摘 要

信息技术的发展，在数字家庭中逐渐形成了三大信息孤岛：以手机为代表的通信设备，以数字电视为代表的信息设备，以及以电脑为代表的网络设备。目前对于如何使这三大信息孤岛无缝的互联是一个研究的热点，本文阐述的 IGRS 便是其中的一个解决方案。

IGRS（Intelligent Grouping & Resource Sharing）是为了实现信息设备智能互联、资源共享、协同服务而制定的基础协议。支持 IGRS 协议的信息设备能够在网络上自动的发现对方，并自动的进行设备信息的交互，满足了一定条件的设备间就能互相共享对方的资源。

在 IGRS 中所有的资源都是以服务的形式呈现的，因此共享资源的行为称作服务调用。在服务调用之前，客户端和服务端必须建立起相应的会话。建立会话的目的是为了让服务端能够对调用自己的客户进行身份的鉴别，只有通过鉴别的客户才能顺利的调用自己的服务。鉴别的方式由服务端规定，根据不同的应用场合和安全要求可以有不同的鉴别机制：不鉴别、基于对称密钥的鉴别机制、基于公钥的鉴别机制、基于可信赖第三方的鉴别机制。根据不同的鉴别机制，建立起的会话的安全机制也是不同的。

一个 IGRS 设备可以有多个服务，每个服务用唯一的服务序列号标识。对于同一个设备上的不同的服务，各自的鉴别机制可以不一样。在实际应用中，一个功能的实现需要调用多个服务，因此一个客户需要与设备上的若干个服务建立起会话才能实现正常的应用。

本文总结笔者参与相关科研项目的心得，主要论述 IGRS 基础协议栈的实现原理，尤其对 IGRS 会话机制的实现原理作了详细的阐述。最后，论文总结了研究和实现的成果，并指出了下一步的研究方向。

关 键 字：IGRS，服务，会话

ABSTRACT

With the development of Information Technology, three information islands are formed gradually and respectively in digital home: communication devices represented by mobile phone, information devices represented by digital TV, and Internet devices represented by PC. Now it is a research hotspot to make the devices communicate smoothly among three islands, and IGRS (Intelligent Grouping & Resource Sharing) expatiated in this article is a solution to this problem.

IGRS is a basic protocol, which is constituted to make intelligent grouping & resource sharing come true among information devices. Information devices based on IGRS will discover and exchange devices information each other automatically in the network, also they can share the resources when certain qualifications are satisfied.

All the resources are presented as services in IGRS, so sharing resource is called service invoke. Before invoking service, session must be created between client and server. Server can authenticate the client by session, and only the client who passed the authentication can invoke the service. The method of authentication is decided by server, and there are several different methods which will be chosen according different application situations and security requirements: No authentication、authentication mechanism based on symmetry key、authentication mechanism based on public key、and authentication mechanism based on 3rd trusted devices. The security of session is different because of different authentication mechanism.

There are several services in one IGRS device, and services are distinguished by service ID. The authentication mechanism of the different services can be different in the same device. In application, more than one service should be invoked to accomplish one function, so one client will create sessions with several services in the same device to accomplish the application.

The dissertation summarizes the author's achievements and experiences gained in the related research project. This dissertation mainly discussed the realization theory of IGRS stack, especially the IGRS session theory. Finally, the dissertation summarizes the outcome of the researches, and points out the direction of further study.

Keywords: IGRS, Service, Session

术语和缩略语

DLNA	数字生活网络联盟	Digital Living Network Alliance
DHWG	数字家庭工作组	Digital Home Working Group
DHCP	动态主机分配协议	Dynamic Host Configuration Protocol
DNS	域名服务器	Domain Name Server
GENA	通用事件通告体系结构	General Event Notification Architecture
HTTP	超文本传输协议	Hypertext Transport Protocol
IGRS	智能互联、资源共享与协同服务	Intelligent Grouping & Resource Sharing
IIOP	互联网内部对象请求代理协议	Internet Inter-ORB Protocol
NGN	下一代网络	Next Generation Networking
RMI	远程方法调用	Remote Method Invocation
SOAP	简单对象访问协议	Simple Object Access Protocol
SSDP	简单服务发现协议	Simple Service Discovery Protocol
TCP	传输控制协议	Transport Control Protocol
UDP	用户数据报协议	User Datagram Protocol
URL	统一资源定位	Universal Resource Locator
URI	统一资源标识符	Universal Resource Identifier
UPnP	通用即插即用	Universal Plug&Play
WSDL	Web服务描述语言	Web Service Discription Language
XML	可扩展标识语言	Extensible Markup Language
DeviceId	IGRS设备标识符，一个设备的ID全球唯一。固定格式为“urn：IGRS：Device：DeviceId：xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx”，其中x为16进制数。	Device Identifier

ServiceId	IGRS服务标识符（32位无符号整数）	Service Identifier
ClientId	IGRS客户标识符（32位无符号整数）	Client Identifier

南京邮电大学学位论文独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得南京邮电大学或其它教育机构的学位或证书而使用过的材料。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了谢意。

研究生签名：夏宏春 日期：2007.7.23

南京邮电大学学位论文使用授权声明

南京邮电大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权南京邮电大学研究生部办理。

研究生签名：夏宏春 导师签名：  日期：2007.7.23

第一章 绪论

1.1 引言

随着信息技术的不断发展，在我们的家庭网络中逐渐形成了三大信息孤岛，分别是以移动电话为代表的移动通信终端设备，以数字电视为代表的信息家电设备，以个人电脑为代表的网络通信设备。处于各个孤岛内的设备之间的连接与通信技术已经发展到了相当成熟的地步，因此打破信息孤岛，使得不同孤岛之间的设备能够方便的连接与通信成了当今研究的一个热点。

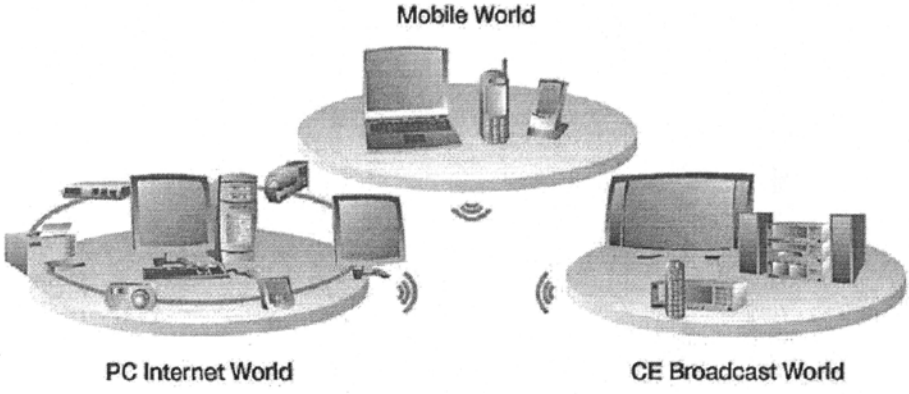

图 1-1 数字家庭中的三大信息孤岛[1]

数字家庭网络是 IT、信息家电、通信高度融合的具体产物。随着信息化的普及，家庭中的电脑、家电、通信终端越来越多，不同设备之间信息传递和协同工作也越来越频繁，设备间如何方便、智能的互联和协同工作已经日益成为人们关注的焦点，但由于缺乏统一的信息格式、接口协议，导致难于互通共享。加上设备越来越多，从而使用管理也越来越麻烦。如何更有效地发挥这些设备的作用，提高居家生活的智能化和方便性，已成为现代人生活乃至办公的重要问题。人们自然地产生了将各种电脑、家电之间合理地互连互通起来的强烈需求，于是，数字家庭网络的概念和设备便应运而生了。从产业发展的潮流看，通信、信息、娱乐等技术迅速成熟，3C 融合已成为总的趋势，信息资源在个人终端上得以整合，个人用户可以直接面对一个虚拟的信息平台进行操作，设备和信息之间实现无缝的连接，所有这些使得“数字家庭网络”越来越热。ITU2T 已经将家庭网络纳入到了 NGN 网络的一部分。

从结构和功能上看，数字家庭网络通过家庭网关将公共网络功能和应用延伸到家庭，并以有线网络或无线网络，连接各种信息终端(如电视、电脑等)，提供集成的话音、数据、多媒体、控制和管理等功能，达到信息在家庭内部终端之间及其与外部公网的充分流通和共享。数字家庭网络的组成上可以分为：家庭网关；各种信息终端设备和智能家电设备。家庭智能联网环境的构筑，即通过家庭内部有线或无线方式的互联技术，将数字家庭内各类终端互联并与家庭网关相连。其中各种信息终端互连成网络为家庭主干网，采用较复杂的协议；各种智能电器(如微波炉、空调、冰箱、三表等)连接成监控子网；主网和子网之间由子网网关连接。

数字家庭网络的业务定位于三大方面：家庭影音娱乐、电信及广播业务、控制和监视功能。具体业务和应用有：VOD点播、娱乐游戏、家庭内部各种信息终端的高速信息共享、对各种家用电器设备的远程控制、调节和监测、实现防盗报警器、自动报警、实时监控家庭安全等。[2]

1.2 主流标准简介

数字家庭涉及的各种业务和技术非常繁杂。由于通用的网络和终端技术已经非常成熟，所以数字家庭网的核心技术主要体现在网络协议和接口上。近年来，国内外许多大公司提出了相应的解决方案，各国正努力研制适合于本国国情的智能家居系统，已逐渐形成了一些相关的标准。家庭网络物理层的接口标准和数据传输协议是国内外学术机构和厂商普遍关注的问题。

目前从事数字家庭标准化的组织非常多，国外有DLNA、ITU2T、UPnP、UOPF、ECHoNET、Lonworks、CEBus、Jini、OSGi等，中国有“闪联”(IGRS)和“e家佳”。每个标准化组织所涉及的情况不尽相同，本节首先叙述国际上最流行的DLNA标准，接着谈谈国内的“e家佳”标准，最后介绍本文的重点--IGRS标准。

1.2.1 DLNA 标准介绍

数字生活网络联盟（DLNA）创立于 2003 年，由 Intel、Sony、Microsoft 等十七家国际大型 IT 公司创建，其前身名为数字家庭工作组（DHWG）。DLNA 的成员拥有共同的远景目标，即在家庭内外建立一个集中管理个人电脑（PC）、家电（CE）和移动电子设备的互操作性网络，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。该联盟建立并维护着一个基于开放式工业标准的互操作性平台，在制造商使用时可通过

有线或无线网络共享媒体内容。迄今为止，已有来自全世界超过200家不同行业的公司加入DLNA，为实现这一远景目标投入了宝贵的时间和资源。

DLNA的基础是UPnP协议，UPnP是针对智能家电、无线设备以及各种外观尺寸的个人电脑的普遍对等（peer-to-peer）网络连接而设计的一种架构。它旨在为家庭、小型企业、公共场所中或连接到互联网的ad-hoc网或未管理网络提供易于使用、灵活且基于标准的连接。UPnP是一个充分利用TCP/IP和Web技术的分布式开放型网络体系结构，除能够在家中、办公室和公共场所联网设备之间的完整控制和数据传输之外，还可建立无缝紧密的连接网络。

UPnP不仅仅只是即插即用外设模式的简单扩展。它设计用于支持零配置、“不可见”联网，以及对众多厂商的广泛设备类型的自动发现。这就意味着，一台设备能够动态加入一个网络，获取一个IP地址，通报其功能，以及了解其它设备的存在和功能。DHCP和DNS服务器为可选服务器，仅当它们在网上存在时可以使用。最后，设备能够顺利地自动离线，而不会造成任何不期望的影响。

UPnP充分利用了包括IP、TCP、UDP、HTTP和XML在内的互联网组件。正如互联网那样，合约基于陈述性的有线协议，以XML来表达，并通过HTTP进行传输。IP网间协议凭借其以下已经被验证的能力而成为UPnP的一个有力选择：跨越不同的物理媒体、支持实际的多厂商互操作，以及实现与互联网、大量家庭和办公室内联网的协作融合等等。UPnP的设计明确用于支持这些环境。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。

UPnP网络共有五步，分别是发现、描述、控制、触发和展示。针对这五步，UPnP采用如图1-2的协议栈架构：

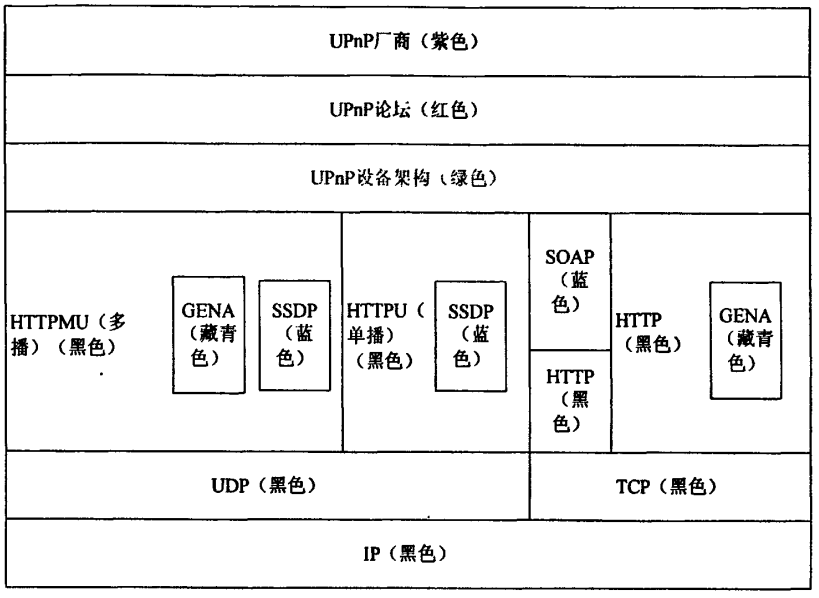

图 1-2 UPnP 协议栈架构图

在最高一层，消息在逻辑上仅仅包含关于厂商设备的UPnP厂商特定信息。移至下一层协议栈后，厂商内容由UPnP论坛工作委员会定义之信息提供。来自以上各层的消息存储在UPnP特定协议中。之后以上消息通过采用简单服务发现协议（SSDP）、通用事件通知架构（GENA）和简单对象访问协议（SOAP）来进行格式化。然后消息通过运行于UDP上的多播或单播类HTTP，或是运行于TCP上的标准HTTP进行传输。最终，以上所有消息均通过IP进行传输。

UPnP网络互连的基础是IP寻址。每台设备均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DHCP服务器可以使用，即网络处于管理状态，则设备必须采用分配给它的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来获取一个地址。简言之，Auto IP说明了一台设备如何从一组保留地址中智能地选出一个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移动。如果设备在DHCP交易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操作中采用该名称；否则即应采用其IP地址。

如果获取了一个IP地址，则UPnP网络的第1步是发现。在将一个设备添加到网络上之后，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，当一个控制点被添加到网络后，UPnP发现协议允许该控制点在网上搜索感兴趣的设备。两种情况下的根本信息交换均为一个发现消息，包含有关该设备或其服务之一的一些基础信息（例如其类型、标识符和指向更详细信息的一个指针）。UPnP发现协议基于简单服务发现

协议（SSDP）。

UPnP网络中的第2步是描述。控制点在发现一个设备之后仍然对其知之甚少。为了使控制点了解到更多关于设备及其能力的信息或与设备进行交互，则控制点必须取得来自该设备在发现消息中所提供的URL的设备描述。设备可能包含其它逻辑设备，以及功能单元或服务。对于设备的UPnP描述通过XML表达，并包括诸如模型名称和号码、序列号、制造商名称和厂商专门网站URL等专门针对厂商的制造商信息。该描述还包括一列任意的嵌入式设备或服务，以及用于控制、事件触发和展示的URL。对于每项服务，此描述均包括一系列命令或动作，而服务（参数或变量）对于每个动作做出响应；针对服务的描述还包括一系列变量；这些变量模型化服务在运行时的状态，并通过数据类型、范围和事件特征进行描述。

UPnP网络中的第3步是控制。当一个控制点取得设备描述后，该控制点可将动作发送到一个设备的服务。为此，控制点将一条适当的控制消息发至服务的控制URL（在设备描述中提供）。控制消息同样利用简单对象访问协议（SOAP）通过XML来表达。类似于功能调用，该服务针对控制消息返回了所有的专门动作取值。动作的效果可以通过描述服务运行时状态的变量进行描述。

UPnP网络的第4步是事件触发。针对服务的UPnP描述包括一个服务响应的动作列表，以及一个对服务器运行时状态进行展示的变量列表。在这些变量变更时服务会发布更新，一个控制点可以预订接收此信息。服务通过发送事件消息来发布更新。事件消息包含一个或多个状态变量名和这些变量的当前值。这些消息同样通过XML来表达，并采用通用事件通知架构（GENA）格式。当控制点首次预定时，会发送一个特殊的初始事件消息；此事件消息包含所有事件变量的名称和值，并允许订阅者对服务状态模式进行初始化。为了支持拥有多个控制点的环境，事件触发设计用于将任何动作的效果通知所有控制点。因此，所有订阅者均会收到全部的事件消息。订阅者收到关于所有已变更事件变量的事件消息，此事件消息无论状态变量为何改变都被发送（由于响应一个要求动作，或由于服务建模状态的变更）。

UPnP网络中的第5步是展示。如果设备有用于展示的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，并且根据页面的功能，支持用户控制设备和/或浏览设备状态。每一项完成的程度取决于展示页面和设备的具体功能。[3]

1.2.2 “e家佳”标准介绍

家庭网络标准产业联盟—ITopHome（e家佳）是由海尔集团、清华同方、中国网通、上海广电集团、春兰集团、长城集团、上海贝岭等七家公司发起组建，意在推广家庭网络系统标准和平台产业化，进而形成规模化的产品和市场。该联盟以家庭网络系统为中心，包括了电子、家电、通讯、计算机、网络运营等多领域企业，共同探索家庭网络商业运作模式，为家庭网络技术发展方向及产业的可持续发展提供产业环境。

联盟今后将加强市场需求的调研、分析，搭建家庭网络系统技术平台，推广家庭网络产品和服务，促进家庭网络系统的产业化。同时，家庭网络产业的发展需要有家庭网络标准来规范产品和服务，以标准化规范产业化，并以产业化推动标准化，完善相关技术标准是联盟的另一项重要任务。

一个典型的“e家佳”的网络设备（家庭网关）的架构如图1-3：

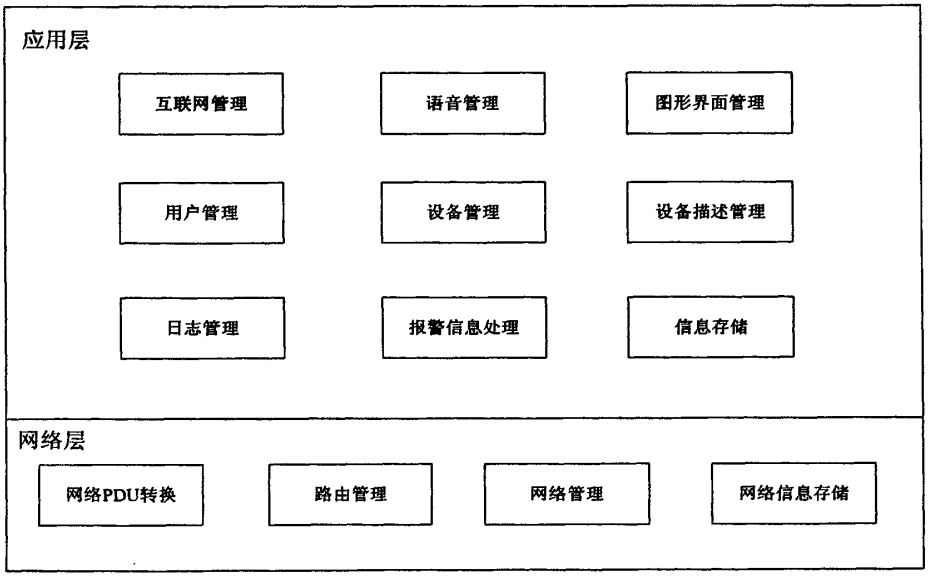

图1-3 “e家佳”家庭网关结构图

“e家佳”联盟采用一种开放和灵活的设备描述方法(DeviceFile)，即只规定设备功能及其参数、控制命令和用户界面描述方法，对具体功能和参数并不进行具体规定，这样就可以为终端设备厂商带来更多的灵活性和发展空间。每个家庭网络设备都有一份描述自身功能和命令的设备描述文件，在设备描述文件中不仅包含了设备的厂家信息，还包含了对设备的命令图标索引、操作描述、每个操作对应的命令、设备的状态回码等信息。通过设备描述文件可以了解设备具有哪些功能，发送命令实现某一功能，还可以从状态回码中得到设备的当前状态。

在“e家佳”中，特地将家庭网络根据网速进行了分类，根据数据传输率的不同，家庭网络可以分为3类：高数据传输率短程无线家庭网络(UWB、802.11n，传输率大于100Mb/s)，主要用于娱乐网络；中等数据传输率短程无线家庭网络(802.11a/b/g，传输率小于50Mb/s)，主要用于数据(文件)传输网络；低数据传输率短程无线家庭网络(802.15.4、蓝牙等，传输率小于250kb/s)，主要用于监测和控制网络。[4]

1.2.3 IGRS标准介绍

信息设备智能互联与资源共享协议（简称IGRS）自2003年7月开始由闪联标准工作组负责制定。其标准1.0版本已于2005年6月被正式颁布为国家行业推荐性标准，成为我国3C协同领域的第一个国家标准。此外，基于闪联标准的各项开发工具和测试认证工具也已经基本完成，并在逐渐完善和更新中。2005年8月30日，由信息产业部电子工业标准化研究所信息处理产品标准符合性检测中心牵头，闪联工作组成员共同参与，历时一年半时间开发完成的闪联标准符合性检测系统通过了信息产业部组织的专家鉴定。闪联标准工作组和闪联信息技术工程中心曾先后承接了国家发改委的重大项目“信息设备资源共享协同服务”标准研究制定工作等项目，具有较高的科研水平和项目管理水平。

闪联标准为闪联应用提供统一的网络资源发现、使用和管理机制，它由三部分构成：闪联基础协议、闪联智能应用框架、闪联基础应用。IGRS标准1.0版目前包括的是两个基础部分：基础协议标准和智能应用框架标准。

闪联设备是对现实世界中各种信息终端设备的一种抽象，闪联设备是其所拥有的共享资源的容器和管理者。应用程序通过闪联协议栈API与共享资源管理器交互实现对由闪联设备构成的网络中的共享资源的使用与管理。

闪联设备所提供的共享资源可以分为两类，一类是设备自身所固有的可共享资源，如计算资源、存储资源等，该类资源在闪联设备中以服务对象形式接受共享资源管理器的管理；一类是应用程序开发者通过开发可向其他应用程序提供某种形式的服务，如媒体播放服务、文件打印服务等，该类资源在闪联设备中以服务存根形式接受共享资源管理器的管理。

需要使用闪联设备上的共享资源的应用程序通过客户代理形式接受共享资源管理器的管理，同时使用资源管理器所提供的API实现对闪联设备共享资源的使用和管理。不同的闪联设备上的共享资源管理器之间通过与具体网络传输协议无关的通用闪联设备交互协议实现设备访问与控制、服务访问与控制、服务数据分发、事件订阅与通知等

功能。

通用闪联设备交互协议可以通过协议映射规范映射到不同的传输协议上，如基于TCP/UDP，HTTP，IIOP，RMI等网络协议的闪联设备交互协议。

1.3 论文的章节安排

本文源于广东省关键领域重点突破项目“数字家庭公共技术支撑平台”中的IGRS子项目。文章共分为五章：

第一章中介绍了数字家庭技术的背景与现状，重点介绍了几个主流的技术标准，DLNA，“e家佳”和IGRS。

第二章对IGRS的整个系统架构做了叙述，包括IGRS应用架构、IGRS特色应用，IGRS基础协议三部分；对于IGRS基础协议，还将其分成各个模块进行详细的介绍与说明。

第三章详细的说明了IGRS基础协议的实现原理与方法。从程序实现的角度，按照各个线程的方式对IGRS基础协议的实现作了详细的阐述。

第四章对IGRS会话机制的实现做了详细的论述。会话包括一般会话和同一设备组内设备间的会话，该章节分两大部分分别对这两种会话的实现方式进行了论述。

第五章则对本文的取得的成果做了说明，并且提出了课题今后的发展方向。

第二章　IGRS 基础协议的原理

IGRS 是为了实现信息设备互联、资源共享、协同服务而制定的基础协议。它的目的是在企业、公共场所、个人以及家庭所涉及的信息设备互连时，通过遵循共同的资源描述及功能服务接口标准，使设备能够有效实现资源开放及协同服务，提高设备间的功能的互操作性，为用户提供全新的应用体验。

2.1　IGRS 系统介绍

从 OSI 的七层架构模型看，IGRS 是应用层的协议。它支持各种设备通过有线局域网、无线局域网、蓝牙等网络连接，传输与网络协议基于 TCP/IP 协议簇，设备交互消息框架基于 HTTP/1.1，设备发现与资源共享平台基于 IGRS 基础协议，设备协同服务平台基于 IGRS 应用框架。图 2-1 是 IGRS 系统架构示意图：

图 2-1　IGRS 体系结构[①]

2.1.1　IGRS 基础协议

IGRS 基础协议确定了 IGRS 设备间相互发现及自动共享的机制，包括设备发现机制、设备管道创建机制、设备组管理机制、服务发现机制、会话管理机制及服务访问机

[①] 见参考文献[5]第 12 页。

制。各个机制间的层次关系如图 2-2：

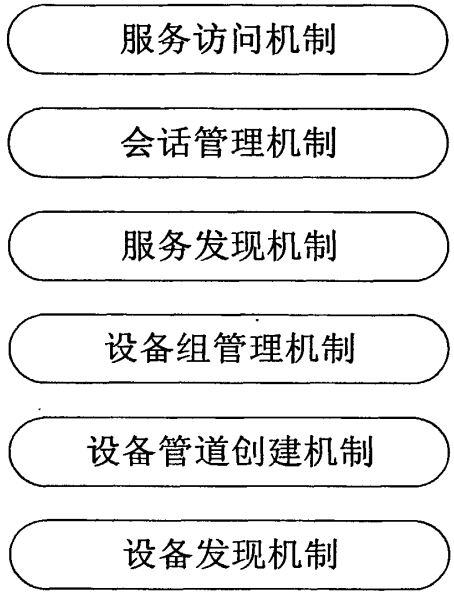

图 2-2 IGRS 基础协议[②]

IGRS 设备交互模式描述了一个 IGRS 设备从加入网络，发现其它 IGRS 设备，加入某个 IGRS 设备组，发现设备组内其它 IGRS 设备上的服务并进行服务调用，到最后退出网络的全部过程，一个典型的 IGRS 设备交互过程如图 2-3：

[②] 见参考文献[5]第 12 页。

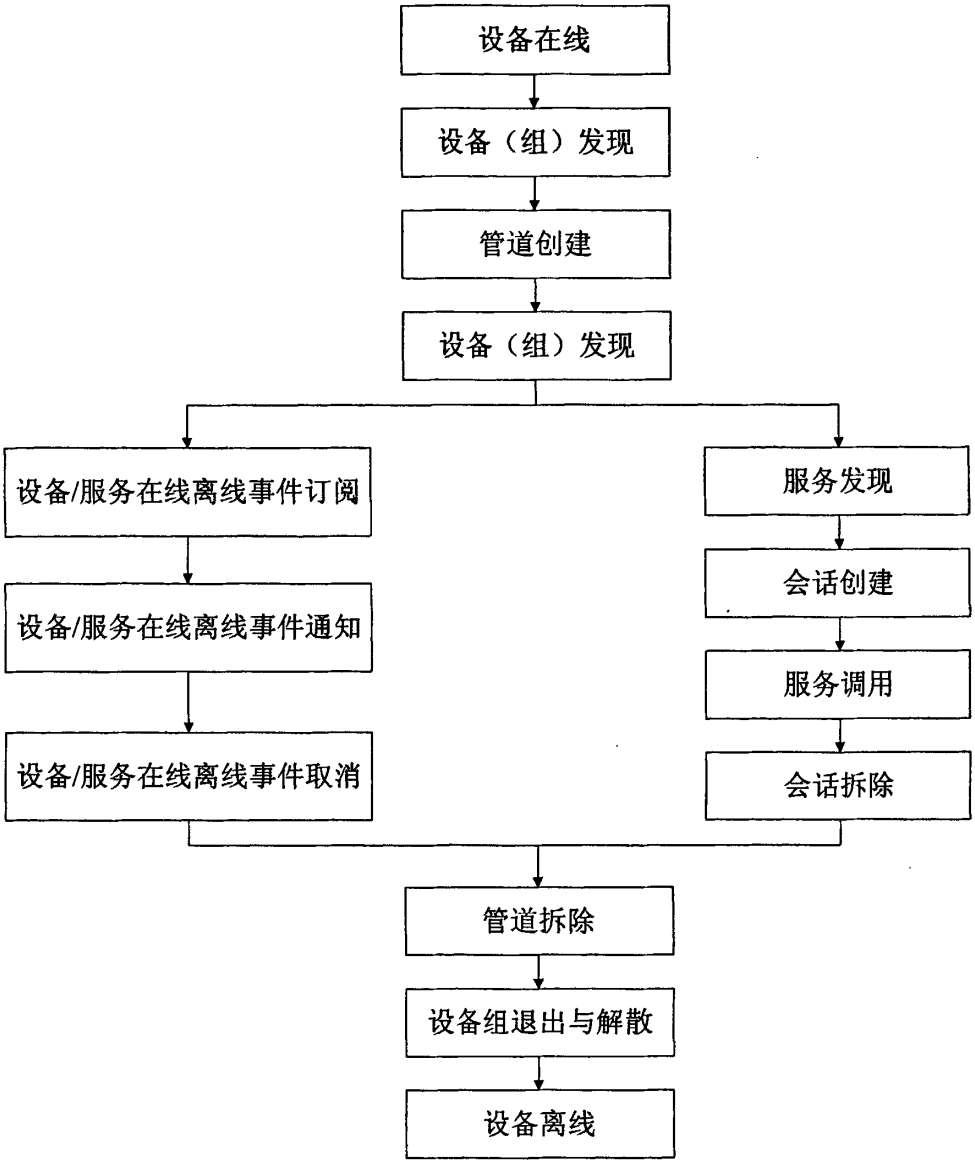

图 2-3 IGRS 设备交互示意图[③]

2.1.2 IGRS 应用框架

IGRS 应用框架是基于基础协议、面向最终应用而定义的一系列应用交互规则，如面向家庭多媒体应用的音频/视频应用框架定义家庭场景中各种音频/视频相关设备为实现音频/视频应用所应具备的 IGRS 服务与 IGRS 客户间的配合关系。IGRS 应用框架是 IGRS 应用和扩展应用的基础。

[③] 见参考文献[5]第 14 页。

2.1.3 IGRS 应用

IGRS 应用基于 IGRS 基础协议和 IGRS 应用框架，包括 IGRS 基础应用和 IGRS 扩展应用两种类型。IGRS 设备按用途可分为多种设备类型，IGRS 基础应用是 IGRS 标准规定的与具体设备类型相关、具有 IGRS 设备功能的标准应用，某种类型的 IGRS 设备上存在某些标准的 IGRS 应用。IGRS 扩展应用是应用程序开发者基于 IGRS 基础协议和 IGRS 应用框架开发的符合 IGRS 标准的应用，以更好地发挥 IGRS 设备功能。

一个 IGRS 应用由一个或多个 IGRS 服务和一个或多个使用 IGRS 服务的 IGRS 客户交互完成。典型情况下一个 IGRS 应用交互如下图所示：

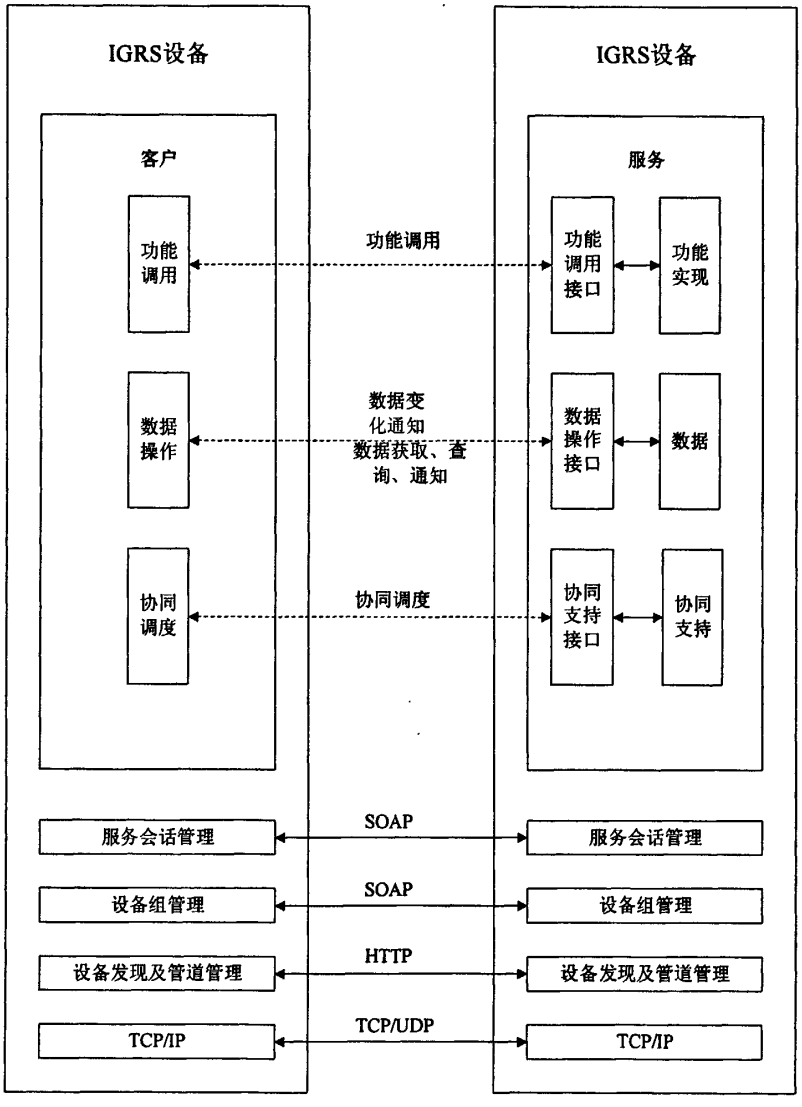

图 2-4 IGRS 应用交互模式[④]

[④] 见参考文献[5]第 13 页。

2.2 IGRS 基础协议详细介绍

对于 IGRS 基础协议，为了便于模块化的开发，可以将其分成若干模块，各个模块间的关系如图 2-5 所示：

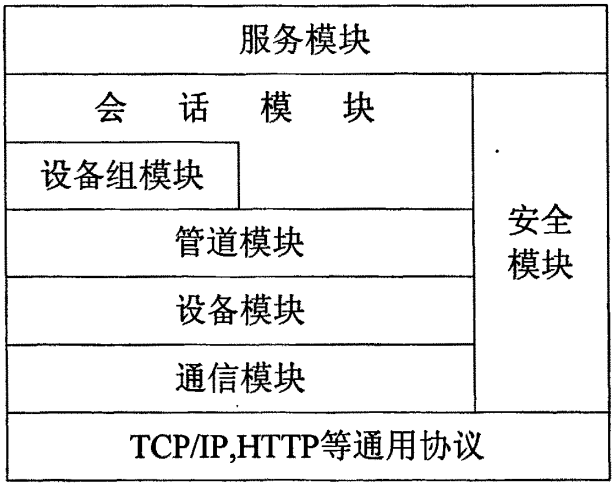

图 2-5 IGRS 基础协议模块示意图

2.2.1 通信模块

通信模块是一个比较独立的模块，它提供有限的几个接口供上层模块调用。在 IGRS 设备间交互的控制信息有两种：TCP 消息和 UDP 消息。相对应的通信模块也分成了两个模块：SSDP 模块（负责 UDP 的单播消息和多播消息的发送与接收处理）和 WEB 模块（负责 TCP 消息的发送与接收处理）。

SSDP 模块由 SSDP 客户端和服务器端两部分组成，给设备在线宣告、设备离线宣告、特定对等设备组宣告、主从设备组宣告、退出特定对等设备组通知、主从设备组解散通知、组播设备查找请求、设备组查找请求、服务宣告、服务离线宣告、组播服务查找请求等报文提供 SSDP 封装、发送功能，同时在 SSDP 服务器端接收到这些报文，进行最初处理，并传递给相应模块。

WEB 模块由 WEB 客户端和 WEB 服务器端两部分组成，WEB 客户端部分完成系统建立和管理、数据对象管理、连接、断开连接、请求发送和回调、资源释放，WEB 服务器端部分完成系统的建立、管理和注销，连接请求响应、主动或被动断开连接请求响应、接收数据的处理、发送响应报文。

2.2.2 设备模块

设备模块处在通信模块和其它上层模块之间，是一个必备的模块。设备模块主要负责的内容有以下几个方面：

1. 存储本设备的基本信息，并对该信息进行初始化及维护。一个设备的主要信息如表2-1所列

名称	含义
DeviceId	设备标识符（全球唯一）
DeviceType	设备类型
DeviceName	设备名称
DeviceGroupIdList	本设备从属设备组列表
DeviceSecurityIdList	本设备采用的安全机制列表
SecureListenerList	建立安全设备管道的IP地址列表
ListenerList	建立非安全设备管道的IP地址列表
ConfigId	设备配置变化指示值
BootId	设备重新启动指示值

表2-1 IGRS设备信息表

2. 本设备上线时负责进行设备在线宣告，设备在线期间要进行不间断的定期宣告，本设备离线时要进行设备离线宣告；
3. 负责对本设备感知到的其它IGRS设备的上下线情况进行管理，并及时的告知其它上层模块；
4. 基于组播发送设备查找消息，并对响应进行处理；接收组播设备查找报文，生成设备查找响应报文；
5. 基于单播给指定设备发送设备查找报文，并处理响应；接收单播指定设备查找报文，生成单播设备查找响应；
6. 发送设备在线检测消息，并对响应进行处理；接收设备在线检测报文，经处理后根据情况回复响应消息。

2.2.3 管道模块

管道是IGRS中提出来的一个新的概念。IGRS设备之间除了组播的查找与单播的响应这一交互过程（如发送一个组播的设备查找请求和得到相应设备的查找响应）不需要事先在设备间建立管道，其它的任何交互都要在两个设备建立好管道的基础之上。

当IGRS设备发现网络中的其它IGRS设备在线时，就通过设备管道机制和对方设

备建立设备管道。IGRS 设备间的管道分为两种：一种是非安全管道，即两个设备间没有共同支持的安全机制，没有经过相应的鉴别交互而建立的管道；另一种是安全管道，即两个设备有共同支持的安全机制，并且基于该机制进行了相互的鉴别交互而建立起来的设备管道。

两个设备间建立了 TCP 的连接即可认为它们之间存在了非安全的设备管道，因此在实现上非安全管道的建立过程比较简单：当本设备接收到其它 IGRS 设备的在线宣告时，进行检测，发现本设备保存的远程设备列表中没有该设备的信息，这表示是第一次收到该设备的在线宣告，那么就把该设备的信息保存到远程设备列表中，同时自己立刻发出一个组播的在线宣告，以便让对方能及时收到自己的在线宣告。完成上述操作后，本设备就认为和对方设备已经建立好一个非安全的设备管道。当对方设备完成相同的操作后，在这两个 IGRS 设备间就建立起了非安全的设备管道。

安全管道建立的过程就要复杂的多。为了适应不同的应用场合，IGRS 协议中定义了四种不同的安全管道，每一种管道对应着不同的安全算法。这四种管道分别是：基于对称密钥的消息鉴别机制的安全管道；基于对称密钥的消息鉴别、消息加密机制的安全管道；基于公钥的消息鉴别、消息加密机制的安全管道；基于可信赖第三方的消息鉴别、消息加密机制的安全管道。每一种安全管道的建立原则就是它们名称中提到的方式，在每一种方式下有几种安全算法可供选择使用。

图 2-6 是管道建立过程的流程示意图。

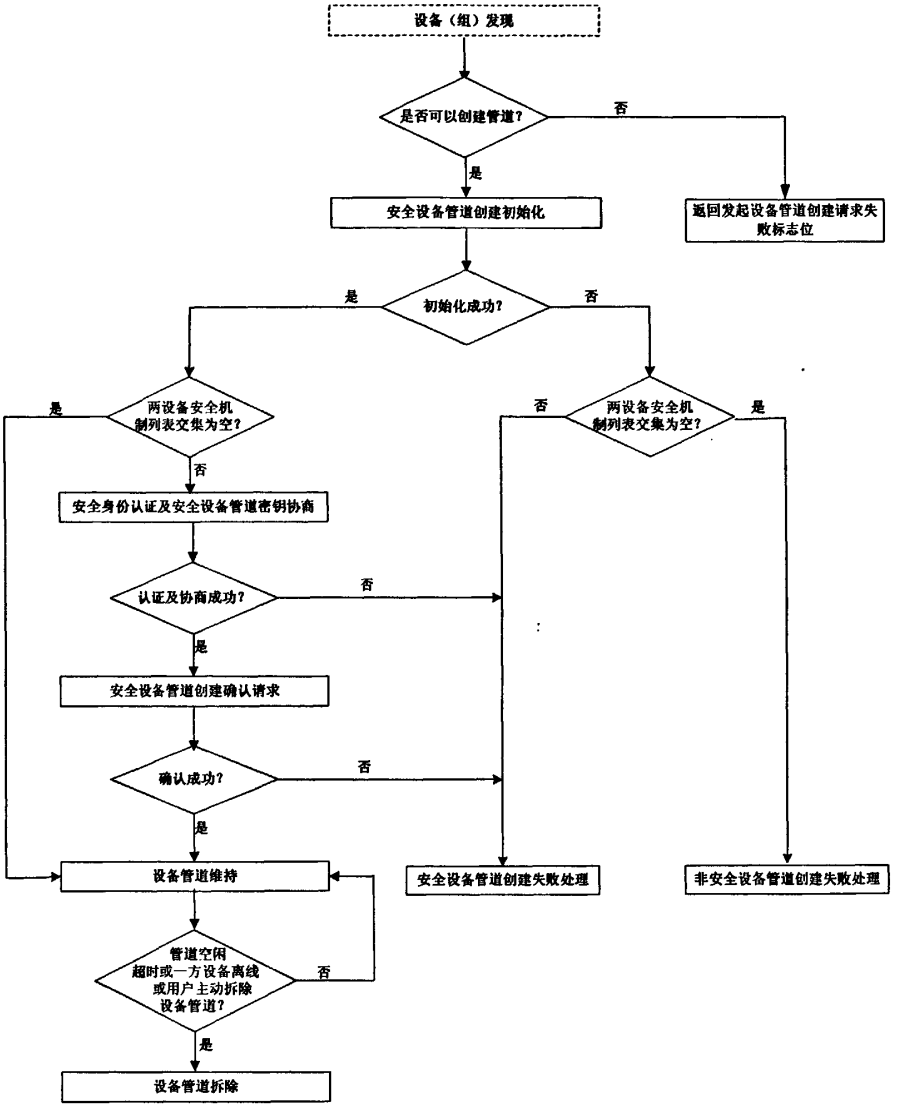

图 2-6　管道模块流程图

一般情况下，两个 IGRS 设备间的管道一直要维持到其中某一个设备离线后拆除。为了防止异常情况的发生，在程序中我们采用了定时器，即与一个设备建立好管道之后立刻启用一个定时器开始计时。在定时器到达设定时间之前，若通过该管道发生了信息的交互或者再次收到了对方设备的在线宣告，那么就将该定时器置零并重新开始计时；若到定时时间该管道一直处于空闲状态，即没有任何消息通过该管道，并且也没有收到该设备的在线宣告，那么本设备就会主动向对方设备发送一个设备在线情况的检测请求以检测对方设备是否在线，若在 30 秒内收到检测响应则定时器依旧置零并重新开始计时，若没有收到响应则管道断开。下面的示意图说明了这个定时器的流程：

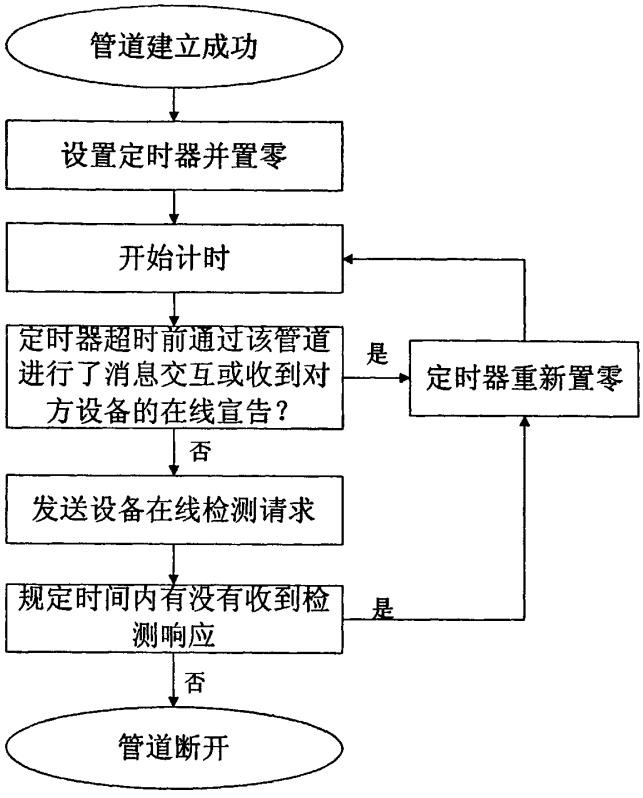

图 2-7 管道定时器示意图

2.2.4 设备组模块

IGRS 中的 IG 的全称是 Intelligent Grouping，所以在 IGRS 协议中设备组模块是必不可少的。IGRS 规定 IGRS 设备组共有三种：全局对等设备组、特定对等设备组以及主从设备组。

1) 全局对等设备组

一个 IGRS 设备一旦上线，就默认的处在一个全局对等组中，设备本身不需要做任何其它的动作。

2) 特定对等设备组

一个 IGRS 设备若想创建一个特定对等设备组，首先要进行一个设备组在线宣告，该宣告消息中要包含该设备组的 DeviceGroupId，DeviceGroupType，DeviceGroupName 等信息；同时需要在本设备的在线宣告中，将要创建的设备组的 DeviceGroupId 加入到宣告信息的 DeviceGroupIdList 中，并将其从在线宣告中宣告出去。而且该设备必须周期的对该设备组进行宣告。

其它设备收到该设备组的宣告，若自己也想加入到该对等设备组中，只要将该设备

组的 DeviceGroupId 加入到自己的 DeviceGroupIdList 中，并且在下一次的设备在线宣告中宣告出去即可。

当一个设备想退出一个对等设备组时，若该设备组不是自己创建的，那么只需要将该设备组的 DeviceGroupId 从本设备的 DeviceGroupIdList 除去，并且更新自己的设备在线宣告，即可完成退出工作；若该设备组是自己创建的，在退出之前，自己必须指定另外一个设备负责对该设备组进行周期宣告，完成指定工作后自己方能退出该设备组。关于指定的规则，在 IGRS 基础协议中并没有规定，一个比较简单的方法是可以按照加入设备组的时间先后附上权值，然后根据权值的高低决定负责宣告的设备。

特定对等设备组中的设备地位均等，没有主从之分。另外，关于设备组的创建与加入这一操作，可以由设备根据设定好的规则自动的进行，也可以通过人在 GUI（人机交互界面）上进行相关的操作完成。

3) 主从设备组

主从设备组的组成是一个主设备和若干个从设备。一个 IGRS 设备若想创建一个主从设备组，其过程同创建一个特定对等组相同，同时自己成为该主从设备组的主设备。但是其它设备若想加入这个主从设备组，就必须向该主设备发送一个加入请求，得到主设备的允许后才能加入该主从设备组。

同样，从设备想退出一个主从设备组，也必须向主设备发出退出的请求，得到允许后才能退出该主从设备组。

主设备可以发出设备组离线宣告来解除自己创建的主从设备组。该离线宣告是组播的消息，所有处在该设备组的设备都可以收到该宣告，收到宣告后即自动的退出该设备组。

2.2.5 会话模块

会话模块主要是对会话进行管理，完成会话的创建、维持、拆除的工作。会话，建立在一个客户端和一个服务之间，主要是让服务对欲调用自己的客户的身份进行鉴别，防止非法调用的发生，同时实现服务调用这个过程的管理。

1. 会话的创建

当一个客户想要调用其它设备上的某个服务时，必须事先按照服务端的要求同该服务间建立起相应的会话。一个会话创建的过程如下：

1) 客户端获取服务的描述报文

客户端主动请求获取该服务的“服务描述报文”，该报文描述了服务的详细信息，包括服务支持的安全机制。按照不同的安全机制，IGRS中规定会话种类共有15种，由表2-2所示：

	任意设备	同组设备	可信设备	同组可信设备	指定设备
任意用户	允许或禁止访问	允许或禁止访问	允许或禁止访问	允许或禁止访问	允许或禁止访问
可信用户	允许或禁止访问	允许或禁止访问	允许或禁止访问	允许或禁止访问	允许或禁止访问
指定用户	允许访问的用户标识符列表	允许访问的用户标识符列表	允许访问的用户标识符列表	允许访问的用户标识符列表	允许访问的用户标识符及设备标识符列表

表2-2 服务访问控制列表[®]

根据表中显示，服务对访问自己的对象有两个维度上的要求：设备维度和用户维度。根据不同的要求，服务可以支持其中的任意一种，前提是服务所在的设备能够支持该机制涉及到的安全算法。

2) 客户进行身份自检

当客户获取到该服务的“服务描述报文”，就会针对服务提出的会话机制进行自我身份的检验。客户端自检发现自身满足该要求，则会进行第三步，否则会话不能创建，不能调用该服务。

3) 客户发送创建会话请求

客户根据IGRS中会话创建请求消息的格式，组织一个请求报文发送到服务所在的远程设备上，并等待响应消息。

4) 服务端发送创建会话响应

服务端收到来自客户端的请求后，将进行消息的鉴别。鉴别主要从两方面进行：一是验证该消息中的会话机制和自己规定的机制是否一致，这是为了防止非法客户故意用低级别的安全机制进行非法访问；另一个是从自己的角度检查对方设备和对方用户是不是满足该会话要求。只有在检验全部合格的情况下，服务端才会响应一个会话创建成功的消息，用响应消息中的ReturnCode字段100表示，否则发送会话创建失败的响应，不同的失败原因由不同的ReturnCode字段表示。

5) 客户端进行完成确认

客户端收到响应后，对会话是否创建成功进行确认。若成功，则可以进行接下来的

[®] 见参考文献[5]第38页。

服务调用的过程，否则将错误的 ReturnCode 提交给上层由上层决定如何处理。

2. 会话的拆除

会话的创建过程是一个单向的过程，都是由客户端发送请求服务端响应，一旦建立起来后会一直存在。当客户端或者服务端任意一方想要退出该会话时，只需要按照 IGRS 中会话拆除的通知消息格式向对方发送一个通知消息，这个消息不需要经过对方的确认即可生效。

2.2.6 服务模块

服务模块在 IGRS 基础协议栈中是最上层的模块。相比与其它模块而言，服务模块与实际应用关系最密切。

在 IGRS 设备中，所有向其它设备提供的功能，如显示、音乐播放、电影播放、打印等都是以“服务”的形式出现的。一个 IGRS 设备若具备一项服务，就必须按照协议的规定基于组播宣告该服务，同时对其它设备的服务查找请求给出响应。

同时，对于服务调用这一交互过程的所有消息的处理也是由服务模块来全部处理的。对于正确的调用，服务模块需要将信息交给 IGRS 应用层的程序执行，对于错误的服务调用请求，服务模块本身就可以做出服务调用失败的响应发送给请求设备，不同的错误类型有不同的错误码。

当本设备上的服务离线时，服务模块也要进行该服务的离线宣告，这个类似于设备的离线宣告。

同时，服务模块还负责对本设备感知的远程设备上的服务信息进行存储与维护。本设备试图要调用其它设备上的服务时，都由服务模块作为统一的信息出口。

2.2.7 安全模块

安全模块在整个 IGRS 协议栈系统中属于底层模块，主要用于管道及会话模块中通信双方的身份鉴别、消息鉴别以及消息加解密，调用该模块时只需提供被处理的字符串以及选择处理的方式，经过模块内部函数的运行即可得到所需的结果。

安全模块的功能在于身份鉴别和消息加密，是供管道模块和会话模块调用的底层模块。根据安全机制的区别，本模块提供以下几种功能：

1) 提供散列算法进行身份鉴别和消息鉴别，主要是 Hash 散列算法。

2) 提供对称密钥加密方法对消息加密和解密，包括 3DES(64 位块长度/112 位密钥长

度)算法、AES（128 位块长度/128 位密钥长度）算法以及国密办给定的对称密钥算法。

3) 提供非对称密钥加密方法对消息加密和解密，主要是 RSA-1024 算法。

4) 提供 Kerberos 认证机制以及第三方认证的 X.509 证书的提供和发放。Kerberos 协议是 80 年代由 MIT 开发的一种协议。Kerberos 是一个三路处理方法，根据称为密钥分配中心（KDC）的第三方服务来验证计算机相互的身份，并建立密钥以保证计算机间安全连接。Kerberos 协议基本上是可行的，因为每台计算机分享 KDC 一个秘密，KDC 有两个部件：一个 Kerberos 认证服务器和一个授票服务器。如果 KDC 不知请求的目标服务器，则求助于另一个 KDC 完成认证交易。Kerberos 是一种网络认证协议，允许一台计算机通过交换加密消息在整个非安全网络上与另一台计算机互相证明身份。一旦身份得到验证，Kerberos 协议给这两台计算机提供密钥，以进行安全通讯对话。Kerberos 协议认证试图记录上网用户的身份，并通过使用密钥为用户间的通信加密。采用基于 X.509 证书的认证技术类似于 Kerberos 技术，它也依赖于共同信赖的第三方来实现认证。所不同的是它采用非对称密码体制（公钥制），实现上更加简单明了。这里可信赖的第三方是指称为 CA（Certificate Authority）的认证机构。该认证机构负责认证用户的身份并向用户签发数字证书。数字证书遵循 X.509 标准所规定的格式，因此称为 X.509 证书。持有此证书的用户就凭此证书访问那些信任 CA 的服务器。

2.3 本章小结

本章首先介绍了 IGRS 整个系统的架构，叙述了 IGRS 设备交互与 IGRS 应用交互的原理。接下来分模块对 IGRS 基础协议作了一个较为详细的介绍。本章对 IGRS 系统尤其是 IGRS 基础协议提供了一个较全面的介绍，也成为下面章节论述的基础。

第三章 IGRS 基础协议栈的实现

考虑到 IGRS 协议栈实际的运行环境是在数字电视、电脑等具有较高处理能力的设备上（在诸如冰箱、空调等处理信息能力较弱的设备上可以运行 IGRS 微协议栈，IGRS 微协议栈是用单线程实现的），IGRS 协议栈的实现采用了多线程的机制，包含了四个线程：主线程、侦听本地客户端上下线的线程、处理与本地客户端信息交互的线程、关闭协议栈线程。IGRS 协议栈进程启动后会连续的启动上述四个线程，然后前三个线程都处于不停的循环轮询的状态，关闭协议栈线程处于阻塞状态；一旦关闭协议栈线程被激活，则整个进程都会立刻进行资源释放并退出运行。四个线程功能上保持高内聚、低耦合的特性，线程之间采用信号量的机制实现同步。

3.1 主线程

主线程的功能在于负责和其它设备的信息交互，并且对交互的信息进行最初的处理。主线程是消息驱动的模式，正常的时候它在不停循环，对两种情况作相应的处理：一种情况是本地的“处理与本地客户端信息交互的线程”调用了主线程提供的 API，这时主线程需要完成相应的消息生成与发送的工作；另一种是关闭协议栈的线程被激活，这时主线程停止轮询，进行资源释放与退出。

主线程的工作原理示意如图 3-1：

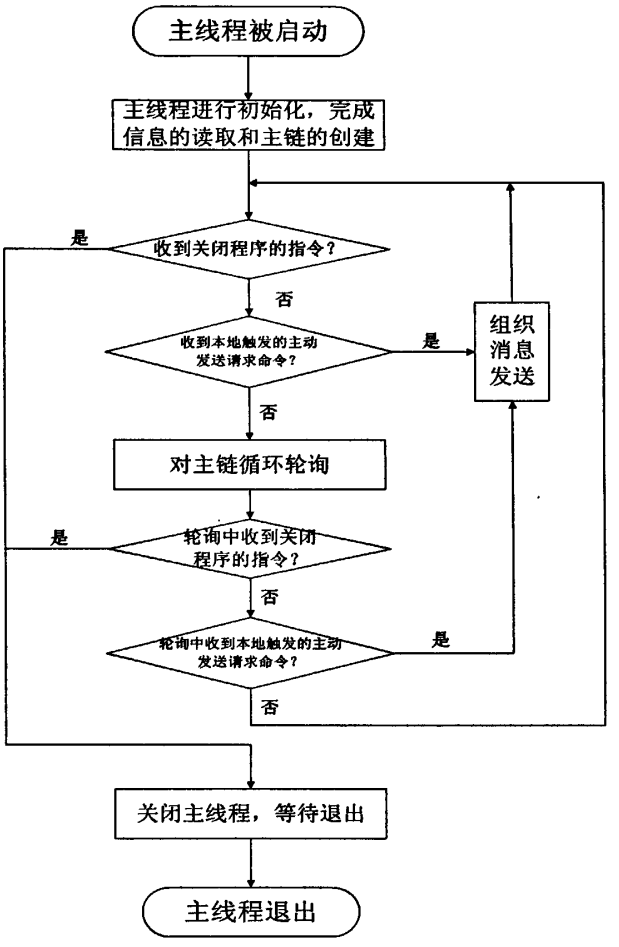

图 3-1 主线程工作示意图

3.1.1 主线程初始化

主线程启动后首先完成相关的初始化工作，该项工作包括以下内容：

1) 生成主线程的主链表 Chain。在下面的初始化工作中会不断的往该主链上添加节点元素，同时当初始化完成后主线程也是在不停的循环轮询这条主链表上的节点。

2) 生成和初始化主结构体变量 IGRSDataObject，并将其作为一个节点加到 Chain 中。IGRSDataObject 是协议栈主线程唯一的一个全局变量，结构体中包含了设备的基本信息，如 DeviceId、DeviceName 等，同时 IGRSDataObject 也包含了各个模块信息链表的根节点，如管道状态信息链表的根节点 PipeStaList、会话状态信息链表的根节点 SessionStaList、设备组信息链表根节点 DeviceGroupList 等。

3) 初始化日志文件，生成一个名为 IGRS.log 的日志文件。该日志记录了程序运行中的所有有用的信息，该日志支持信息分级，有 Error、Info、Debug 三个级别，可以

通过在 IGRS.ini 文件中进行设置。当在调试阶段，设为 Debug 级，这样 Error、Info、Debug 级别的信息就能全部记录下来，当用户使用时可以设为 Error 级，这样 IGRS.log 只记录程序运行中的错误信息，避免了设备长时间运行产生大量的 log 文件。

4) 生成系统的定时器管理器，该管理器负责主线程中所有的定时器的生成、重置、销毁等所有的工作。

5) 读取本地存储的设备描述报文，从中获得本设备的 DeviceId、DeviceName、DeviceType 等基本信息，将这些消息存储到 IGRSDataObject 中。

6) 读取本地的 IP 地址，并将其作为本设备的侦听地址。

7) 生成通信用的 TCP Socket Pool，其中的每一个 Socket 都作为节点元素加入到 Chain 中，并且对于每一个 Socket 都会为其生成一个定时器，这些定时器由上面提到的定时器管理器统一管理。

8) 生成用于组播和单播的 UDP Socket 各一个，分别加入到 Chain 中。

完成上述的初始化工作后，主线程将会发出一个本设备的在线宣告，接下来就会对 Chain 的节点进行不停的循环轮询，并对相应的状态进行处理。

在轮询的过程中，有两种情况会造成循环轮询的停止：一是处理与本地客户端信息交互的线程向主线程发出命令，让其向其它设备发出相应的请求消息。这时，轮询暂时终止转为处理该请求消息并用一个 Client Socket 将其发送出去。当该项工作完成后，轮询工作继续从原来的位置进行。另一个是关闭协议栈线程被激活，这时所有线程必须即刻完成资源释放，然后关闭。由于整个系统被关闭，所以主线程的循环轮询工作也全部结束。

除此之外主线程不断的对主链上的所有节点进行循环轮询。由于主线程的主要任务是与其它设备的信息交互，而交互采用了 Socket 的方式，所以在轮询工作中主要是对每一个 Socket 的状态进行查询和处理。在主链中每一个 Socket 有四个节点与之对应，分别是：预处理节点、主处理节点、后处理节点和定时器节点。本文接下来就分别叙述一个 Client Socket 和一个 Server Socket 的工作原理。

3.1.2 Client Socket 的工作原理

一个 Socket 的工作原理实际上就是它状态循环改变的过程，Client Socket 在系统的初始化工作中生成与初始化，完成这些工作后该 Socket 处于空闲状态，等待上层的调用。

在一个循环中，主线程对空闲的 Socket 不作处理。

当该 Socket 被从 Socket Pool 中选中用来发送一个消息时，状态就会变成准备发送状态。当循环到这种状态的 Socket 的预处理节点时，预处理函数就会把这个 Socket 置入 writeset 和 errorset 中，然后退出预处理函数。

当该 Socket 成功的发送了一个消息后，状态就会变成准备接收响应消息的状态。当循环到这种状态的 Socket 的预处理节点时，预处理函数就会把这个 Socket 置入 readset 和 errorset 中，然后退出预处理函数。

Socket 的主处理函数是统一的，采用的是 TCP/IP 中的 select 函数。select 函数是检查若干个 Socket 的状态的函数，主要是检查 writeset 中 Socket 的可写性、readset 中 Socket 的可读性以及 errorset 中 Socket 是否出错。一旦有一个 Socket 满足要求，即 writeset 中有一个 Socket 可写或者 readset 有一个 Socket 可读或者 errorset 中有 Socket 出错，该函数即可返回；否则就在设定的等待时间后返回，该时间可以从 0 到无限大。等待时间设为 0 时不管有没有满足条件的 Socket，select 函数总是及时返回的；设为无限大时，如果没有满足条件的 Socket，select 函数会一直处在等待的状态不返回。为了调试的方便，该时间被设定为 10 秒，10 秒内若没有 Socket 可用的话循环也会重新进行。

退出主处理函数后，对于不同状态的 Socket 又有不同的后处理函数。对于处于准备发送的 Socket，后处理函数将会利用该 Socket 将消息发送出去。当消息正常发送后，该 Socket 的状态就会改变成准备接收响应消息的状态。

对于处于准备接收响应消息的状态的 Socket，后处理函数会从该 Socket 中收取响应消息并提交给相应的模块进行处理。然后为 Socket 添加一个定时器（时间长为 3 秒），在 3 秒内若该 Socket 又被选中则其状态又变成准备发送的状态，超时则该 Socket 变成空闲状态等待再次被选中，这样就完成了 Socket 的循环转变。

上述的过程有一个前提：发送和接收消息都正常。当发生异常的情况比如消息发送不成功或者接收失败时，就会立刻将该 Socket 的状态重置。这就是在预处理函数中对于每一个状态的 Socket 都要置入 errorset 中以便主处理函数检查的缘故。

下面的流程图可以说明 Client Socket 的整个工作状态的详细变化的过程：

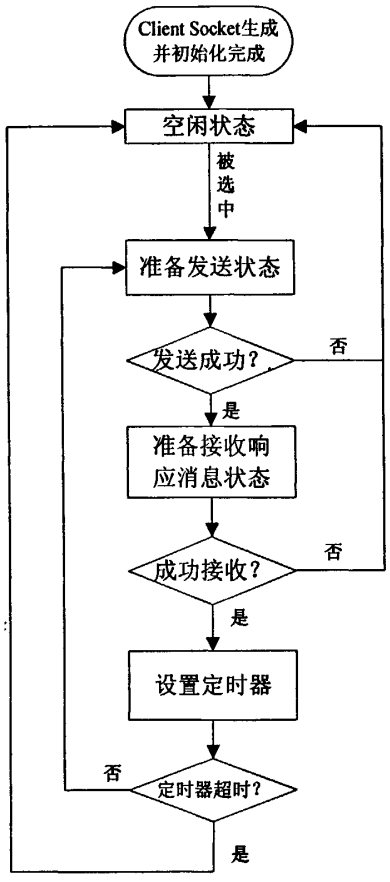

图 3-2　Client Socket 工作原理图

3.1.3　Server Socket 的工作原理

Server Socket 同样在系统的初始化工作中生成与初始化，完成这些工作后该 Socket 处于空闲状态。

当该 Socket 被从 Socket Pool 中选中用来接收来自远程设备的连接请求时，状态就会变成准备接收状态。当循环到这种状态的 Socket 的预处理节点时，预处理函数就会把这个 Socket 置入 readset 和 errorset 中，然后退出预处理函数。

当该 Socket 成功的接收完消息后，系统立刻会试图通过它发送响应消息，这时它的状态就会变成准备发送响应消息的状态。当循环到这种状态的 Socket 的预处理节点时，预处理函数就会把这个 Socket 置入 writeset 和 errorset 中，然后退出预处理函数。

Server Socket 的主处理函数和 Client Socket 的是一致的，这里不再赘述。

对于处在准备接收状态的 Server Socket，后处理函数会试图从该 Socket 中获取消息，成功获取后会将其状态置为准备发送响应消息的状态，获取失败则重置该 Socket。

对于处在准备发送响应消息的状态的 Server Socket，后处理函数会试图从该 Socket 中将响应消息发送给客户端。发送成功后，则继续等待试图接收下一个消息，若在定时器超时或收到对方断开连接的消息的情况下，重置该 Socket，其状态变为空闲。图 3-3 说明了 Server Socket 的工作过程：

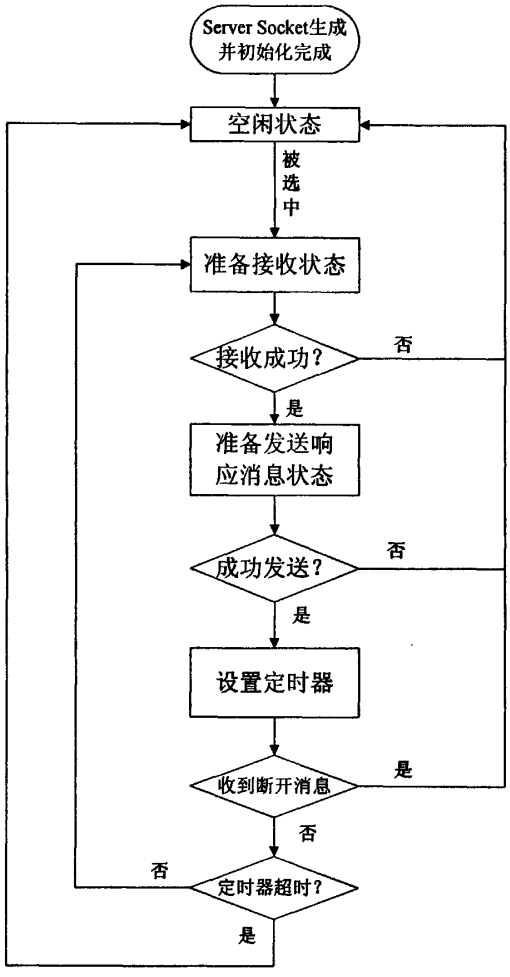

图 3-3　Server Socket 工作原理图

3.1.4　IGRS 消息处理的工作原理

主线程的主要功能是和远程设备的信息交互，因此对交互信息的处理是主线程的一个主要任务。IGRS 报文的消息头的第一个字段表明了消息的类型，有 NOTIFY、M-NOTIFY、M-POST、M-GET 等，所以主线程收到 Socket 的消息后首先就根据消息头的第一个字段将消息进行分类。在 IGRS 中交互的信息可以分为两大类，一类是基于 UDP 的，一类是基于 TCP 的。

1) 对UDP消息的处理

在主线程的初始化中生成用于单播和多播的UDP Socket各一个，它们分别处理单播和多播的消息。主线程的每一次轮询都会查看这两个Socket中有没有可读的消息。用于多播的Socket接收的消息有设备在线宣告、设备离线宣告、服务在线宣告、服务离线宣告、组播的服务查找请求消息、组播的设备查找请求消息，用于单播的Socket接收的消息是对组播查找请求消息的单播响应消息。UDP的消息的消息头的第一个字段的类型有NOTIFY（属于多播），M-SEARCH（属于单播）两种，所以根据这个字段进行区分就可以找到对应的处理函数。

2) 对TCP消息的处理

接收到TCP的消息后，首先也是解析该消息的消息头的首字段，在IGRS中，请求消息的类型有M-POST，M-GET，M-NOTIFY，解析完这些消息后会有相应的函数入口来处理，如IGRSProcessPostPacket、IGRSProcessGETPacket等。在每一个入口函数中，又会解析消息头中另一个字段 01-IGRSMessageType，根据这一个字段能够把请求消息分发到各个相应的模块中去处理。

在IGRS协议栈中，每一个请求消息都有一个回调函数，用来对请求消息对应的响应消息进行处理工作。该回调函数在请求发送时必须设置好，与请求消息的SequenceId要一一对应。对于TCP的响应消息，消息头的第一个字段都是“HTTP/1.1 200 OK”，如果解析到这个字段，表明这是一条响应消息。那么程序会再解析消息中的“01-AcknowledgeId”，根据它找到对应的SequenceId，然后再找到对应的回调函数就可以对处理这条响应消息。

3.1.5 主线程退出

当“关闭协议栈线程”被激活时，主线程将立刻进行资源的释放工作，并按照下面叙述的顺序完成系统退出工作：

1. 检查本设备上是否有服务存在，是否有其它客户端与本地的服务建立会话。若有则作为服务端向客户端发送一个会话拆除的通知消息，然后基于组播发送本服务离线的宣告，最后完成服务相关资源的释放。
2. 检查自己作为客户端是否有与其它设备上的服务建立会话。如果有则作为客户端向服务端发送一个会话拆除的通知，并在本地删除该会话的信息。
3. 检查本设备保存的所有管道信息，拆除每一个与本设备建立的管道，然后删除本地

关于该管道的信息资源。

4. 检查与本设备相关的设备组信息，如果本设备建立了主从设备组则拆除该设备组，若本设备建立了特定对等设备组则退出该设备组并指定一个设备对该特定对等设备组进行宣告，若本设备加入了某个设备组，则只需要按照不同设备组的退出方法退出即可。
5. 组织报文发送本设备的离线宣告。
6. 系统资源的释放工作，这是对应于主线程初始化工作的。

3.2 侦听本地客户端上下线的线程

本线程启动后首先进行简单的变量的定义与初始值的设定。初始工作完成后，生成一个 TCP Server Socket 在本地地址某一个端口上进行侦听，端口号可以配置，但是要和客户端进程（独立于 IGRS 协议栈，是闪联电视运行需要的另外一个进程）配置的端口号一致。

当线程收到一个连接请求就会立刻去获取请求消息。该消息的格式被商定如下："客户端标识符#供协议栈写消息的内存地址#供协议栈读消息的内存地址#动作标记[⑥]"，其中"#"为字段间隔符。

得到一个消息后，首先对动作标记的值进行分析。若该值表示是一个客户端进行注册，那么就会去获取客户端标识符、供协议栈写消息的内存地址、供协议栈读消息的内存地址并将它们加入到一个链表中。本系统设计成可以一个 IGRS 协议栈同时处理若干个客户端的模式（每一个客户端的标识符是唯一的），所以采用了链表的形式来保存注册的客户端的信息。其中客户端标识符作为在该链表中搜索指定的客户端的关键字；供协议栈写消息的内存地址和供协议栈读消息的内存地址用作内存的映射，映射工作完成后供协议栈写消息的内存地址用于在以后和该客户端的交互中发送消息用，供协议栈读消息的内存地址则用于协议栈读取来自该客户端发送的消息，这两项工作在处理与本地客户端信息交互的线程中进行。

若消息中的动作标记值表明是一个客户端的注销请求，那么同样去获取客户端标识符、供协议栈写消息的内存地址、供协议栈读消息的内存地址。用标识符作为关键值搜索前面提到的链表，找到对应的客户端的信息，然后检测本次获取的内存地址的值和以前保存的值是否一致。若每项工作正常进行，则完成对应的节点的删除以及对应资源的

⑥ 动作标记包括客户端注册、客户端注销两种，分别用 1、2 来标识。

释放。

完成对消息的处理后，就会释放掉新建的Socket的资源，重新进入到侦听的工作中即进入下一个循环。本系统底层有一套完善的链表处理系统，当有节点的增添或减少时，系统会自动的对链表的各项属性的值做及时的更新。

在设计中，为了防止异常情况对整个系统造成影响，做了以下的防护工作：

1) 在一个客户端来注册时，若发现记录客户端信息的链表中已经有相同的客户端存在，则不再对内存做映射工作以减少工作量。（当客户端程序异常中断，这时它并没有向协议栈发出注销的消息，当它重启后再次去注册时，就会造成这种现象的发生。）
2) 当一个客户端注销时，记录客户端信息的链表中没有对应的客户端的信息或者对应的内存地址不一致，则注销工作直接被结束不做任何的资源释放工作。
3) 当一个客户端注销时，记录客户端信息的链表显示节点数为0（表示没有客户端注册到本协议栈上），则注销工作同样直接被结束不做任何的资源释放工作。
4) 不管该线程运行在什么阶段，一旦关闭协议栈线程被激活，立刻完成资源的释放工作，结束本线程。

图3-4说明该线程的工作原理：

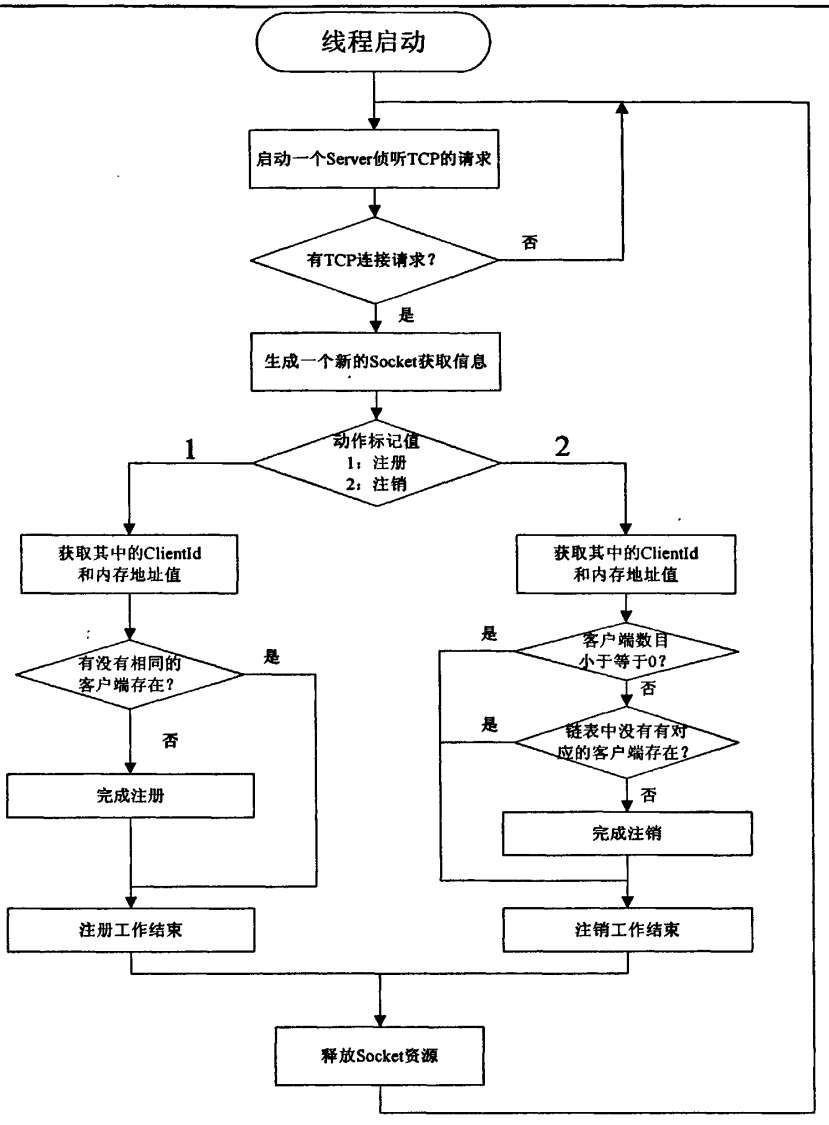

图 3-4 侦听本地客户端上下线的线程工作原理图

3.3 处理与本地客户端信息交互的线程

本线程完成的任务是实现协议栈进程和客户端进程实现数据的传输，传输采用了共享内存的方式。共享内存是指两个或多个进程共用内存中相同的区域，其目的是节省内存空间，实现进程间通信，提高内存空间的利用效率。共享内存可以说是最有用的进程间通信方式，也是最快的 IPC 形式。两个不同进程 A、B 共享内存的意思是，同一块物理内存被映射到进程 A、B 各自的进程地址空间。进程 A 可以即时看到进程 B 对共享内存中数据的更新，反之亦然。由于多个进程共享同一块内存区域，必然需要某种同步机制，互斥锁和信号量都可以。采用共享内存通信的一个显而易见的好处是效率高，因为

进程可以直接读写内存，而不需要任何数据的拷贝。对于像管道和消息队列等通信方式，则需要在内核和用户空间进行四次的数据拷贝，而共享内存则只拷贝两次数据：一次从输入文件到共享内存区，另一次从共享内存区到输出文件。实际上，进程之间在共享内存时，并不总是读写少量数据后就解除映射，有新的通信时，再重新建立共享内存区域。而是保持共享区域，直到通信完毕为止，这样，数据内容一直保存在共享内存中，并没有写回文件。共享内存中的内容往往是在解除映射时才写回文件的。因此，采用共享内存的通信方式效率是非常高的。

本线程启动后，只要没有异常情况，就不停的对两个链表执行循环遍历：一个是前面所说的存储了与客户端交互信息的共享内存地址的链表，其中的消息都是来自某个客户端进程，准备通过该线程交由主线程发送给其它设备的；另一个则是存储了来自其它设备的消息，由主线程接收准备通过该线程提交给某个客户端进程。

3.3.1 对客户端进程交给主线程的消息处理

在 IGRS 中交互的每一条请求信息，都会有一个唯一的序列号与之对应，称为 SequenceId；对于该请求消息对应的响应消息有一个唯一的 AcknowledgeId，值与 SequenceId 一样。SequenceId 是 32 位的无符号整型数据，由主线程生成，从 1 开始，每发送一个请求消息就增加 1，直到达到临界值（根据处理器位数的不同而不同）再跳转为 1。

客户端传给本线程的消息是结构体的形式，该结构体中包含两个变量：MessgaeType 和 Message。MessageType 用来标识传送的消息的类型，如服务调用请求，基于会话的通知消息等，Message 是包含该消息具体信息的内容的字符串。交互中最多的消息就是服务调用请求，对于这种类型的消息 Messgae 字符串定义为“DeviceId#ServiceId#ClientId#InvokeMessage”，其中 DeviceId 标识请求消息发送的目标设备，ServiceId 标识请求消息发送的目标服务，ClientId 标识发送者的身份，InvokeMessage 是具体的消息片段（该消息由客户端进程生成与处理，内容是具体的调用信息），“#”是字段间隔符。

本线程收到来自客户端进程的消息后，首先进行消息类别的判断，也就是对 MessageType 进行判断，然后解析 Message 中的各个字段。完成这些工作后，就会调用主线程提供的 API 将消息发送出去。这时主线程将停止它的循环来处理这个事件。主线程首先会为该消息生成一个 SequenceId，然后将该 SequenceId 和消息中的 ClientId 绑定

并且存储在本地。与 ClientId 绑定是为了便于稍后对该请求消息的响应消息进行处理，这在另外一种消息处理中会详细说明。当消息比较多时，用链表的形式将消息按照时间的先后排列。接下来主线程会按照 IGRS 基础协议的规定将有用的信息封装成规定的格式将消息发送出去。

3.3.2 对主线程提交给客户端进程的消息处理

主线程提交的消息有两大类：一类是对应于前面请求消息的响应消息，还有一类是通知消息，如远程设备在线宣告、设备离线宣告、远程服务在线宣告、远程服务离线宣告等。

对于响应消息，主线程首先会完成部分解析工作。主线程完成的解析包括：

a) 分析消息中的 SourceDeviceId，该字段用来判断消息来自哪个设备；
b) 分析消息中的 SourceServiceId，该字段用来判断消息来自哪个服务；
c) 分析消息中的 TargetClientId，该字段用来判断对应的请求消息的发送者；
d) 分析消息中的 AcknowledgeId，该字段用来找到对应的 SequenceId 的请求消息；
e) 将具体的响应消息（和 InvokeMessage 相对应）提取成一个字符串。

主线程完成这些工作后，就会按照一定的格式将消息存储在本地的一个链表中，并且将该消息的状态设为等待发送。至此主线程对于响应消息的处理工作结束。本线程会对主线程的这个链表进行不停的循环查找。当查找到节点的消息状态是等待发送时，就会根据 ClientId 的值找到与之对应的共享内存的地址，然后将其按照 MessageType 和 Message 的形式组成结构体写入到与该客户端进程共享的内存中等待对方来获取，同时将其状态改为发送成功，并不进行资源的释放；当查找到状态是发送成功时，才将该消息有关的资源释放。这样做的原因是在以后的扩展中，有一些消息发送后状态可能改变成等待响应而不是发送成功，那么就不能立刻将其删除。

对于设备在线宣告等消息的处理和上述消息类似，主要的差别在于该类宣告消息并不是针对某一个客户端的，因此只要注册到本地协议栈的客户端，本线程都有义务将该类消息及时的发送到每一个共享内存中，以便让所有的客户端知道最新的设备在线的信息。

3.4 关闭协议栈线程

该线程启动后一般处在阻塞的状态，一旦该线程被触发，立刻完成整个系统的资源

释放工作然后关闭整个 IGRS 协议栈。触发的方式可以根据不同的应用有所不同，一般可以设计为由客户端进程触发，发送一个特定的字符给协议栈即可完成该项工作。

3.5 程序调试

3.5.1 编译环境简介

在开发初期，为了加快开发的进度，VC.net 被选择作为开发工具。原因在于它具有强大的调试功能，而且调试工作非常的简单，在后来转到 Linux 下开发时能深刻的体会到。因此在起初的代码编写、单元测试以及大部分的集成测试和系统测试都是在 VC.net 的环境中进行的。

当在 VC 中测试通过后，我们将该程序移植到了 Linux 机器上进行调测。由于很多家庭设备的操作系统都是基于 Unix 的，但是同样的内核下，用安装了 Linux 系统的电脑代替家电设备进行调试可以提高调试的效率。

在上述两个开发环境中的调试都进行完后，我们将我们的协议栈移植到了目标主板上进行测试。在开发中采用了 TI 的“Davinci”主板，该主板是 TI 最新的一个产品，采用 DM6443 或 DM6446 的处理器，具有强大的图片处理功能，对于开发电视而言是非常适合的。

3.5.2 程序调试过程

对于整个程序采用了单元测试、集成测试和系统测试的形式进行。为了加快开发与调测的进度，使得各个模块的开发能够独立进行，特地开发了一个测试平台，该测试平台可以按照要求发送各种模拟的报文来测试开发程序的功能。

首先是各个模块的编辑者自行完成本模块的单元测试，该过程不需要其它开发它模块的介入，需要时由测试平台模拟相应的开发模块来完成本模块的单元测试。

当每一个模块的单元测试进行完后，就会按照测试文档中的要求将相应的模块组合起来进行集成测试，如将设备模块和管道模块进行组合测试管道创建与拆除的功能，将服务模块和会话模块进行组合来测试会话的创建与拆除。

最后进行的是系统测试。该测试由开发的 IGRS Stack 的可执行程序和测试平台模拟的一个 IGRS Stack 进行综合的交互，按照测试文档中的要求对于每一种情况进行测试。

当以上的测试进行完后，将两个开发的 IGRS Stack 同时运行在两台设备上来测试两

者之间的交互情况，并进行长时间的交互来测试系统运行的稳定性。

由于 IGRS 是一个通信协议，因此按照标准开发出来的协议栈应该能够顺利的进行交互。所以，又将 TCL 协议栈同北京联想的协议栈进行了交互测试，即用 TCL 的测试平台对联想的协议栈进行了测试，用联想的测试平台对 TCL 的协议栈进行测试，这两个测试均通过后又让两个协议栈直接进行交互来进一步的验证测试的结果。

3.5.3 程序结果描述

经过了一系列的测试与修改，完成了 IGRS Stack 的可执行程序。程序启动后会按照顺序做如下工作：

1. 读本地的配置文件 IGRS.ini，从中可以确定运行产生的日志文件写在何处等信息。
2. 读取本地的本设备描述文档 DeviceDesc.xml，从中可以获取到 DeviceId、DeviceName、DeviceType 等一系列设备相关信息。
3. 进行设备在线宣告，接着进入循环状态，当有消息需要处理时进行相应的处理工作。
4. 收到关闭命令后，程序释放资源退出运行。

该协议栈可以独立稳定的运行在 Windows、Linux 的操作系统上，顺利与其它设备上的 IGRS 协议栈进行交互工作。IGRS Stack 也可以和 IGRS 客户端进程同时运行在一台设备上，通过两者的合作实现闪联电视的功能。通过和“闪联之家”（安装在电脑上的闪联软件）的交互，可以实时的将电脑上的电影、音乐、图片在闪联电视上进行播放，实现了数字家庭内设备间的智能互联与资源共享。

3.6 本章小结

在第二章叙述的基础上，本章详细的说明了 IGRS 基础协议的实现方法。IGRS 协议采用了多线程的方式，本章对每一个线程的功能和工作方式都作了相应的说明。对于主线程，详细的介绍了它初始化和正常工作的原理，对 IGRS 消息处理机制也进行了叙述；对于侦听本地客户端上下线的线程，叙述了客户端正常上下线的原理，同时也给出了非正常情况时的处理方法；对于处理与本地客户端信息交互的线程，按照消息的传输方向对其进行了分析；对于关闭协议栈线程则简单的进行了叙述。最后简单的论述了程序的编译环境、调试的过程和程序的运行方式。

第四章　IGRS 会话机制的实现

会话是建立在 IGRS 客户和 IGRS 服务之间的，客户与服务之间通过 IGRS 基础协议栈通信，简单的结构示意图如下：

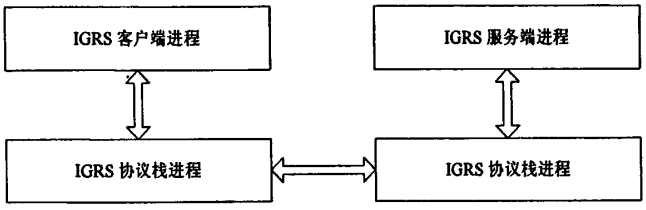

图 4-1　会话建立的对象示意图

一个完整的会话过程流程图如图 4-2 所示：

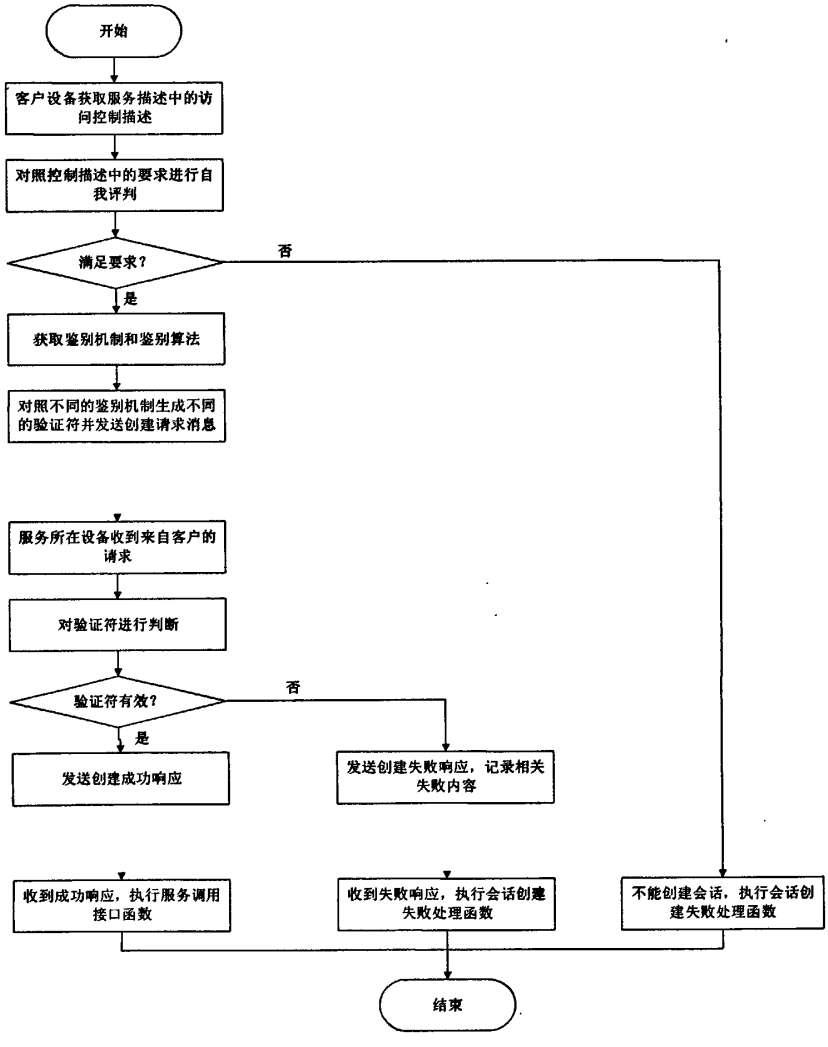

图 4-2　会话过程流程图

IGRS 客户端进程和 IGRS 服务端进程都是通过注册的方式和本地的 IGRS 协议栈进程取得联系，之后通过前面所述的共享内存的机制与各自的协议栈交互信息。IGRS 会话分为一般会话和同一设备组内两设备间的会话两种，本文接下来就分别对两种会话的每一步的实现过程进行详细的阐述。

4.1 一般会话的实现

4.1.1 服务信息的获取

当一个 IGRS 协议栈进程启动后，会向组播地址 239.255.255.250：1900 发送本设备的在线宣告，同时自己也在这个地址上侦听其它设备的在线宣告等组播信息。当接收到某设备上线的通知后，本设备会基于一定的安全机制和该设备建立起相应的管道，一般建立的是非安全管道。管道建立成功后本设备就会主动向对方获取该设备的“设备描述报文”，同时向该设备发送一个指定设备的服务查找请求（在 IGRS 中，服务查找可以分情况进行，如按照 DeviceId，DeviceName 进行查找），要求其告知注册在该设备上的所有的服务的基本信息。得到查找服务的响应后，本设备还会对响应消息中包含的每一个服务发送一个“服务描述报文”的获取请求，然后将所有得到的服务信息在本地进行保存。相同设备上的所有的服务保存在同一个链表中，该链表的根节点就是该设备的 DeviceId，每一个服务占据一个节点。每个节点是一个结构体，包含了服务的详细信息。

服务的基本信息的获取还可能是接收到了该服务的在线宣告消息。但是服务在线宣告不同于设备在线宣告，服务只需第一次上线时进行一次服务在线宣告即可。当本设备上线时，远程设备上的服务可能已经宣告过了，因此每一个设备上线针对其它设备进行服务查找这一步是必须的。

4.1.2 验证符的生成

完成服务信息的获取后，IGRS 协议栈就等待本地的 IGRS 客户端进程来注册。当有本地客户端完成正确的注册后，协议栈就会将其保存的远程在线设备的信息通过共享内存的方式告知客户端，信息的格式为

“DeviceId#DeviceName#DeviceType#DeviceSecurityIdList#SecureListenerList”，同时若某个远程设备上注册有服务，也会将该服务的消息告知注册在本地协议栈的客户端，信息的格式为“ServiceId#ServiceName#ServiceType#DeviceId#AuthenMechanism”。这样可

以让客户端得到最新的设备与服务信息。

提交完以上的信息后，协议栈就会以该客户的名义试图与本地保存的其它设备上的服务建立会话。一个会话只能建立在一个客户端和一个服务之间，因此有几个服务就要建立起相同数目的会话。

对于已成功建立的会话，本地会以链表的形式保存该会话，一个会话的信息有以下内容：

一个会话包含的字段信息	信息的解释
DeviceId	会话涉及服务所在设备的标识符
ServiceId	会话涉及服务的标识符
sockaddr_in Remote	会话涉及服务所在设备的 IP 地址
SessionSecMech	该会话采用的安全机制
ClientId	会话涉及客户的标识符
Token	会话采用的验证符
ServicePort	端口标记，客户端为 1，服务端为 0
SucceedFlag	会话建立的成功标记，0 为成功，-1 为不成功

表 4-1 会话包含的信息

对于与每一个服务建立会话，本地协议栈都会首先验证客户端与该服务之间已经是否存在建立好的会话，若有则直接返回，没有则进行下一步的处理。验证的方法就是搜索本地的会话信息链表，查找是否有以该客户端和服务端为关键字的成功建立的会话信息存在。

本地协议栈以服务的 ServiceId 和服务所在设备的 DeviceId 作为关键字去搜索本地链表找到该服务的节点，然后分析该服务的“服务描述报文”获取该服务的安全机制描述。根据不同的安全机制，客户端需要生成不同的验证符 Token。Token 是一种表示某种特定信任属性的证书，它的产生方式和结构定义见表 4-2。对于不同的安全机制 Token 也需要采用不同的生成方式，一般会话建立的过程中可能使用到的是前三种生成方式。

1) 安全机制为 urn：IGRS：ServiceSecurity：NULL 的 Token 生成

NULL 表示采用非安全机制，不用生成 Token 字段。

2) 安全机制为 urn：IGRS：ServiceSecurity：PreSharedKey 的 Token 生成

PreSharedKey，即预共享密钥，是一对对称密钥，采用该安全机制进行通信的两个 IGRS 设备在通信前都应该拥有该密钥，并且只有这两个设备拥有此密钥。nonce 在 IGRS

协议中被称为“协议新鲜子”，在实现中用随机数生成器生成 nonce。

用户身份鉴别机制（安全机制）	鉴别算法	Token
urn：IGRS：ServiceSecurity：NULL	无	无
urn：IGRS：ServiceSecurity：PreSharedKey	PreSharedKeyAuthen	{nonce，PreSharedKeyAuthen(nounce||用户 ID，预共享密钥)}
urn：IGRS：ServiceSecurity：PKICertificate	PublicKeyAuthen	{nonce，用户公钥证书，（基于 PublicKeyAuthen 用私钥对消息（nonce||用户 ID 进行数字签名的结果））}
urn：IGRS：ServiceSecurity：3rdPartyAuthenServie	Kerberos V5	可信第三方根据约定的密码算法生成的凭证（Ticket）

表 4-2 验证符产生的机制与内容[7]

因此在该安全机制下，一个 Token 的生成步骤如下：

a) 用随机数生成器生成 nonce；

b) 将 nonce，用户 ID（ClientId）以及 PreSharedKey 三者分别转换成字符串的格式，并且进行“拼接”，得到一个新的字符串，如 nonce 为 123，ClientId 为 456，PreSharedKey 为 abcdef，那么拼接三者得到的结果就是“123456abcdef”；

c) 对拼接后的字符串进行加密运算得到密文字符串 EncryptString，这里采用的是对称密钥的加密，密钥采用的是 PreSharedKey 本身，在 IGRS 协议中规定支持的加密算法有 DES，AES 等常用的对称密钥加密算法；

d) 将第一步产生的 nonce 和 EncryptString 再一次进行拼接。

经过上述四步的工作，预共享密钥机制下的 Token 即可生成。

3) 安全机制为 urn：IGRS：ServiceSecurity：PKICertificate 的 Token 生成

这种安全机制是基于非对称密钥加解密算法的，因此涉及到公钥的公布方法。在 IGRS 协议中，采用的是 X.509v3 公钥证书（见图 4-3），相对应的私钥由本设备自身保存不对外公布。

[7] 见参考文献[5]第 105 页。

版本号
证书序列号
签名算法标识
颁发者名字
证书有效期
主体名
主体的公开密钥消息
颁发者标识符
主体标识符
扩展
签名

图 4-3 X.509V3 公钥证书

在实际应用中，依然是将公钥证书写成字符串的形式进行传送。所以本机制下的 Token 的生成步骤为：

a) 用随机数生成器生成 nonce；

b) 将公钥证书写成字符串的形式；

c) 用公钥对应的私钥对 nonce 和用户 ID 拼接的结果进行加密得到加密的字符串；

d) 将上述三步的结果进行拼接。

在 Token 生成后，本设备进行设备与用户访问权限的自我验证。该验证按照第二章提到的两个维度进行自检，依次验证本设备及本客户是否满足服务端对访问的设备和客户的要求。通过这个自我验证后才能进行下一步的工作，否则会话创建步骤到此结束。

4.1.3 请求消息的生成

完成前面两步的工作后，协议栈会按照规定的格式生成会话创建请求消息，该消息的内容与格式规定为：

会话创建请求消息	消息字段说明
M-POST /IGRS HTTP/1.1	扩展 HTTP 命令行
Host：目标设备 IP：端口	必备字段
MAN："http：//www.igrs.org/session"；ns=01	必备字段
01-IGRSVersion：IGRS/1.0	必备字段，IGRS 版本号
01-IGRSMessageType：CreateSessionRequest	必备字段，消息类型
01-TargetDeviceId：目标设备标识符	必备字段，服务所在设备标识符
01-SourceDeviceId：源设备标识符	必备字段，本设备标识符
01-SequenceId：设备管道消息序列号	必备字段

Content-type：text/xml：charrset=utf-8	必备字段
Content-length：消息体长度	必备字段，消息体的长度
Man：“http：//schemas.xmlsoap.org/soap/envelope/”；ns=02	必备字段
02-SoapAction：”IGRS-CreateSession-Rsquest”	必备字段
	空行，隔开消息头和消息体
<SOAP-ENV：Envelopexmlns：SOAP-ENV="http：//schemas.xmlsoap.org/soap/envelope/" SOAP-ENV：encodingStyle="http：//schemas.xmlsoap.org/soap/encoding/">	必备字段
<SOAP-ENV：Body>	必备字段
<Session xmlns=“http：//www.igrs.org/spec1.0”>	必备字段
<SourceClientId> 源客户标识符 </SourceClientId>	必备字段，客户标识符
<TargetServiceId> 目标服务标识符 </TargetServiceId>	必备字段，服务标识符
<SequenceId> 会话创建请求序列号 </SequenceId>	必备字段
<UserInfo>	必备字段
<SourceUserId> 用户标识符 </SourceUserId>	必备字段
<ServiceSecurityId> 服务安全机制描述符 </ServiceSecurityId>	必备字段，服务支持的安全机制
<Token> 用户生成的Token的内容 </Token>	必备字段，生成的 Token 字符串
</UserInfo>	必备字段
</Session>	必备字段
</SOAP-ENV：Body>	必备字段
</SOAP-ENV：Envelope>	必备字段

表 4-3　一般会话创建请求消息

IGRS 中交互的消息是对 SOAP 的扩展，消息分为消息头和消息体两个部分，消息体和消息头用一个空行隔开。消息头是一行一个字段，每一行结束都需要加上“\r\n”（回车换行符）来表明，最后用一个空行表示消息头的结束。因此在解析的时候用“\r\n\r\n”（连续两组的回车换行符）来确认消息头的结束处。消息体是一个字符串，所有的字段按照表中的顺序排列即可。

在消息头中有一个字段是“Content-length：消息体长度”，它要求填上消息体的长度，所以在实际操作中先生成消息体，统计完消息体的长度后再生成对应的消息头，最后完成消息头和消息体的封装（封装后消息头在前，消息体在后）并用 Socket 发送目标设备。

发送完请求消息后，本设备会保存该会话的信息，包含的内容是表 4-1 所列出的，但其中 SucceedFlag 字段设置为不成功，表明该会话请求消息已经发出，但是成功的响应消息还没有到达。

4.1.4 请求消息的验证

前面三步完成后，客户端就将请求消息发送到服务所在的设备上，接下来就是服务端对请求消息进行验证。验证的步骤如下：

1) 基本字段验证

首先对请求消息的字段进行基本的验证，即验证必备字段是否存在；在进行基本验证的同时获取其中的一些基本信息，如 SourceClientId，SourceUserId，TargetServiceId。

2) 该服务是否存在的验证

将请求消息中的 TargetServiceId 作为关键字搜索本地服务验证本设备上是否有该服务存在：没有检测到就直接跳到 4.1.5 生成会话创建失败的响应消息，检测到有该服务存在则进行下一步的验证。

3) 该服务已建会话数是否已满的验证

在 IGRS 中规定，一个服务只能同时和有限个客户端建立会话（在程序中设置为 5）。本步骤完成对该服务已经建立好的会话数目的验证，如果该数目已经达到 5，则直接跳往 4.1.5 生成会话创建不成功的响应消息，小于 5 的话则执行下一步验证。

4) 管道是否存在的验证

在对管道的叙述中提到，IGRS 的交互除了多播查找和单播响应这一模式不需要管道的支持，其它所有的交互都是在管道的基础上进行的。所以，这一步是验证这两个设备间是否有管道存在。验证只要将对方设备的 DeviceId 作为关键字遍历搜索本地存储的与其他设备建立的管道列表，找到有管道存在则至下一步，否则直接跳往 4.1.5 生成会话创建不成功的响应消息。

5) 安全机制的验证

在来自客户端的请求消息中有一个字段是“<ServiceSecurityId>服务安全机制描述符</ServiceSecurityId>”，该字段表明了客户端认为服务采用的安全机制方式。为了防止客户端故意降低安全机制非法访问，服务端需要检验该服务的相关信息，将本地设置的该服务的安全机制和请求消息中的安全机制做一个比较，若两者一样则进行下一步验证，不一样的话就直接跳往 4.1.5 生成会话创建不成功的响应消息。

6) 客户身份的验证

对于不同的安全机制，身份验证的具体方式是不一样的，但原理都是一致的。在请求消息中有一个字段是 Token，该字段包含了请求客户的身份信息。当服务端得到请求消息后，会根据相应的安全机制，按照客户端生成 Token 的方式重新生成一个 Token，

然后将这个 Token 和客户端传来的 Token 进行比较验证，仅当二者完全一致时服务端才能认为客户端传来的 Token 是有效的，从而实现了对该用户的身份的验证；否则验证工作直接结束，直接跳往 4.1.5 生成会话创建不成功的响应消息。

7) 对方设备与用户权限的验证

每一个服务都有自己的服务描述报文，其中对访问与调用该服务的设备及用户提出了限制。所以，服务端也要检验访问自己的设备与用户是否真正满足以上要求。当且仅当设备与用户均满足服务描述报文中的规定，对方设备与用户属性的验证才能通过，可以执行下一步的工作；否则验证工作直接结束，直接跳往 4.1.5 生成会话创建不成功的响应消息。

8) 生成并保存该会话信息

上述的验证工作均进行结束并且结果全是符合的话，服务所在的设备会将该用户和设备的信息记录下来，信息和请求方保存的是完全对应的，不同的是 SucceedFlag 字段是设置为成功的，表明本服务和该客户间已经建立了一个成功的会话。该信息一直保存，直到该会话拆除信息才会被删除。

4.1.5 响应消息的生成

不论对客户端的请求消息的认证是否通过，服务端都要基于管道给请求的客户端发送一个会话创建的响应消息。响应消息是和请求消息对应的，详细内容如下：

会话创建响应消息	消息字段说明
HTTP/1.1 200 OK	扩展 HTTP 命令行
Ext:	必备字段
Cache-control: no-cache="Ext"	必备字段
MAN: "http: //www.igrs.org/session"; ns=01	必备字段
01-IGRSVersion: IGRS/1.0	必备字段，IGRS 版本号
01-IGRSMessageType: CreateSessionResponse	必备字段，消息类型
01-TargetDeviceId: 目标设备标识符	必备字段，服务所在设备标识符
01-SourceDeviceId: 源设备标识符	必备字段，本设备标识符
01- AcknowledgeId: 设备管道消息序列号	必备字段
Content-type: text/xml: charrset=utf-8	必备字段
Content-length: 消息体长度	必备字段，消息体的长度
Man: "http: //schemas.xmlsoap.org/soap/envelope/"; ns=02	必备字段
02-SoapAction: "IGRS-CreateSession-Rsponse"	必备字段
	空行，隔开消息头和消息体
<SOAP-ENV : Envelopexmlns : SOAP-ENV="http : //schemas.xmlsoap.org/soap/envelope/" SOAP-ENV : encodingStyle="http :	必备字段

//schemas.xmlsoap.org/soap/encoding/">	
<SOAP-ENV：Body>	必备字段
< DeviceOperation xmlns="http：//www.igrs.org/spec1.0">	必备字段
<TargetClientId> 源客户标识符 </TargetClientId>	必备字段，客户标识符
<SourceServiceId> 目标服务标识符 </SourceServiceId>	必备字段，服务标识符
<TargetUserId> 用户标识符 </TargetUserId>	必备字段，用户标识符
< AcknowledgeId > 会话创建请求序列号 </ AcknowledgeId >	必备字段
<ReturnCode> 创建会话过程的响应状态码 </ReturnCode>	必备字段，
</ DeviceOperation>	必备字段
</SOAP-ENV：Body>	必备字段
</SOAP-ENV：Envelope>	必备字段

表 4-4 一般会话创建响应消息

在该表中，同请求消息对应的字段的产生方式都是一样的，唯一的区别是在响应消息中有一个新的字段叫 ReturnCode。ReturnCode 代表了会话是否创建成功，如果创建失败了，也需用不同的 ReturnCode 表明不同的失败原因。在 IGRS 中，ReturnCode 为 100 表明会话创建成功，失败以及对应的代码如下规定：

错误代码	错误原因
400	会话创建失败，鉴别失败
401	目标服务不存在
402	用户权限不符
403	设备权限不符
404	并发数已满，会话建立失败
405	设备管道创建失败，鉴别失败
406	设备管道创建失败，目标设备不存在

表 4-5 会话创建响应消息中的错误代码定义

当响应消息的所有字段都已经具备，就通过 TCP Socket 将响应消息发送给请求设备。

4.1.6 响应消息的处理

客户端所在的设备发送完一个会话创建的请求后，就会启动一个定时器，期待在定时器超时之前能够收到对方的会话响应消息，根据 IGRS 规定，该定时器的超时时间为 30 秒。

若在 30 秒之内，客户端没有收到来自服务端的响应，就会触发定时器超时函数。会话创建的超时函数将会释放掉存储的和该会话有关的资源，并且向服务端发送一个会

话拆除的通知消息。这里我们没有采用只是简单的删除该会话的信息，而是发送了一个会话拆除的通知给服务端，原因在于：客户端收不到响应消息有两种可能，一种是请求消息没有正确的发送到服务端，另一种是服务端收到了请求消息也做了处理，但是响应消息没有正常的发送回客户端。对于这两种不同的情况在服务端引起的处理是不一样的，为了防止后一种情况的发生，客户端还是发送了一个会话拆除的通知消息给服务端，至于在第一种情况下服务端收到该会话拆除通知消息则不作任何处理。

正常的情况下，客户端能够在很快的时间内收到来自服务端的响应消息。除了一般的报文验证外，客户端主要查看的是响应消息中的 ReturnCode，对不同的 ReturnCode 作出不同的处理。如果 ReturnCode 为 100，客户端就会将原先保存的该会话信息中的 SucceedFlag 置为成功，否则认为本次会话创建失败那么就会删除原先保存的没有建成功的该会话的信息，并将错误信息返回给上层（ReturnCode 为 403 时除外，处理方式见 4.2 节）。

4.1.7 会话的维持

会话建立成功后就会一直存在，通过会话客户才能对该服务进行调用。在每一次客户发送服务调用请求前，客户端设备都会检查该客户与目标服务间是否有相应的会话存在，若有则直接发送服务调用请求，没有的话则必须按照前面所述述的步骤和目标服务先建立起会话，当会话建立好后才将服务调用请求发出。服务端也会根据会话的状态来对调用自己的客户进行限制。

4.1.8 会话的拆除

一个服务同时建立的会话的数目是有限制的，所以当某个用户长期不调用该服务时应当拆除相应的会话资源，以便其他用户能够调用该服务的资源。不同于会话的建立过程，客户端和服务端都可以发起会话的拆除通知消息，而且发起的一方只需要向对方发送一个会话拆除的通知消息，不需要经过对方的响应即可认为会话已经拆除，同样接收到该通知的设备也会立刻拆除相关的会话。

由于发起方不一样，会话拆除通知的信息内容会有细小的差别，当会话拆除通知是由客户端发起的，它的消息内容为：

客户端发起的会话拆除通知消息	字段的解释
M-NOTIFY /IGRS HTTP/1.1	HTTP 命令行

HOST：主机 IP 地址：端口	必备字段，IP 地址
MAN："http：//www.igrs.org/session"；ns=01	必备字段
01-IGRSVersion：IGRS/1.0	必备字段，版本号
01-IGRSMessageType：DestroySessionNotify	必备字段，消息类型
01-SourceDeviceId：源设备标识符	必备字段，本设备标识符
01-TargetDeviceId：目标设备标识符	必备字段，远程设备标识符
Content-type：text/xml； charset=utf-8	必备字段
Content-length：消息体长度	必备字段，消息体长度
MAN："http：//schemas.xmlsoap.org/soap/envelope/"；ns=02	必备字段
02-SoapAction："IGRS-DestroySession-Notify"	必备字段，消息类型
	空行，隔开消息头和消息体
<SOAP-ENV：Envelope xmlns：SOAP-ENV="http：//schemas.xmlsoap.org/soap/envelope/" SOAP-ENV：encodingStyle="http：//schemas.xmlsoap.org/soap/encoding/">	必备字段
<SOAP-ENV：Body>	必备字段
<Session xmlns="http：//www.igrs.org/spec1.0">	必备字段
<SourceClientId>源客户标识符</SourceClientId>	必备字段，表明是客户端发出的
<TargetServiceId> 目标服务标识符</ TargetServiceId >	必备字段，目标服务的标识符
<Token> 本会话的 Token </Token>	必备字段，建立会话时的 Token
</Session>	必备字段
</SOAP-ENV：Body>	必备字段
</SOAP-ENV：Envelope>	必备字段

表 4-6 客户端发出的会话拆除通知

当发起方是服务端时，会话拆除的通知消息为：

服务端发起的会话拆除通知消息	字段的解释
M-NOTIFY /IGRS HTTP/1.1	HTTP 命令行
HOST：主机 IP 地址：端口	必备字段，IP 地址
MAN："http：//www.igrs.org/session"；ns=01	必备字段
01-IGRSVersion：IGRS/1.0	必备字段，版本号
01-IGRSMessageType：DestroySessionNotify	必备字段，消息类型

01-SourceDeviceId：源设备标识符	必备字段，本设备标识符
01-TargetDeviceId：目标设备标识符	必备字段，远程设备标识符
Content-type：text/xml； charset=utf-8	必备字段
Content-length：消息体长度	必备字段，消息体长度
MAN："http：//schemas.xmlsoap.org/soap/envelope/"；ns=02	必备字段
02-SoapAction："IGRS-DestroySession-Notify"	必备字段，消息类型
	空行，隔开消息头和消息体
<SOAP-ENV：Envelope xmlns：SOAP-ENV="http：//schemas.xmlsoap.org/soap/envelope/"SOAP-ENV：encodingStyle="http：//schemas.xmlsoap.org/soap/encoding/">	必备字段
<SOAP-ENV：Body>	必备字段
<Session xmlns="http：//www.igrs.org/spec1.0">	必备字段
<TargetClientId>源客户标识符</TargetClientId>	必备字段，目标客户端标识符
<SourceServiceId>目标服务标识符</SourceServiceId>	必备字段，本服务的标识符
<Token>本会话 Token</Token>	必备字段，建立会话时的 Token
</Session>	必备字段
</SOAP-ENV：Body>	必备字段
</SOAP-ENV：Envelope>	必备字段

表 4-7 服务端发出的会话拆除通知

4.2 同一设备组内两设备间会话的实现

当某个服务的访问安全控制要求对其请求的客户所在设备与提供该服务的设备之间需建立一个安全的设备管道，但两设备间已经建立的设备管道不符合该要求，在一般情况下客户与服务之间是不能建立起会话的。但是，如果这两个设备是属于同一个主从设备组的从设备时，它们就能在该设备组的主设备的帮助下建立起同一设备组间的会话。

建立这种特殊的会话，开始的工作是和一般会话相同的，即客户端发送一个普通的创建请求给目标服务所在的设备，在这种情况下会得到一个会话创建失败的响应消息，失败的原因是“设备访问权限不符”，即 ReturnCode 为 403。

4.2.1 向主设备申请会话密钥

客户端收到这个响应后并不是直接将错误告知上层，而是试图通过所属设备组的主设备来获取一个会话用的密钥。客户从服务端的服务描述所支持加密算法列表中选择适合己方的相应基于可信赖第三方的服务安全机制描述符，将客户自己的设备标识符、服务提供端的设备标识符、加密算法封装成一个向主设备申请会话加密密钥生成的请求消息，发给主设备。该消息内容如下：

向主设备申请会话密钥的消息	字段解释
M-POST /IGRS HTTP/1.1	扩展 HTTP 命令行
HOST： 目标主机 IP：端口	必备字段
MAN："http：//www.igrs.org/session"；ns=01	必备字段
01-IGRSVersion：IGRS/1.0	必备字段，IGRS 版本号
01-IGRSMessageType：ApplySessionKeyRequest	必备字段，消息类型
01-TargetDeviceId：目标主设备标识符	必备字段
01-SourceDeviceId：源设备标识符	必备字段
01-SequenceId：设备管道消息请求序列号	必备字段
Content-type：text/xml； charset=utf-8	必备字段
Content-length：消息体长度	必备字段
MAN："http：//schemas.xmlsoap.org/soap/envelope/"；ns=02	必备字段
02-SoapAction："IGRS-ApplySessionKey-Request"	必备字段
	空行，隔开消息头和消息体
<SOAP-ENV：Envelope xmlns：SOAP-ENV="http：//schemas.xmlsoap.org/soap/envelope/"SOAP-ENV：encodingStyle="http：//schemas.xmlsoap.org/soap/encoding/">	必备字段
<SOAP-ENV：Body>	必备字段
<Session xmlns="http：//www.igrs.org/spec1.0">	必备字段
<SourceClientId> 源客户标识符 </SourceClientId>	必备字段
<TargetServiceId> 目标服务标识符 </TargetServiceId>	必备字段

<SequenceId> 请求会话加密密钥消息序列号 </SequenceId>	必备字段
<ServiceHOSTDeviceId>目标服务所在设备序列号</ ServiceHOSTDeviceId >	必备字段
<ServiceSecurityId> 服务安全机制描述符 </ServiceSecurityId>	必备字段
</Session>	必备字段
</SOAP-ENV：Body>	必备字段
</SOAP-ENV：Envelope>	必备字段

表 4-8 向主设备申请会话密钥的请求消息

4.2.2 主设备生成会话密钥

当主设备收到来自客户端的请求后，经过必要的消息解析，就会根据请求消息中的安全机制生成一个随机比特串 RandomString（用随机数生成器即可生成）作为客户端和服务端建立会话用的密钥。在同一主从设备组内，任意的从设备和主设备间都有预共享密钥，主设备就用与客户端设备之间的预共享密钥对该 RandomString 进行加密得到密文 Cipher1，同时也将用主设备与服务提供端设备之间预共享密钥和加密算法对{服务标识符，客户标识符，客户和目标服务支持的基于可信赖第三方的服务安全机制描述符，会话加密密钥 RandomString}进行加密形成密文 Cipher2，最后将 Cipher1 和 Cipher2 通过响应消息的方式传给客户端。该响应消息的定义如下：

主设备发送会话密钥的消息	字段解释
HTTP/1.1 200 OK	HTTP 命令行
Ext：	必备字段
Cache-control：no-cache="Ext"	必备字段
MAN："http：//www.igrs.org/session"；ns=01	必备字段，IGRS 版本号
01-IGRSVersion：IGRS/1.0	必备字段，消息类型
01-IGRSMessageType：ApplySessionKeyResponse	必备字段
01-SourceDeviceId：源设备标识符	必备字段
01-TargetDeviceId：目标设备标识符	必备字段
01-AcknowledgeId：设备管道消息序列号	必备字段

Content-type：text/xml； charset=utf-8	必备字段
Content-length：消息体长度	必备字段
MAN："http://schemas.xmlsoap.org/soap/envelope/"; ns=02	必备字段
02-SoapAction："IGRS-ApplySessionKey-Response"	必备字段
	空行，隔开消息头和消息体
<SOAP-ENV：Envelope xmlns：SOAP-ENV="http：//schemas.xmlsoap.org/soap/envelope/"SOAP-ENV：encodingStyle="http：//schemas.xmlsoap.org/soap/encoding/">	必备字段
<SOAP-ENV：Body>	必备字段
<Session xmlns="http：//www.igrs.org/spec1.0">	必备字段
<SourceClientId> 源客户标识符 </SourceClientId>	必备字段
<TargetServiceId> 目标服务标识符 </TargetServiceId>	必备字段
<ReturnCode> 会话加密密钥获取响应状态码 </ReturnCode>	必备字段
<AcknowledgeId>会话创建请求序列号</AcknowledgeId>	必备字段
<Cipher1>主设备发来的 Cipher1 的内容</Cipher1>	必备字段
<TargetServiceHOSTDeviceId>目标服务所在从设备标识符</TargetServiceHOSTDeviceId>	必备字段
<TargetServiceId> 目标从设备上的服务标识符 </TargetServiceId >	必备字段
<Cipher2>主设备发来的Cipher2的内容</Cipher2>	必备字段
</Session>	必备字段
</SOAP-ENV：Body>	必备字段
</SOAP-ENV：Envelope>	必备字段

表 4-9 主设备分发会话密钥的消息

4.2.3 向服务端发送会话密钥

客户端成功收到主设备的响应消息后，首先用自己与主设备间的预共享密钥对 Cipher1 解密得到会话密钥 RandomString，然后通过消息将 Cipher2 发送给服务端设备：

向服务端发送会话密钥	字段解释
M-POST /IGRS HTTP/1.1	扩展 HTTP 命令行
HOST： 目标主机 IP：端口	必备字段

MAN：”http：//www.igrs.org/session”；ns=01	必备字段
01-IGRSVersion：IGRS/1.0	必备字段，IGRS 版本号
01-IGRSMessageType：TransferSessionKeyRequest	必备字段，消息类型
01-TargetDeviceId：目标主设备标识符	必备字段
01-SourceDeviceId：源设备标识符	必备字段
01-SequenceId：设备管道消息请求序列号	必备字段
Content-type：text/xml；charset=utf-8	必备字段
Content-length：消息体长度	必备字段
MAN：”http：//schemas.xmlsoap.org/soap/envelope/”；ns=02	必备字段
02-SoapAction：”IGRS-TransferSessionKey-Request”	必备字段
	空行，隔开消息头和消息体
<SOAP-ENV：Envelope xmlns：SOAP-ENV="http：//schemas.xmlsoap.org/soap/envelope/"SOAP-ENV：encodingStyle="http：//schemas.xmlsoap.org/soap/encoding/">	必备字段
<SOAP-ENV：Body>	必备字段
<Session xmlns=“http：//www.igrs.org/spec1.0”>	必备字段
<SourceClientId>源客户标识符</SourceClientId>	必备字段
<TargetServiceId>目标服务标识符</TargetServiceId>	必备字段
<SequenceId> 请求会话加密密钥消息序列号 </SequenceId>	必备字段
<Trusted3rdDeviceId>目标服务所在设备序列号</ Trusted3rdDeviceId>	必备字段
<Cipher2> Cipher2 的内容</Cipher2>	必备字段
</Session>	必备字段
</SOAP-ENV：Body>	必备字段
</SOAP-ENV：Envelope>	必备字段

表 4-10 向服务端发送会话密钥

4.2.4 服务端确认会话密钥

服务端设备收到客户端发来的 Cipher2，就会用自己与主设备间的预共享密钥对 Cipher2 解密，通过解析 Cipher2 的内容实现对客户端的认证，同时服务端也得到了 RandomString。最后服务端将会发送一个成功的响应给客户端设备，表明通过了对其的身份验证。

通过上述的几个步骤，客户端和服务端都掌握了主设备生成的密钥 RandomString，客户端就将 RandomString 作为对称密钥再次向服务端发送一个会话创建请求，情况同一般会话中“基于对称密钥的安全机制”是一样的。这样服务端再次对 Token 进行验证和客户端建立起会话。至此，该客户端和服务端的会话就建立成功，该会话的维持和拆除工作是和一般会话相同的，这里不再阐述。

4.3 会话调试过程

会话的调试主要是进行单元功能的调试。从图 4-1 中可以看出，在 IGRS Stack 中的会话模块既要能完成作为客户端的功能，也要能够完成作为服务端的功能。因此在测试中我们采用两个方向分别进行的方式。

会话的测试主要是看两个指标：一个是交互的报文的正确性，主要是关键字段的检验，一个是会话信息列表的正确性，这个通过屏幕显示和写成文本的方式来测试。

当测试会话作为客户端主动发起的功能时，测试平台模拟服务端进行响应。对于得到创建成功的响应，程序能够及时的将信息进行保存下来并且显示创建成功，通过 SucceedFlag 字段可以查看；对于创建不成功的响应应当删除该会话的信息，并且正确的发送出会话拆除的通知消息；对于规定时间内没有收到响应也需完成与创建不成功同样的操作。

测试会话作为服务端时，测试平台会主动发送会话创建请求的消息。此时测试平台会发送各种各样的请求消息，正确的与错误的，主要是测试会话能否正确的发送相应的 ReturnCode。

前面文章指出，会话是在管道之上的，因此当两个设备的管道断开后它们之间所有的会话也应该全部拆除。测试这个功能就要和管道模块、设备模块进行集成测试，通过测试的指标同样是报文的正确性与会话信息列表的正确性。

在调试的过程中发现以下问题：当客户端进程由于异常原因退出时，它不会向协议

栈发出注销的信息，因此协议栈也不会拆除与该客户端相关的会话信息。当客户端再次启动并注册到协议栈时，协议栈依然会向服务端发送建立会话的请求。但是，在调试中发现服务端（“闪联之家”软件）不允许同一台设备上的客户没有拆除会话就试图再次建立新的会话，所以这样就造成了无法调用该服务信息。所以对协议栈进行了改动：当每一次客户端来注册时，协议栈都会检测是否已经有与该客户端相关的会话的信息存在，如果有则先发送会话拆除的通知消息然后再新建会话，如果没有则直接发送会话建立请求消息。通过这样的改动，不管客户端进程是否正常退出，每一次启动都能顺利的与服务端建立起会话；唯一区别是当客户端进程正常退出时，会话拆除的消息能及时发送给服务端，异常退出时则不能。

4.4 本章小结

本章对 IGRS 会话机制的实现进行了详尽的分析，全面细致的论述了 IGRS 会话的目的、建立的方法、建立的步骤、建立的原则。

对于一般会话的的创建、维持、拆除的每一步都作了说明，并且对其中涉及到消息中的每一个重要的字段都作了解释与说明。对于主从设备组中设备间的会话，着重对其与一般会话不同的地方进行了说明。

最后简单的叙述了会话的调试原理。

第五章 总结与展望

在 TCL 研究院一年的课题研究中，笔者经历了从项目的总体设计到“闪联”电视的第一个版本的面世，在最后将所做的工作总结如下：

5.1 取得的成绩

笔者从 2005 年 10 月加入 IGRS 项目组，首先参与其中的 IGRS 基础协议栈的实现工作。最初的总体设计工作对 IGRS 基础协议进行了详尽的分析与研究，并且将其划分为若干功能独立的模块，包括设备模块、管道模块、会话模块、安全模块、通信模块等；随后在详细设计中承担了会话模块、安全模块的全部设计工作。设计工作在 2006 年 2 月结束，接下来参与了 IGRS 基础协议栈代码的编写，除完成会话模块、安全模块的全部编写工作外，还负责了通信模块、服务模块、设备组模块的检查、维护与修改工作。

IGRS 基础协议栈于 2006 年 5 月底诞生 Version1.0，随后本人负责带着该协议栈两次去北京联想研究院对两家的协议栈进行了互通的调测，在一共为期 3 周的调试工作后，又全权负责对 IGRS 协议栈进行修改。

在 IGRS 协议栈开发结束后，笔者又参与了 IGRS 服务应用框架项目的开发工作，参与其中的总体设计、详细设计以及部分编码的工作，完成了实现基本的 IGRS AV 框架的 IGRS 客户端进程。

IGRS Stack 和 IGRS 客户端进程是两个独立的进程，两者同时运行在一个设备上。在 TCL 的项目中，采用了 TI 的最新的技术“Davinci”作为开发板，在 2006 年 9 月，闪联电视的 Demo 版本面世。

目前闪联电视能够通过有线或无线的方式同联想的安装“闪联之家”软件的电脑自动快速的发现。通过会话机制的支撑，闪联电视能调用“闪联之家”提供的服务，能够将“闪联之家”所在的电脑上的电影、音乐、图片在闪联电视上播放和显示。

5.2 今后的工作

从整个数字家庭技术来看，闪联电视只是其中的一个小小的方面，还有很多后续的工作需要完成：

首先是标准的完善。IGRS 已经成为国家的标准，即将成为一个国际标准，截至笔

者结束项目之前，IGRS 基础协议 V0.95 版还存在不少不完善的地方，比如对错误字段的代号还需要进一步添加；设备组的实现也是一个重要的方面，在当前的协议栈中，设备组的应用没有得到重视，但是以后家庭设备的数量会越来越多，对这些设备进行分组管理是必不可少的。

其次是标准的开发。IGRS 基础协议在整个 IGRS 体系中是最基础的，需要不断的增强其功能，对于处在上层的 IGRS 应用和 IGRS 特色应用则需要不断的开发新的应用。目前，IGRS AV 框架的首个版本在闪联电视上已经应用，但是这些只是 IGRS 应用中最最基础的，要想 IGRS 能够广泛的应用，就需要开发出更多的特色应用来满足用户的需要。

最后是不同标准之间的兼容问题。从目前来看，未来的数字家庭不会只有一个标准，但是支持不同标准的设备必须能够互联才能满足用户的要求。在 IGRS 的开发中，需要注意其与诸如 DLNA、“e 家佳”等设备的互联，真正实现数字家庭中所有设备的互联。

附录一：程序清单

源文件及头文件：

- ILibAsyncSocket.c 异步 Client Socket 源文件
- ILibAsyncSocket.h 异步 Client Socket 头文件
- ILibAsyncServerSocket.c 异步 Server Socket 源文件
- ILibAsyncServerSocket.h 异步 Server Socket 头文件
- ILibSSDPClient.c SSDP 客户端源文件
- ILibSSDPClient.h SSDP 客户端头文件
- ILibWEBClient.c WEB 客户端源文件
- ILibWEBClient.h WEB 客户端头文件
- ILibWEBServer.c WEB 服务端源文件
- ILibWEBServer.h WEB 服务端头文件
- ILibParsers.c 解析函数库源文件
- ILibParsers.h 解析函数库头文件
- ILibSecurity.c 安全模块源文件
- ILibSecurity.h 安全模块头文件
- ILibUtility.c 通用接口函数源文件
- ILibUtility.h 通用接口函数头文件
- ILibLog.c 日志文件模块源文件
- ILibLog.h 日志文件模块头文件
- comIGRS.c 进程通信模块源文件
- comIGRS.h 进程通信模块头文件
- IGRSDev.c 设备模块源文件
- IGRSDev.h 设备模块头文件
- IGRSDevGrp.c 设备组模块源文件
- IGRSDevGrp.h 设备组模块头文件
- IGRSMicroStack.c 消息主处理模块源文件
- IGRSMicroStack.h 消息主处理模块头文件
- IGRSPipeLine.c 设备管道模块源文件

- IGRSPipeLine.h 设备管道模块头文件
- IGRSService.c 服务模块源文件
- IGRSService.h 服务模块头文件
- IGRSSession.c 会话模块源文件
- IGRSSession.h 会话模块头文件
- IGRSDevSubEvent.c 设备事件订阅模块源文件
- IGRSDevSubEvent.h 设备事件订阅模块头文件
- IGRSServSubEvent.c 服务事件订阅模块源文件
- IGRSServSubEvent.h 服务事件订阅模块头文件
- Main.c 主函数源文件
- Main.h 主函数头文件

运行所需文件：

- IGRS.exe（由上述源文件生成的可执行文件）
- IGRS.ini（配置文件）
- DeviceDesc.xml（设备描述报文）

附录二：攻读硕士学位期间参与的项目与撰写的论文

参与的项目：

- 2005 年广东省关键领域重点突破项目--“数字家庭公共技术支撑平台”下的子项目 IGRS 项目

撰写的文章：

- 夏宏春 张勤 鄢广增 “浅谈 IGRS 安全” 《中国新通信》 2006 年 17 期，已发表

致 谢

随着论文工作的完成，我的研究生生活也即将结束。在我读研期间，周围的老师和同学给了很多鼓励和支持，有了他们的帮助我才能顺利完成学业。

首先衷心感谢我的导师酆广增教授。三年来，他在平时生活中关心照顾我，在学术科研时悉心指导我，在遇到困题时启发帮助我。他那孜孜以求的学习精神和严谨认真的治学态度令我感到由衷的敬佩。酆老师言传身教，使我在科研能力和学术水平上都有了较大的提高，这将是我一生中最宝贵的财富。在今后的工作学习中，我将铭记酆老师的教诲，争取更大的进步。

其次衷心感谢我在 TCL 研究院的导师张勤教授，是张教授让我有幸参加到 IGRS 项目的开发中来。在一年的课题研究中张教授给了我悉心的指导，对于工作中遇到的问题进行详细的指点。在此对张勤教授表示诚挚的谢意。

真心感谢 IGRS 项目组的项目经理孙翀博士，在一年的工作中孙博士给了我很大的帮助，使得我在一年的实践中得到了很大的提高。

感谢张晓宇同学，在 TCL 研究院的实习的一年中无论是工作上还是生活上都给我提供了很大的帮助，他让我在深圳的一年能够快乐充实的度过，丝毫没有感到孤单。

感谢 IGRS 项目组的所有成员，他们在工作上中给了我很多帮助，让我顺利的完成了在 TCL 研究院的实习工作。在此，向他们表示深深的谢意。

同时，感谢我的同门夏飞、吴淑花、孔媛媛、董欣、肖亮、张琛、陈君。他们在生活中给了我很多帮助，在学习上给我提了很多宝贵的建议，我们一起度过了人生中最快乐美好的时光。衷心祝福他们一帆风顺、事业有成。

感谢我的家人。有了他们的照顾和支持，我才能全身心投入学习科研。

最后，还要感谢审稿的各位专家，以及参加我硕士论文答辩的评委和同学。

参考文献

[1] 《DLNA Overview and Vision White Paper》

[2] 苗在良 “数字家庭网络现状和发展趋势”,《信息技术与信息化》2005(3)

[3] 1999-2000UPnP 论坛贡献成员 《UPnP 协议文档》中文版，2000.6

[4] 冯承文，钟丽静 “E 家佳数字家庭系统”，《电子产品世界》， 2006(8)

[5] IGRS 标准工作组 《信息设备资源共享协同服务（IGRS）基础协议送审稿》2004.7

[6] 联想（北京）有限公司 《IGRS AV 应用框架 V0.92》

[7] 徐刚,邓中亮,杨军 “IGRS 和 UPnP 协议互联机制分析”,《电子设计应用》2006(1)

[8] 廖国威,杨军,邓中亮“IGRS 基础协议中安全机制浅析”,《中国集成电路》2005(11)

[9] 阙喜戎，孙锐，龚向阳，王纯 《信息安全原理和应用》，清华大学出版社