到媒体渲染器上进行渲染。因此，本课题不分析上述服务的实现过程。

4.4 用户界面的实现

为了检验 UPnP 设备架构和 UPnP Audio/Video 架构的实现情况，下面来实现控制点系统的演示操作界面。

在控制台管理模块中，为了管理用户的输入操作，定义一个命令列表。用户输入列表中的某一个命令，控制点就调用 UPnP Audio/Video 架构提供的接口，执行相应的操作，并将执行结果返回在屏幕上。当用户输入“quit”命令时，控制点系统结束运行。控制

点提供的操作命令如表 4.4 所示。

表 4.4 控制点的操作命令

Table 4.4 Control point operation commands

命令名称	功能
quit	关闭控制点
setavr	从可用的媒体渲染器列表中选择一个作为选定的媒体渲染器
getavr	输出当前选定的媒体渲染器的名称
setavm	从可用的媒体内容列表中选择一个作为选定的媒体内容
getavm	输出当前选定的媒体内容名称
play	使用选定媒体渲染器播放选定的媒体内容
pause	暂停选定媒体渲染器的播放
stop	停止选定媒体渲染器的播放
prev	使用选定媒体渲染器播放上一个媒体内容
next	使用选定媒体渲染器播放下一个媒体内容
volup	增加选定媒体渲染器的音量
voldown	减小选定媒体渲染器的音量
mute	切换选定媒体渲染器的静音开关
help	输出控制点的操作命令帮助信息

DLNA 控制点系统的运行过程如下：

(1) 初始化事件链表；

(2) 创建一个控制点，在初始化控制点的过程中，依次创建并初始化控制点的各个成员，将控制点及其成员添加到事件链表中；

(3) 开始轮询事件链表；

(4) 启动控制台管理，接收并处理用户的输入；

(5) 用户输入退出命令，停止轮询事件链表；

(6) 控制点系统运行结束。

4.5 本章小结

本章详细分析了 DLNA 控制点系统的实现过程。

首先阐述了实现基于阻塞/唤醒的单线程轮询机制的基本原理，设计了线程轮询过程中事件链表的数据结构和常用操作，根据阻塞/唤醒的实现机制，定义了事件链表中每一个结点需要实现的事件触发处理接口。基于阻塞/唤醒的单线程轮询机制的设计和实现，是对 DLNA 控制点系统设备发现与控制过程的高度概括。UPnP 设备架构的各个部分的实现都是围绕着基于阻塞/唤醒的单线程轮询机制来展开的。

在实现基于阻塞/唤醒的单线程轮询机制的基础上，本章逐步分析了 UPnP 设备架构、

UPnP Audio/Video 架构和用户界面的实现过程，分析了控制点系统各个模块中使用的数据结构的设计思路和设计方法，分别实现了 UPnP 设备架构的各个模块作为事件触发对象时需要提供的事件触发处理接口，剖析了控制点系统各个模块的运行机制和执行过程。其中 UPnP 设备架构的实现是控制点系统实现过程中的重点。

第五章 DLNA 控制点系统的互连性测试

5.1 测试环境

为了检验控制点系统与其它 DLNA 设备的互操作性，验证 DLNA 协议的实现成果，对控制点系统进行测试。搭建系统测试环境如下：

(1) 配置有 DHCP 服务器的以太网，DHCP 服务器用于对设备的网络地址进行管理；

(2) 一台连接到网络的计算机，运行 DLNA 控制点系统；

(3) 一台连接到网络的计算机，储存媒体内容文件，支持 DLNA 协议，充当媒体服务器；

(4) 一台连接到网络的计算机，可以播放媒体内容，支持 DLNA 协议，充当媒体渲染器。

系统测试环境如图 5.1 所示。

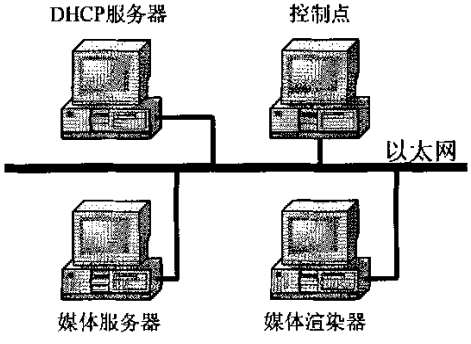

图 5.1 系统测试环境

Fig. 5.1 System test environment

5.2 测试方法

(1) 设备发现

启动 DLNA 控制点系统，查看它能否自动发现网络上的媒体服务器和媒体渲染器。测试结果如图 5.2 所示。控制点可以自动发现媒体服务器和媒体渲染器。

图 5.2 设备发现测试结果

Fig. 5.2 Test result of device discovery

(2) 媒体内容浏览

在控制台中输入“setavm”命令，查看控制点系统能否将搜索到的媒体服务器提供的媒体内容列出来。

测试结果如图 5.3 所示。控制点可以将媒体服务器上的媒体文件以列表的方式显示出来，供用户选择。

```
command>setavm
Select one Media Content of the following:
1)      music1.mp3
2)      music2.wma
3)      movie1.wmv
4)      movie2.avi

Select:_
```

图 5.3 媒体内容浏览测试结果

Fig. 5.3 Test result of media content browse

(3) 媒体内容渲染流程

首先在控制台中输入“setavm”命令，从媒体内容列表中选择一个作为当前媒体；然后在控制台中输入“setavr”命令，从可用媒体渲染器列表中选择一个作为当前渲染器。最后在控制台中输入“play”、“pause”、“stop”等命令，查看媒体内容的渲染流程。

测试结果如图 5.4 所示。在控制台中顺序输入“play”、“pause”、“stop”命令时，媒体内容在媒体渲染器上相应的开始播放、暂停和停止。

```
command>setavr
Select one Media Renderer of the following:
0)      Media Renderer Application(NEUSOFT-ZHY)

Select: 0
command>play
Current Renderer: Media Renderer Application(NEUSOFT-ZHY)
Current Media Content: music1.mp3
Current Status: Playing
command>pause
Current Renderer: Media Renderer Application(NEUSOFT-ZHY)
Current Media Content: music1.mp3
Current Status: Paused
command>_
```

图 5.4 媒体内容渲染流程测试结果

Fig. 5.4 Test result of media content render flow

(4) 媒体内容渲染方式

首先在控制台中输入“setavm”命令，从媒体内容列表中选择一个作为当前要渲染的媒体内容，输入“setavr”命令，从可用媒体渲染器列表中选择一个作为当前媒体渲染器；然后在控制台中输入“play”命令，使指定的媒体渲染器开始渲染指定的媒体内容；最后在控制台中输入“volup”、“voldown”和“mute”命令，查看媒体内容渲染方式的变化情况。

测试结果如图 5.5 所示。在控制台中顺序输入“volup”、“voldown”和“mute”命令时，媒体渲染器相应地增大音量、减小音量、切换静音。

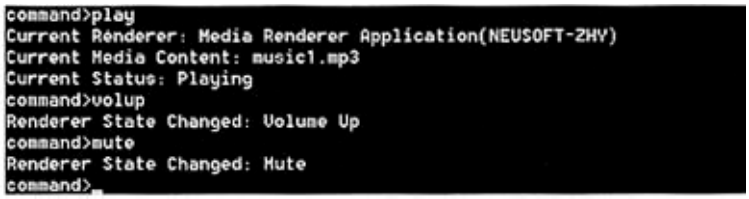

图 5.4 媒体内容渲染方式测试结果

Fig. 5.4 Test result of media content render manner

5.3 测试结论

DLNA 控制点系统能够自动发现网络上的媒体服务器和媒体渲染器设备，自动发现网络上可用的媒体内容信息，能够协调使得媒体内容从媒体服务器传送到媒体渲染器上进行渲染，并控制媒体的渲染流程和渲染方式，实现媒体管理。

DLNA 控制点系统无需配置就可以与其它网络设备组成无缝的互操作网络，很好地实现了 DLNA 协议。在测试过程中，控制点系统的稳定性和运行效率也是很不错的。

第六章 结束语

DLNA 技术在数字家庭媒体中心中建立了一个集中管理电脑设备、家电设备和移动设备的互操作网络，创造了一个能够共享和发展全新数字媒体和内容服务的无缝环境。

本课题以由 Intel 主导的、在实现设备互连和媒体管理方面有显著优势的 DLNA 技术为研究对象，深入分析了 DLNA 技术进行设备互连和媒体管理的原理、过程和方法，设计并实现了一种基于阻塞/唤醒的单线程轮询机制，实现了 DLNA 协议，并在此基础上实现了一个 DLNA 控制点系统。

在本课题的研究与开发过程中，本人受益匪浅，主要总结为以下几点：

(1) 系统了解了 DLNA 互操作架构中的各个技术要素，重点掌握了其中的 UPnP 设备架构和 UPnP Audio/Video 架构。

(2) 深入理解了 UPnP 设备架构实现设备发现与控制的原理、方法和过程；深入理解了 UPnP Audio/Video 架构实现媒体管理和控制的原理、方法和过程。

(3) 理解了 TCP/IP 协议和 Web 技术在 DLNA 技术中的应用，在设计实现控制点的过程中查阅了很多书籍和相关资料。

(4) 掌握了大量网络通信开发技术。在开发过程中，为了设计使控制点系统有较好的可移植性，研究了不同平台下的网络开发技术。

(5) 获得了丰富的测试方法和调试经验。

本课题以研究 DLNA 技术如何实现设备互连和媒体管理为主要目标，围绕着这个目标，本人做了很多工作，并取得了显著的成果。总结整个课题的研究过程，本人的主要工作成果如下：

(1) 实现了 UPnP 设备架构和 UPnP Audio/Video 架构，实现了 DLNA 协议。

(2) 在实现 UPnP 设备架构的过程中，对系统的稳定性、可移植性和运行效率做了透彻地研究，设计了一种基于阻塞/唤醒的单线程轮询机制。这种机制是对设备发现与控制过程的高度抽象和概括，将设备发现、描述、控制与事件等机制集中在一个线程中实现，提高了系统运行效率，降低了系统对设备资源的需求，增强了系统可移植性。

(3) 设计并实现了一个完整的 DLNA 控制点系统。DLNA 控制点系统可以在家庭网络中使用，自动识别支持 DLNA 协议的家电设备，实现设备之间的互连和互操作，发布和管理设备上的媒体内容信息，控制设备的运行。另外，DLNA 控制点系统是基于 Windows XP 平台开发的，经过简单地移植后就可以在 Linux 平台和 Windows CE 平台上运行，对于操作系统的依赖性非常低。

DLNA 控制点系统实现了 DLNA 协议，能够自动发现并控制媒体中心中的 DLNA 设备，能够对媒体内容进行控制和管理。但是由于 DLNA 技术本身的问题，使得控制点系统仍然存在很多不足之处：

(1) 控制点系统不支持中文名称的媒体文件。控制点系统无法获取到媒体服务器提供的中文名称的媒体内容，这大大降低了控制点系统的使用范围。

(2) 控制点系统要求家电设备具备相当的处理能力，支持 TCP/IP 协议栈，这对传统家电设备来讲是难以实现的。

(3) 控制点系统在版权管理和内容保护方面存在不足。由于控制点系统无需通过任何验证就可以访问家庭网络中的任何设备，这使得一些需要保护的内容可能会遭到非法复制和使用，可能会破坏某些设备的安全性。

上述缺点需要未来对 DLNA 技术做进一步研究与完善，使 DLNA 技术更好地实现家电设备的互连和互操作，更好地推动数字家庭的发展，更好地满足人们物质和文化生活的需要。

参考文献

1. Digital Living Network Alliance Home Networked Device Interoperability Guidelines. http://www.dlna.org/members/DLNA_Home_Networked_Device_Interoperability_Guidelines_v1.0.pdf
2. 苗再良. 数字家庭网络现状和发展趋势[J], 信息技术与信息化, 2005, 3: 7-10
3. 曹玖新, 张德运, 普杰信. 家庭网络技术与发展[J], 微型机与应用, 2000, 19 (9): 4-8
4. 张选芳, 李廷元. 网络即插即用技术的实现与比较[J], 西南民族大学学报, 2004, 30(3): 346-352
5. 杨思忠, 刘锦德, 骆志刚. 家庭网络及相关技术[J], 计算机应用, 2000, 20(7): 24-28
6. 刘云, 张红. 国外家庭网络技术标准进展分析[J], 信息技术与标准化, 2005, 3: 23-26
7. 周晓, 沈振宇, 陈鸣. 服务发现机制的比较与分析[J], 计算机工程与科学, 2003, 25(2): 56-60
8. 常锋. 中间件在家庭网络中的应用[D], 济南: 山东大学, 2005
9. 周新华. 智能家庭网关的兼容性研究及其发现[D], 上海: 东华大学, 2004
10. UPnP Device Architecture. http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf
11. Understanding Universal Plug and Play. http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc
12. Overview of UPnP AV Architecture. http://cache-www.intel.com/cd/00/00/21/87/218764_218764.pdf
13. 谢希仁. 计算机网络（第二版）[M], 北京: 电子工业出版社, 1999, 35-48, 92-127, 178-206
14. Douglas E Comer. 用 TCP/IP 进行网际互连——第一卷:原理、协议与结构[M], 北京: 电子工业出版社, 2001, 53-60, 144-174, 279-308
15. Anthony Jones. Windows 网络编程技术[M], 北京: 机械工业出版社, 2000, 172-179
16. 周晓, 蒋序平, 陈鸣. 网络即插即用及其体系结构[J], 解放军理工大学学报, 2002, 3(2): 1-5
17. Extensible Markup Language (XML) 1.0 (Third Edition). http://www.w3.org/TR/2004/REC-xml-20040204/
18. Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/2000/NOTE-SOAP-20000508
19. Brent A. Miller, Toby Nixon, Charlie Tai and Mark D. Wood. Home networking with Universal Plug and Play[J], IEEE Communications Magazine, 2001, 39(12): 104-109

20. UPnP Technology. http://www.intel.com/labs/connectivity/upnp
21. UPnP AV 1.0 Specifications. http://www.upnp.org/standardizeddcps/mediaserver.asp
22. MediaServer:1 Device Template Version 1.01. http://www.upnp.org/standardizeddcps/documents/MediaServer1.0.pdf
23. MediaRenderer:1 Device Template Version 1.01. http://www.upnp.org/standardizeddcps/documents/MediaRenderer1.0_000.pdf
24. ContentDirectory:1 Service Template Version 1.01. http://www.upnp.org/standardizeddcps/documents/ContentDirectory1.0.pdf
25. ConnectionManager:1 Service Template Version 1.01. http://www.upnp.org/standardizeddcps/documents/ConnectionManager1.0.pdf
26. AVTransport:1 Service Template Version 1.01. http://www.upnp.org/standardizeddcps/documents/AVTransport1.0.pdf
27. RenderingControl:1 Service Template Version 1.01. http://www.upnp.org/standardizeddcps/documents/RenderingControl1.0.pdf
28. 司端锋, 韩心慧, 龙勤, 潘爱民. SIP 标准中的核心技术与研究发展[J], 软件学报, 2005, 16(2): 239-250
29. 杨志明. 基于 UPnP 技术的媒体服务器的设计和实现[J], 计算机应用与软件, 2005, 22(6): 126-128
30. 郑振宇. 家庭数字媒体中心与 UPnP 架构[J], 闽江学院学报, 2004, 25(2): 40-44
31. 蔡安妮, 孙景鳌. 多媒体通信技术基础[M], 北京: 电子工业出版社, 2003: 128-150
32. 王玉林. 家庭网络中间件技术 UPnP 的研究与实现[D], 成都:电子科技大学, 2003
33. 蔡觉婷. 数字家庭网络技术与发展[J], 电信网技术, 2005, 6: 63-67
34. 韩麟. 通用即插即用（UPnP）体系结构在可靠性方面的改进与实现[J], 计算机工程, 2004, 30: 580-583
35. 江波, 韩江洪, 张利, 张建军. 中间件技术在智能家庭网络中的应用[J], 合肥工业大学学报, 2005, 28(5): 456-458
36. 杨思忠, 刘锦德. 网络即插即用及相关问题[J], 计算机科学, 2001, 4: 104-107
37. 叶朝辉, 杨士元. 智能家庭网络研究与开发[J], 计算机应用研究, 2002, 19(6): 38-40
38. Razss K. A plug-and-play approach with distributed computing alternatives for network configuration management[D], Ottawa, Canada: Systems & Computer Engineering Dept, Carleton University, 1999
39. Guttman E. Service location protocol: automatic discovery of IP network service[J], IEEE Internet Computing, 1999, 3(4): 71-80
40. Guttman E, Perkins C, Veizades J. Service Location Protocol, v2[S], Internet Engineering Task Force (IETF), RFC 2608, 1999

41. 董炜, 杨士元, 汪锐. 家庭网络体系结构及其服务发现技术[J], 计算机应用研究, 2005, 10: 9-13
42. Sumi H. Standards for Service Discovery and Delivery[J], IEEE Pervasive Computing, 2002, (2): 95-100
43. Pavlin D, David F, Christian K. Device and Service Discovery in Home Networks with OSGi[J], IEEE Communication Magazine, 2002, (8): 86-92
44. *Allard J, Chinta* V, Gundala S. Jini Meets UPnP:An Architecture for Jini-UPnP Interoperability[C], IEEE Proceedings of the 2003 Symposium on Applications and the Internet, 2003, 268-275
45. Stevene C, Beny Z, Toddd H. An architecture for a secure service discovery service[A], In: Fifth Annual International Conference on Mobile Computing and Networks[C], Seattle, WA, 1999

致 谢

两年半的研究生学习即将结束，回顾在东软软件股份有限公司进行课题设计的这段日子里，我深深地感受到了这里浓厚的学习氛围、良好的科研环境和先进的企业文化，同时东软人务实创新的工作态度和精诚团结的团队精神也都给我留下了深刻印象。

在课题即将结束之际，我首先要衷心感谢我的导师袁淮老师，袁老师在网络技术领域的丰富经验和开拓精神使学生受益匪浅；袁老师严谨的治学态度和踏实的工作作风将永远是学生学习的榜样。

我还要特别感谢我的项目组长杨梦晖和项目组的其他同事。每当我在学习和工作中遇到困难时，他们总是一如既往地支持我、鼓励我，使我能够顺利地完成毕业设计。

感谢我的母校东北大学，给我提供了良好的学习和生活环境，让我能够潜心学习文化知识和专业技能。

最后要感谢父母对我的养育之恩，他们在物质和精神上不断地给我鼓励和支持。我今天所取得的成果与他们的付出是分不开的。