开放式体系架构的数字家庭中心服务器
——智能化构件容器的研究与实现

罗克露，姜连波

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】介绍了基于双向数字机顶盒实现互联互通控制中心即数字家庭中心服务器，并结合构件技术与设计模式，提出了适用于数字家庭中心服务器的智能化构件容器编程模型；根据该模型在Linux系统下进行了实现。结果表明，该方法能从很大程度上提高软件的开发效率，从而为数字家庭中心服务器提供了一种可行的解决方案，对该领域具有借鉴意义。

关 键 词 中心服务器； 构件； 容器； 数字家庭； 机顶盒

中图分类号 TP311.52 **文献标识码** A doi:10.3969/j.issn.1001-0548.2010.02.022

Center Server of the Digital Home on Open System Architecture
——Research and Implementation of Intelligent Component Container

LUO Ke-lu and JIANG Lian-bo

(School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract This paper presents a new method to develop the interconnection and intercommunication control center, which usually plays a central server role in a digital home system. The proposed method adopts several novel technologies, such as bi-directional digital set-top-box, component technology, and design patterns. Moreover, this paper gives an intelligent component container programming model, and implements a central server for digital home system by utilizing such a model on Linux platform. Simulation experiments show that this method can promote software development efficiency, and provide a feasible solution for designing a central server of digital home.

Key words center server; component; container; digital home; set-top-box

随着多媒体家庭产品、宽带网络和信息服务在数字家庭中的广泛应用，市场上出现了各种各样的数字家庭产品，并逐步影响和改变着人们的生活。但这些产品都还是相对孤立的，没有真正实现智能互联、资源共享、协同服务，还需要人为干预。即便很多终端可以独立完成信息处理和上网通信等任务，但仍只是在自己相对孤立的系统内，通过简单共享外界数据实现其有限的功能。究其原因，3C产品通常为受限设备，计算能力、存储能力、输入输出能力有限，特别是通信能力、速率和通信距离都较差，因此需要有一个计算能力和通信能力都较强的设备，通过提高其发射功率及接受灵敏度，从而使所有的家庭3C产品通过其实现互联互通。同时，也可以通过该设备对所有的资源统一管理，针对这些资源形成各种特色服务，而对用户提供统一服务，方便用户的使用，最终形成基于双向数字机顶盒，实现互联互通控制中心的理念，即数字家庭中心服务器。

1 现有产品的不足

就目前国内外的情况而言，完整的面向数字家庭的中心服务器产品尚未真正出现，而以PC为核心的数字家庭多媒体终端解决方案是基于现有的家庭娱乐、智能家电、安防等产品的简单组合考虑的，都没有脱离原有产品的形态。数字家庭中心服务器对双向数字机顶盒做了进一步的改造，除了兼顾与Internet网络的互联互通外，还与hybrid fiber coax(HFC)[1]双向网络实现互联互通，本质上是让数字机顶盒成为一台多功能的家庭信息化服务终端，并承担家庭网关的功能。当前，已经有一些厂商提

收稿日期：2008－09－15； 修回日期：2009－12－27
基金项目：2007年粤港关键领域招标项目(200749811)
作者简介：罗克露(1948－)，女，教授，主要从事嵌入式系统方面的研究.

供了面向不同领域的服务平台，但大部分平台是有如下的共同缺点：(1) 体系结构的设计不够合理，导致系统扩容困难、服务扩展困难[2]。(2) 系统功能较弱，不能提供丰富的增殖服务。(3) 通信能力较差，不能真正实现智能互联、协同服务。针对这些问题，本文的数字家庭中心服务器在体系结构的设计上应充分考虑系统的上述要求，使系统可以很容易地集成新的业务功能和自我扩展。而实现这些可伸缩性业务的关键是在底层采用构件化技术，通过对其相关业务的分析，将每个具有独立功能的模块抽象出来予以构件化。此外，还应提出一种智能化构件容器的编程模型，通过它对构件进行统一管理和维护，从而达到对不断变化的业务的自适应性，实现对业务的即插即用。

2 数字家庭中心服务器

2.1 数字家庭中心服务器总体系架构

软件系统都有其体系结构[3]，但要设计出一个科学合理的体系结构并不容易，尤其是在面临业务繁多、功能复杂、不断有新的需求增加的系统中则更是如此。数字家庭中心服务器是一个功能强大、结构复杂的综合系统，它能提供上网、远程控制、家庭智能设备互联、家庭网关、数字电视接收、直播回放、精彩节目回放、视频点播、远程教育、电子政务、交互游戏、交互体育、证券信息与银行联网的电子商务等增殖功能。其主要功能特点如下：(1) 基于双向有线数字电视接入，同时在IP网络接入方面支持有线/无线等多种接入方式，实现丰富的双向互动业务功能。(2) 支持多种协议，如TCP/IP、数字电视广播DVB协议、IGRS等大多数当前主流的协议，实现智能家电设备的互联互通。(3) 支持软件自动更新与升级，便于维护。其总体系架构如图1所示。

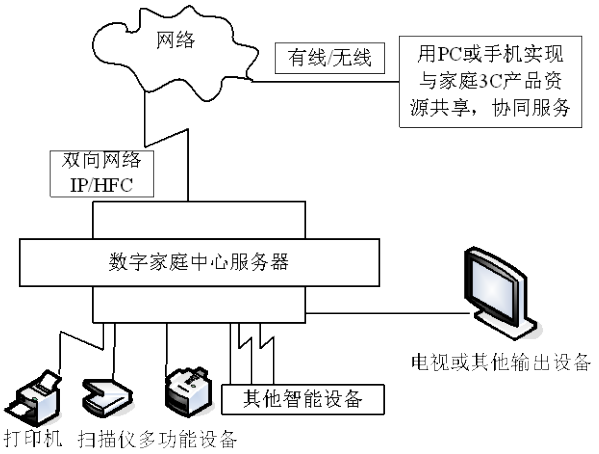

图1 数字家庭中心服务器总体系架构图

2.2 数字家庭中心服务器内部结构模型

数字家庭中心服务器是一个基于双向数字机顶盒的复杂嵌入式系统，它具有开放式的体系架构，并作为一个客户端的服务软件而运行，是一个集网络、数字电视、信息处理、音视频、软硬件设计等多种技术于一体的新型产品。它的内部结构模型主要由以下几个部分组成：(1) QOS安全管理，主要实现家庭网关的功能，通过身份认证的用户可以使用手机或者PC实现对家庭内部信息设备的控制。(2) 业务中心，主要实现数字电视接收、视频点播、游戏、电子商务等增殖服务。(3) 3C产品控制中心，主要负责家庭内部智能设备的控制。在家庭内部网络中，以闪联协议(IGRS)为基础实现各个信息家电的自动发现、智能互联、协同服务，同时还支持与最多8台PC机的互联，具有家庭内部动态组网功能。图2给出了数字家庭中心服务器内部结构模型。

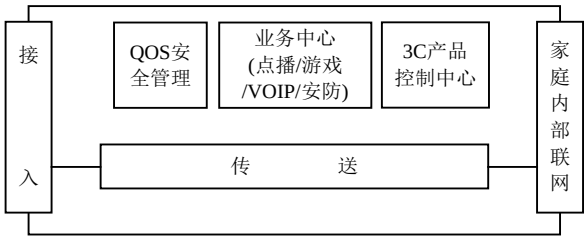

图2 数字家庭中心服务器内部结构模型

2.3 开放式数字家庭中心服务器应用架构

为了让系统的移植性好、扩展性强、可靠性高，数字家庭中心服务器应用系统采用开放式系统[4]架构。服务表示层(控制/管理)通过对不同3C产品服务的整合，能够为上层应用提供统一的、有价值的综合服务。中间件层以服务理念为核心，屏蔽了不同的互联互通协议，具有面向业务的开放性、伸缩性和可移植性，能够实现不同设备的智能互联、资源定位和协同服务，向上层提供统一的服务描述。智能化构件容器能够被用作部署其他的构件，并对可伸缩业务具有较强的自适应性，是数字家庭中心服务器系统适应可变业务的关键。图3给出了开放式数字家庭中心服务器应用系统架构。

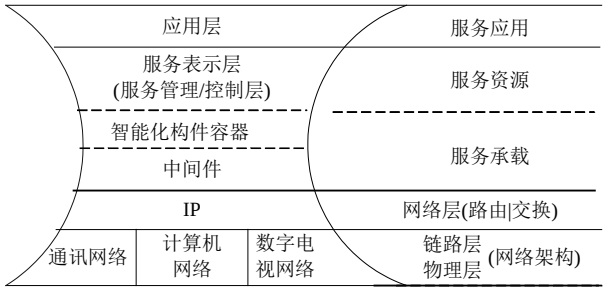

图3 数字家庭中心服务器应用系统架构图

3 数字家庭中心服务器智能化构件容器

3.1 智能化构件容器编程模型描述

通过对数字电视业务、智能家电等的分析与抽象[5]，本文在数字家庭中心服务器里引入了构件化思想，并结合软件体系结构中的设计模式，提出了面向该领域的智能化构件容器编程模型。该模型吸收和借鉴了COM[6]和EJB[7]中的思想，如图4所示。构件[8]是指一个可复用的部署单元，并且它只有通过接口才能够被访问。接口指定了该构件的入口点，是构件内一组方法的集合，外界通过接口引用或接口指针调用构件内的方法。而构件容器则是存放构件的器皿，并为构件运行时提供环境支持[9]和一些必要的服务，如构件的创建、移除、生命周期管理及资源管理等，使嵌入式平台下的开发变得更加容易和简洁，使开发者从繁琐的工作中解脱出来，从而有更多的精力专注于业务逻辑的开发，极大地缩短了产品的开发周期。智能化构件容器主要包括外部接口、内部接口、构件和构件管理器部份。其中，外部接口提供了一个客户与构件容器进行交互的桥梁，提供构件生命周期服务入口，客户能够使用该接口创建、删除、查找一个处于活动状态的构件实例，它由构件容器提供并由客户使用。内部接口提供了构件的应用方法入口，它相对于客户应用而言，暴露了与应用相关的所有接口，还被构件容器用于管理和控制这些对象。构件管理器由构件容器创建，构件容器通过它对各构件的生命周期进行管理，每个构件也可以进行自我控制和管理；当一个新的构件被创建时，就会加入到构件管理器，该构件的相关信息被记录；反之，当某个构件不需要使用时，构件容器首先释放该构件，然后从构件管理器中删除相应信息。

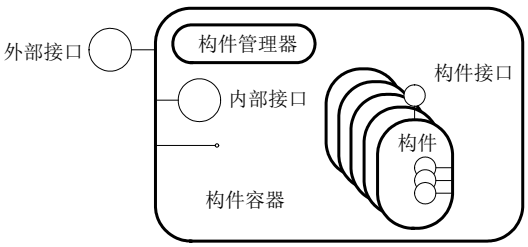

图4　智能化构件容器编程模型

3.2 智能化构件容器编程模型接口协议

一个构件的接口可以被看作是其入口点的规范[10]。构件都是基于二进制复用的，因此，在进行开发之前，必须要为其确定一个统一固定的二进制结构，让该结构成为每一个构件在内存中运行时的存在形式，即成为该结构在内存中的映射，以便于在不同的编译环境下都能让其产生等价形式的二进制代码，达到更大范围内的重用。要达到上述目的，就必须遵守以下的协议：(1) 构件接口要与实现相分离。每一个构件接口都是一个纯虚基类，在该纯虚基类中只定义抽象的方法，不能定义变量。(2) 只能有一个源的纯虚基类(简称ISource接口)，其他所有的构件接口都必须从ISource接口直接或间接地继承而来，即可以从它派生的接口再进行继承，产生新的接口，但不能从它派生的两个以上的接口同时继承，即只允许单继承，不能多继承[6]。(3) 所有从ISource接口继承而来的构件接口都必须以ISource接口中的抽象方法为其的前几个方法，在这些抽象方法之后才能根据需要自定义其他抽象方法。并且继承而来的构件接口的实现类中都必须实现ISource接口中的所有抽象方法。(4) 各类构件要能被构件容器所管理，必须将其部署在该构件容器内。构件容器自身也是一种特殊的构件，它为各类构件提供服务，可以定时自动对构件管理器进行搜索，及时释放不再使用的构件，具有一定程度的智能化。

3.3 智能化构件容器编程模型的实现方法描述

本文根据该编程模型采用C/C++语言，在Linux环境下实现了智能化构件容器，它是采用共享库的方式实现的，应用程序可以通过Linux系统提供的动态装载方法加载共享库，然后调用共享库中的相关接口。在C/C++程序设计中，指针的使用非常频繁、灵活且执行效率较高，为此采用通过对指针跟踪的策略从而达到对构件的管理和生命周期的控制。首先，必须定义一个ISource接口，它是一个纯虚基类，其中包括接口查询方法、指针跟踪方法等基本的抽象方法。其定义形式如下：

```
class ISource
{
    Public:
    //IID表示接口的唯一标识符，ppvIntfc用于存放查询iid的接口指针的地址；
    virtual void LookUp(IID iid, void** ppvIntfc)= 0；
    virtual unsigned long AddRef(void)= 0；//指针个数加1
    virtual unsigned long DecRef(void)=0；//指针个数减1
};
```

其次，再定义一个构件容器接口IContainer，它从ISource继承而来，并在ISource中的抽象方法之后定义自己的抽象方法，如构件创建、构件移除、构件管理器维护等一系列方法。其定义形式如下：

```
class IContainer：public ISource
{
    public：
    //创建新构件或接口，CLSID表示构件的唯一标识符
    virtual void CreateComponent(CLSID clsid, IID iid, void** ppvIntfc)=0
    virtual bool RemoveComponent(CLSID clsid)=0；//移除指定的构件
    …//其他抽象方法
}
```

对构件管理器的处理采用定时触发机制，每隔一段时间对它进行一次扫描，以确定需要被移除的构件，提高资源的利用效率。再次，智能化构件容器从构件容器接口(IContainer)继承，并将该接口中所有的抽象方法实现。智能化构件容器实现类的定义如下：

```
class CintelligentContainer：public IContainer
{
    public：
    //ISource接口中的方法
    virtual void LookUp(IID iid, void** ppvIntfc)；
    virtual unsigned long AddRef(void)；
    virtual unsigned long DecRef(void)；
    //IContainer接口中的方法
    virtual void CreateComponent(CLSID clsid, IID iid, void** ppvIntfc)；//具体实现从略
    virtual bool RemoveComponent(CLSID clsid)；
    …//其他方法
}；
```

最后，在智能化构件容器的实现类CIntelligentContainer源文件CIntelligentContainer.c中提供一个外部方法以便于动态装载。该方法的具体实现如下：

```
extern “C” ISource* CreateContainer(void)
{
    ISource*pIS=static_cast 〈 ISource* 〉 (new CIntelligent Container)；
    if (pIS!= NULL)
    {
        pIS->AddRef()；
    }
    return pIS；
}
```

在Linux中，可以通过g++-fPIC-shared-olibfilename.so filename.c的形式对C/C++源文件进行编译，将它们做成共享库，或者编写Makefile的方式对项目中多个源文件进行编译，编成共享库；并将智能化构件容器的实现类CIntelligentContainer编成共享库libcontainer.so。而用户可以通过Linux系统提供的动态装载方法调用这些共享库，其示例代码片段如下：

```
typedef ISource* (*PISOURCE)(void)；
PISOURCE CreateContainer;  //存放共享库中函数入口地址
void* handle；
char* errorText；
//打开名为libcontainer.so的共享库
    if((handle=dlopen(“./libcontainer.so”，RTLD_LAZY) ) == NULL)
{
    //错误信息处理…
}
// 取得共享库libcontainer.so中函数CreateContainer的地址
    CreateContainer=(PISOURCE)dlsym(handle, “CreateContainer”)；
if ((errorText=dlerror())!=NULL)
{
    //错误信息处理…
}
ISource* pIS=CreateContainer(); //调用共享库中的CreateContainer方法
…//检查pIS指针，代码略去
IContainer* pContainer=NULL；
pIS->LookUp(IID_Icontainer，(void**)&pContainer)；//查询是否支持该接口
…//检查pContainer指针，代码此处略去
//创建新的构件或构件中的某个接口
pContainer->CreateComponent(args…)；
…
dlclose(handle)；//关闭共享库libcontainer.so
```

从上述实现方法中可以看出，本文编程模型主要具有以下的特点：(1) 基于二进制复用的，能够实

现即插即用。(2) 具有较强的自适应性，且易于更新升级。(3) 易于移植和扩容。

4 结 论

为了适应全球数字电视、家庭娱乐、数字家庭等产业的发展，介绍了在客户端建立功能强大的数字家庭中心服务器；并以构件技术为基础，提出面向领域的智能化构件容器编程模型，同时采用了该编程模型在与企业合作的项目中予以实现。

参 考 文 献

[1] 肖宇玲, 胡蔚星. HFC接入网关键技术分析[J]. 中国有线电视, 2004, (9): 64-69.
XIAO Yu-ling, HU Wei-xing. Technique analysis of HFC access network[J]. China Cable Television, 2004, (9): 64-69.

[2] 李文波, 李启炎. 基于开放体系结构的智能型UMS系统[J]. 计算机工程, 2004, 30(4): 96-98.
LI Wen-bo, LI Qi-yan. An intelligent UMS system based on open architecture[J]. Computer Engineering, 2004, 30(4): 96-98.

[3] GARLAN D, PERRY D. Introduction to the special issue on software architecture[J]. IEEE Transactions on Software Engineering, 1995, 21(4): 269-274.

[4] IEEE. IEEE Std 1003.0-1995, IEEE Guide to the POSIX® Open System Environment(OSE)[S].1995.

[5] 施俞行, 高　峰, 罗克露, 等. 一种适用于智能家电嵌入式软件的框架构件模型[J]. 电子科技大学学报, 2006, 35(5): 806-809.
SHI Yu-hang, GAO Feng, LUO Ke-lu, et al. A frame component model approach for embedded software of intelligent household appliances[J]. Journal of University of Electronic Science and Technology of China, 2006, 35(5): 806-809.

[6] Microsoft Corporation and Digital Equipment Corporation. The component object model specification[S]. Draft Version 0.9, 1995.

[7] LINDA G, DEMICHIEL L. ümit Yalcinalp, et al. Enterprise JavaBeansTM specification[S]. Version 2.0, Sun Microsystems, Inc, 2001.

[8] CRNKOVIC I, LARSSON M. Building realiable component-based software systems[M]. [S.l.]: Artech House Computing Library, 2002.

[9] 古幼鹏, 桑　楠, 熊光泽. 嵌入式软件平台的构件化模型研究[J]. 计算机科学, 2005, 32(10): 216-218.
GU You-Peng, SANG Nan, XIONG Guang-ze. Research on component-based model of embedded software platform[J]. Computer Science, 2005, 32(10): 216-218.

[10] SZYPERSKI C. Component software: beyond object-oriented programming[M]. New York, USA: Addison-Wesley/ACM Press, 1998.

编　辑　黄　莘

(上接第254页)

[7] 闵　军, 张海呈, 朱桂斌. 自组网可靠性评价方法[J]. 电子科技大学学报, 2008, 37(3): 436-438.
MIN Jun, ZHANG Hai-cheng, ZHU Gui-bin. Reliability evaluating method of ad hoc network[J]. Journal of University of Electronic Science and Technology of China, 2008, 37(3): 436-438.

[8] 蔡皖东. 网络存储中的数据容错与容灾技术研究[D]. 西安: 西北工业大学, 2006.
CAI Wan-dong. Study on technology of data error tolerance and disaster tolerance in network storage[D]. Xiàn: Northwestern Polytechnical University, 2006.

[9] VAZIRI M, LYNCH N, JEANNETTE M W. Proving correctness of a controller algorithm for the RAID level 5 system[J]. Computer Science, 2007, 5(6): 87-95.

[10] HETRICK W, MOORE J, KRUEGER C. RAID controller firmware product line, LSI logic-engenio storage group [C]//10th International Software Product Line Conference. Baltimore, Maryland, USA: [s.n.], 2006: 21-24.

[12] PENNINGTON A, STRUNK J, GRIFFIN J, et al. Storage-based intrusion detection: watching storage activity for suspicious behavior[C]. Proceedings of the 12th USENIX Security Symposium. Washington, D.C.: [s.n.], 2003: 14-18.

[13] STENSRUD E, MYRTVEIT I. Identifying high performance ERP projects[J]. IEEE Transaction on Software Engineering, 2003, 29(5): 387-416.

[14] 姜宁康, 时成阁. 网络存储导论[M]. 北京: 清华大学出版社, 2007: 113-117.
JIANG Nin-kang, SHI Cheng-ge. Network storage generality[M]. Beijing: Tsinghua University Press, 2007: 113-117.

编　辑　蒋　晓