UPnP 协议的分析及实现

随着计算机产业以及计算机网络技术的迅猛发展，使得嵌入式系统和家庭网络通信成为了热门的研究领域。由于越来越多的嵌入式设备的出现和家庭网络的发展，实现家庭网络中的各种嵌入式设备互联互通已经成为人们的迫切需求。

而实现家庭网络的关键是家庭网络中间件技术。现今世界各著名设备厂商纷纷提出了各自的新技术和解决方案，其中，微软提出的 UPnP 技术最有发展前途，得到了最广泛的支持，是当今各国研究的热点。UPnP 是通用即插即用(Universal Plug and Play)的缩写，它主要用于实现设备的智能互联互通。使用 UPnP 协议不需要设备驱动程序，它可以运行在几乎所有的操作系统平台之上，使得在办公室、家庭和其它公共场所方便地构建设备相互联通的网络环境。

本文介绍了 UPnP 所使用的基本协议（如 SSDP、GENA、SOAP 等），重点分析了 UPnP 实现的基本工作流程。然后，论文在剖析了当前最常用的 Intel SDK 的结构和功能后，以 TV 控制点和 TV 设备的开发为例，给出了如何应用该 SDK 实现 UPnP 设备和控制点的设计开发技术。最后，利用 WiresharkPortable 工具捕获数据包，对各流程传递的消息包进行了详尽分析。

目 录

1 引言

UPnP 全名是Universal Plug and Play，主要是微软在推行的一个标准。简单的来说，UPnP 最大的愿景就是希望任何设备只要一接上网络，所有在网络上的设备马上就能知道有新设备加入，这些设备彼此之间能互相沟通，更能直接使用或控制它，一切都不需要设定，完全的Plug and Play。

举个例子来说：

Mary在她的计算机中存储了大量数码相机拍摄的照片。当朋友Karen 来拜访时， Mary在起居室拿起与等离子电视机配套的红外线（IR）遥控器，从电视所显示的列表中挑选她感兴趣的照片，向Karen在电视屏幕上展示一下这些照片。这过程中就使用了UPnP协议。

2 UPnP的结构规范

2.1 UPnP的基本组件

服务、设备和控制点是UPnP网络的基本组件。其组件图如图1所示：

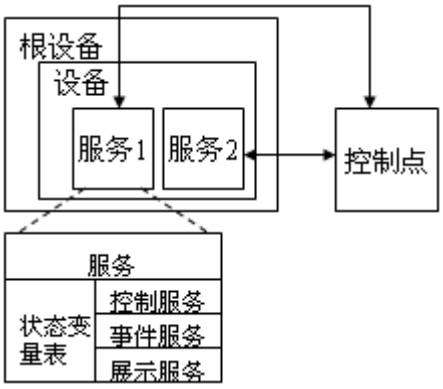

图1 UPnP组件图

● 服务（Service）

在UPnP网络中，最小的控制单元就是服务。服务描述的是设备在不同的情况下的活动和设备的状态。例如，时钟服务可以表述为时间变化(状态变化)、当前的时间(时钟的状态)以及设置时间和读取时间两个活动，通过这两个活动．你就可以控制服务。

● 设备(Device)

UPnP网络中定义的设备具有很广泛的含义，各种各样的家电、电脑外设、智能设备、无线设备、个人电脑等等都可以成为其中一员。一个UPnP设备可以是多个服务的载体和多个子设备的嵌套集。例如一台印表机有提供列印这样的服务；一台电视有提供收讯的服务，这些都属于设备。

● 控制点（ControlPoint）

在UPnP网络中，控制点指的是可以发现并控制其它设备的控制设备。在UPnP网络中，设备可以和控制点合并。也就是说，同一个设备，可以同时具有设备的功能和控制点的功能，即可以作为设备提供服务，也可以作为控制点发现和控制其它设备。

2.2 UPnP部分术语

● UUID

UUID含义是通用唯一识别码（Universally Unique Identifier），其目的是让分布式系统中的所有元素，都有唯一的辨识资讯，而不需要透过中央控制端来做辨识资讯的指定。其格式为xxxxxxxx-xxxx-xxxx-xxxxxxxxxxxxxxxx(8-4-4-16)，分别为当前日期和时间，时钟序列，全局唯一的IEEE机器识别号，如果有网卡，从网卡mac地址获得，没有网卡以其他方式获得。

● UDN

单一设备名（Unique Device Name），基于UUID，表示一个设备。在不同的时间，对于同一个设备此值应该是唯一的。

● URI

Web上可用的每种资源 - HTML文档、图像、视频片段、程序等 - 由一个通用资源标志符（Universal Resource Identifier, 简称"URI"）进行定位。 URI一般由三部分组成：访问资源的命名机制；存放资源的主机名；资源自身的名称，由路径表示。考虑下面的URI，它表示了当前的HTML 4.0规范：http://www.webmonkey.com.cn/html/html40/它表示一个可通过HTTP协议访问的资源，位于主机www.webmonkey.com.cn上，通过路径“/html/html40”访问。

● URL

URL是URI命名机制的一个子集，URL是Uniform Resource Location的缩写，译为“统一资源定位符”。通俗地说，URL是Internet上用来描述信息资源的字符串，主要用在各种www客户程序和服务器程序上。采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。

● URN

URN：URL的一种更新形式，统一资源名称(URN, Uniform Resource Name)。唯一标识一个实体的标识符，但是不能给出实体的位置。标识持久性 Internet 资源。URN可以提供一种机制，用于查找和检索定义特定命名空间的架构文件。尽管普通的URL可以提供类似的功能，但是在这方面，URN 更加强大并且更容易管理，因为 URN 可以引用多个 URL。

2.3 UPnP设备协议栈

UPnP定义了设备之间、设备和控制点、控制点之间通讯的协议。完整的UPnP由设备寻址、设备发现、设备描述、设备控制、事件通知和基于Html的描述界面几部分构成。UPnP设备协议栈如下图所示:

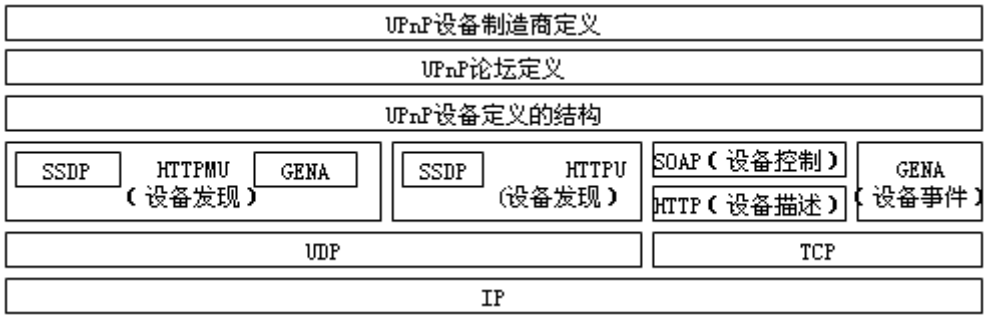

图2 UPnP协议栈[1]

UPnP协议结构最底层的TCP/IP协议是UPnP协议结构的基础。IP层用于数据的发送与接收。对于需要可靠传送的信息,使用TCP进行传送, 反之则使用UDP。UPnP对网络物理设备没有要求,可以使用以太网、无线网、IEEE1394、红外进行连接, 只要支持IP协议即可。同时UPnP还可以使用TCP/IP协议族中的其他协议, 如ARP、IGMP、DHCP、DNS等。

构建在TCP/IP协议之上的是HTTP协议及其变种,这一部分是UPnP协议的核心部分, 所有UPnP消息都被封装在HTTP协议及其变种之中。HTTP协议的变种是HTTPU和HTTPMU, 这些协议的格式沿袭了HTTP协议,只不过与HTTP协议不同的是它们通过UDP而不是TCP来发送消息,并且可以用于多播通信。

2.3.1 SSDP协议

简单服务发现协议（Simple Service Discovery Protocol：SSDP)，内建在HTTPU/HTTPMU 里，定义如何让网络上有的服务被发现的协议。包括控制点如何发现网络上有哪些服务，并取得这些服务的资讯，还有装置本身宣告他提供哪些服务。该协议运用在UPnP工作流程的设备发现部分。

2.3.2 SOAP协议

简易物件存取协议(Simple Object Access Protocol：SOAP)定义如何使用XML与HTTP来执行远端程序呼叫(Remote Procedure Call)。包括控制点如何发送命令消息给设备，及设备接收到命令消息后如何发送响应消息给控制点。该协议运用在UPnP工作流程的设备控制部分。

2.3.3 GENA协议

一般事件通知架构(Generic Event Notification Architecture：GENA)定义在控制点想要监听设备的某个服务状态变量的状况时，控制点如何传送订阅讯息

并如何接收通知讯息用的。该协议运用在UPnP工作流程的事件订阅部分。

2.4 基于XML的UPnP描述

2.4.1 XML简介[2]

XML（Extensible Markup Language）即可扩展标记语言，它与HTML一样，都是SGML(Standard Generalized Markup Language,标准通用标记语言)。XML是Internet环境中跨平台的，依赖于内容的技术，是当前处理结构化文档信息的有力工具。扩展标记语言XML是一种简单的数据存储语言，使用一系列简单的标记描述数据，而这些标记可以用方便的方式建立，虽然XML占用的空间比二进制数据要占用更多的空间，但XML极其简单易于掌握和使用。

与HTML类似，XML描述的内容封装在开始标签<标签名>和结束标签</标签名>之间，一对标签及其封装的内容，如<movie>Gone with the Wind<／movie>，被称为一个元素。元素可以嵌套，一个XML文档正是由许多这样的元素嵌套而成的。元素可以有属性，可赋予属性值。

在实际应用中，人们常常根据需要自定义元素名和属性名，这些名字具有明确易懂的含义。但是由于应用的繁多，所定义的名字很有可能发生冲突，为此XML引入了命名空间(namespace)的概念，它给出元素名和属性名定义的来源处，允许不同应用使用相同的名字，不致引起混淆。

XML命名空间采用“两段式命名法”定义所谓的“合法名称”，例如“学生：姓名”。其中第一段是指代特定命名空间的“命名空间前缀”，第二段是元素或属性的名字，两段之间用西文冒号“:”分隔。需要注意的是，“学生：姓名”和“班主任：姓名”虽然名称都是“姓名”，但却是两个不同的元素名，因为它们分别由“学生”和“班主任”命名空间定义。

命名空间用URI标识，具有唯一性和持久性，所谓命名空间前缀就是命名空间的缩写表示，XML采用下述“命名空间声明"来绑定命名空间前缀和命名空间：

xmlns：【命名空间前缀】=【命名空间名】

其中，xmlns就是XML命名空间的缩写。例如：

xmlns：学生=http://www.xml.net.cn/学生

xmlns：班主任=http://www.xml.net.cn/班主任

分别定义了命名空间前缀“学生”和“班主任”。

其后，XML就可以用命名空间和名的组合，即合法名称来无岐义地表示不同应用中的元素名和/或属性名，给出元素的描述和属性值的描述。例如：

<?xml version=”1.0”encoding=”GB23 12”?>

<学生：学生xmlns：学生=http://www.xml.net.cn/学生>

<学生：姓名>李明</学生：姓名>

〈学生：班级学生：数字类型=”中文”〉三年级二班〈/学生：班级〉

〈学生：住址学生：数字类型=”阿拉伯”〉135楼210室〈/学生：住址〉

〈/学生：学生〉

上述XML文档描述了关于学生的相关信息，所采用的元素名和属性名均源自“学生”命名空间。上面的描述使用了“学生”命名空间定义的4个元素名：学生、姓名、班级和住址，1个属性名：数字类型，并对于“学生：班级”元素的“数字类型”属性赋予属性值“中文”，对于“学生：住址″元素的“数字类型″属性赋予属性值”阿拉伯”。

类似地，可以用XML文档描述关于班主任的相关信息，只是需采用“班主任“命名空间定义的名字。只要不被其他命名空间声明所覆盖，命名空间声明对于它所说明的所有元素以及这些元素包含的所有内容都有效，这就是所谓命名空间的作用域范围。

2.4.2 TV设备的设备描述编写

在UPnP协议中，要实现控制点和设备之间的互相通信，设备的描述文件起着很重要的作用。对于设备的描述文件，是以XML文件形式存在的。在论文的第五节，我们会讨论TV设备的代码实现，在这我们就对TV设备的设备描述文件编写进行介绍。

一般来说，设备描述都是基于UPnP论坛上已定义的模板，这些标准化的模板提供一系列基本的服务和预定义的设备类型，厂商可以在其中作出自己的扩充。对于后面开发的TV设备，因为只是为了阐述UPnP协议的实现，所以自行开发其设备模板及其XML文档，以使其最简化。

根据UPnP规范，包括两个主要部分。第一部分包括根设备类型、特定厂商、制造商信息，如模块名称和编号、序列号、制造商名称、特定厂商网站URL等。第二部分包括设备所支持服务的信息。对于第一部分，为了使设备描述文件更简单，为简单计，省略了厂商名、厂商网址等可选元素，这些省略并不会对系统运行产生任何影响。下面就是TV设备的设备描述文件：

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
 <specVersion>
  <major>1</major>
  <minor>0</minor>
 </specVersion>
 <device>
  <deviceType>urn:schemas-upnp-org:device:tvdevice:1</deviceType>
  <friendlyName>UPnP Television Emulator</friendlyName>
  <UDN>uuid:Upnp-TVEmulator-1_0-1234567890001</UDN>
  <serviceList>
   <service>
    <serviceType>urn:schemas-upnp-org:service:tvcontrol:1</serviceType>
    <serviceId>urn:upnp-org:serviceId:tvcontrol1</serviceId>
    <controlURL>/upnp/control/tvcontrol1</controlURL>
    <eventSubURL>/upnp/event/tvcontrol1</eventSubURL>
    <SCPDURL>/tvcontrolSCPD.xml</SCPDURL>
   </service>
  </serviceList>
  <presentationURL>/tvdevicepres.html</presentationURL>
</device>
</root>
```

图3 TV设备的设备描述

2.4.2.1 基本信息编写

● 设备类型

设备类型元素的格式如下：

<deviceType>命名空间：设备类型：版本号</deviceType>

对于基于标准设备模板的设备，描述文件中这一属性为：

<...urn：schemas-upnp-org...>

我们把设备类型叫做tvdevice，版本号定为1，于是我们得到了一个完整的设备类型元素：

<deviceType>urn:schemas-upnp-org:device:tvdevice:1</deviceType>

● friendlyname

我们为设备取了一个相对简单的用户友好的别名:UPnP Television Emulator，于是得到了<friendlyname>元素：

<friendlyName>UPnP Television Emulator</friendlyName>

● UDN

接下来需设计的是唯一设备名字(UDN)，这是该实例的唯一标识符。为了保持唯一性，采用设备的名字和设备网卡的MAC地址组合而成。假设MAC地址为1234567890001，就可以把UDN元素写为：

<UDN>uuid:Upnp-TVEmulator-1_0-1234567890001</UDN>

2.4.2.1 设备服务编写

设备所含服务是设备描述的主体部分，它们体现为设备的功能。在设计时，可根据设备的功能映射确定可提供给控制点的服务。在这里，TV设备只定义一个control服务。

control服务负责处理TV的开关（power），调整音量（volume）及设置频道（channel），其在设备描述文档中的第一个元素是serviceType，其命名原则与deviceType类似，遵从一定的约定，定义为：

<serviceType>urn:schemas-upnp-org:service:tvcontrol:1</serviceType>

类似地，<serviceId>元素定义为：

<serviceId>urn:upnp-org:serviceId:tvcontrol1</serviceId>

另一需要定义的是服务描述文档的地址<SCPDURL>元素，这个元素非常重要，控制点就是通过这个地址来获取后续的服务描述文档。由于本设备服务描述文档与设备描述文档位于同一目录，名为tvcontrolSCPD.xml，因此该元素为：

<SCPDURL>/tvcontrolSCPD.xml</SCPDURL>

TV控制相关服务最后两个子元素是控制URL和事件URL。这两个URL是控制点发送动作请求和订阅请求的URL。定义为：

<controlURL>/upnp/control/tvcontrol1</controlURL>

<eventSubURL>/upnp/event/tvcontrol1</eventSubURL>

3 UPnP实现的工作流程

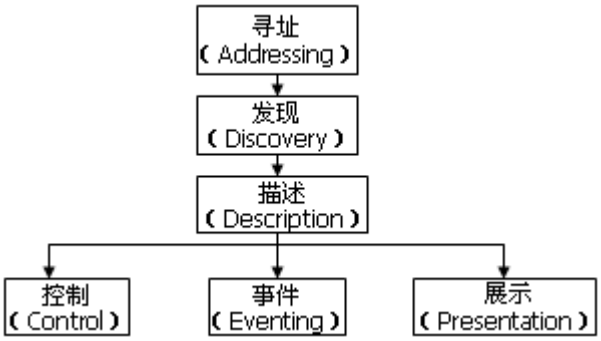

图4 UPnP实现的工作流程

稍微了解有哪些通讯协议后，我们来看UPnP是如何运作的。图4是UPnP的运作流程，我们先介绍各部分在做什么，再做详细介绍：

0. 控制点跟设备都先取得IP地址才能做之后的沟通。
1. 控制点寻找整个网络上的UPnP设备，而设备同时也要宣告他本身的存在。
2. 控制点取得设备的描述，这包括设备提供什么样的服务。
3. 控制点发出动作信息（对设备操作的命令信息）给设备。
4. 控制点监听设备的状态，当状态改变时做出相应的处理动作。

5. 控制点利用HTML界面来控制设备和监看设备状态。

3.1 寻址（Addressing）

UPnP网络的基础是TCP/IP协议族，这就决定了每一个UPnP组件（设备和控制点）必须分配一个IP地址。一个UPnP设备寻址的一般过程是：首先向 DHCP服务器发送DHCPDISCOVER消息，如果在指定的时间内，设备没有收到DHCPOFFERS回应消息，设备必须使用 Auto-IP完成IP地址的设置。在选中一个地址之后，设备测试此地址是否在使用。为了测试选择的地址是否未被占用，设备必须使用地址分辨协议（ARP）。

使用Auto IP的设备必须定时检测DHCP服务器是否存在，若存在，设备必须释放Auto IP分配的地址，此时设备必须取消所有的广告消息并重新发出新的。

一个设备可以使用UPnP之外的更高层的协议，这些协议将为设备使用友好的名称。在这种情况下，将这些友好的主机名解析为IP地址就很必要了，DNS通常是用来实现此功能的。使用此功能的设备可能要包含一个DNS客户端，而且支持动态的DNS注册，通过注册将它自己的名字加入到地址分布图中。

3.2 发现（Discovery）

一旦设备连接到网上并且分配了地址，就要进行发现的操作了。设备发现是UPnP网络实现的第一步。设备发现是由简单发现协议SSDP（Simple Service Discovery Protocol）来定义的。在设备发现操作之后，控制点可以发现感兴趣的设备，并使得控制点获得设备能力的描述，同时控制点也可以向设备发送命令，侦听设备状态的改变，并将设备展示给用户，即是描述、控制、监听、展示的基础。

当一个设备加入到网络中，设备发现过程允许设备向网络上的控制点告知它提供的服务。当一个控制点加入到网络中时，设备发现过程允许控制点寻找网络上感兴趣的设备。在这两种情况下，基本的交换信息就是发现消息。发现消息包括设备的一些特定信息或者某项服务的信息，例如它的类型、标识符、和指向XML设备描述文档的指针。图5画出了发现流程的框架图。

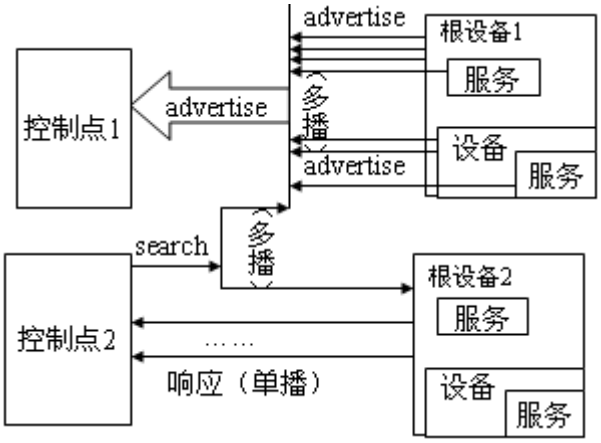

图5 发现过程框图

3.3 描述（Description）

UPnP 网络结构的第二步是设备描述。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，控制点可能仅仅知道设备或服务的 UPnP 类型，设备的 UUID 和设备描述的 URL 地址。为了让控制点更多的了解设备和它的功能或者与设备交互，控制点必须从发现消息中得到设备描述的 URL，通过 URL 取回设备描述。设备描述的一般过程如图 6 所示：

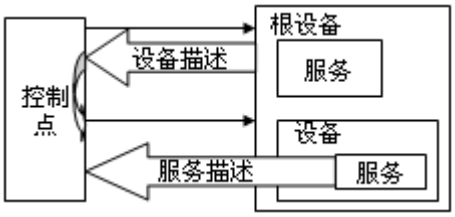

图6　获得设备描述及服务描述

对于一个设备的UPnP描述一般分成两个部分：设备描述和设备的服务描述。

● 设备描述

UPnP对某一设备的描述以XML形式表示出来，设备描述包括制造商信息，包括模块名称和编号，序列号，制造商名称，制造商网站的URL等等。设备描述也包括所有嵌入设备描述和URL地址集。对于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是一个根设备其中嵌入多个设备，也可以是多个根设备的方式实现。设备描述是由设备制造商提供的，采用XML表述，并且遵循UPnP设备模版。此模版是由UPnP工作委员会生成的。

● 服务描述

包括一系列命令或者动作，服务响应，动作的参数。服务的描述也包含一系列变量，这些变量描述了服务运行时刻的状态，这包括数据类型、取值范围和事

件特性的描述。服务描述也是由设备制造商提供的，采用XML方式表述，遵循UPnP服务模版。

3.4 控制（Control）

在接收设备和服务描述之后，控制点可以向这些服务发出动作，同时控制点也可以轮询服务的状态变量值。发出动作实质上是一种远程过程调用；控制点将动作发送到设备服务，在动作完成（或失败）后，服务返回相应的结果或错误。状态变量值轮询是这种场景下的特例，动作及其结果都是预定义的。 其基本过程如下图所示：

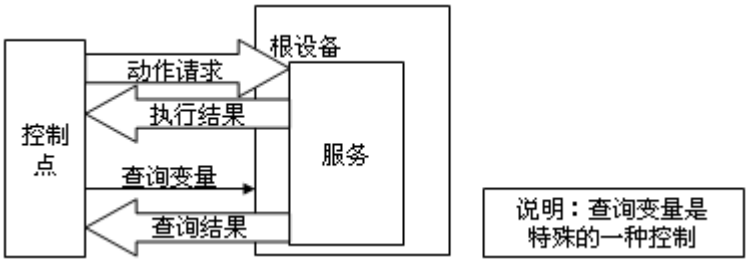

图 7　控制点发出动作和轮询变量

为了控制一个设备，控制点向设备服务发出一个动作。这一般由控制点向服务的控制URL地址（在设备描述的服务元素controlURL子元素部分提供）发送一个适当的控制消息。而服务则会对此动作做出响应，返回相关结果或错误。动作的效果可以通过改变描述服务运行时状态的变量进行建模。在这些状态变量改变时，事件将被发布到所有相关的控制点。

控制点可能会轮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。而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，以便控制点能够轮询并接收到有意义的值。

3.5 事件（Eventing）

正如描述部分所述，一个即插即用服务描述包括服务响应的动作列表和运行时描述服务状态的变量列表。如果一个或多个状态变量可以被事件触发，服务将会在这些变量发生变化时发布更新，控制点可以订阅以获得此信息。通过这一部分，发布者指事件的来源（通常为设备服务），订阅者指事件目的地（通常为控制点）。

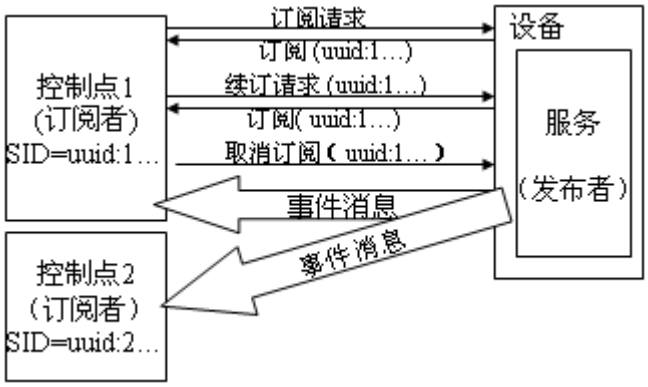

图8 事件过程框图

如图8所示，要订阅事件，订阅者可发送一条请求订阅消息。如果发布者收到此消息，它将以这个订阅的持续时间作为响应。要保持订阅，订阅者必须在订阅过期之前进行续订。当订阅者不再需要发布者发送的事件时，订阅者应当取消其订阅。

发布者通过发送事件消息提醒订阅者状态变量改变。事件消息包含多个状态变量名称和这些变量的当前值，以XML表示。在订阅者第一次订阅时，需要发送一个专门的初始化事件消息。该事件消息包含所有事件变量的名称和值，并且允许订阅者初始化其服务状态模型。为了支持多个控制点，在动作生效后所有订阅者均会收到通知。由此，将向所有订阅者发送全部事件消息，订阅者会收到所有事件触发变量的事件消息（不只是一部分）。不管状态变量因何种原因发生改变（无论为了响应所要求的动作，还是由于服务模拟的状态发生变化）均会发送事件消息。 事件消息使用HTTP协议传送，事件详细定义在通用事件通知结构（General Event Notification Architecture）协议中。

3.6 展示（Presentation）

在控制点发现设备和取得设备描述之后，控制点即准备开始提供展示。如果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/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。

设备表征包含在设备描述的 presentationURL 字段。设备表征可以完全由设备制造商提供，它采用 HTML 页的形式，使用 HTTP 进行发布。图 9 画出了展示流程的示意图。

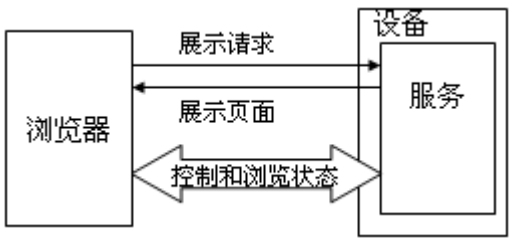

图9 展示（Presentation）示意图

4 基于Linux的UPnP协议实现的源代码模块

为了促进UPnP协议的发展，信息业巨头Intel公司开放了一个基于Linux的UPnP协议栈源代码，该协议栈实现了UPnP规范中SSDP协议模块，GENA协议模块，SOAP协议模块。本文后面列举的TV控制点和设备的源代码实现就是基于这个开源协议栈。本部分将对该协议栈模块的体系结构做简单分析，其协议模块的体系结构如图10所示。

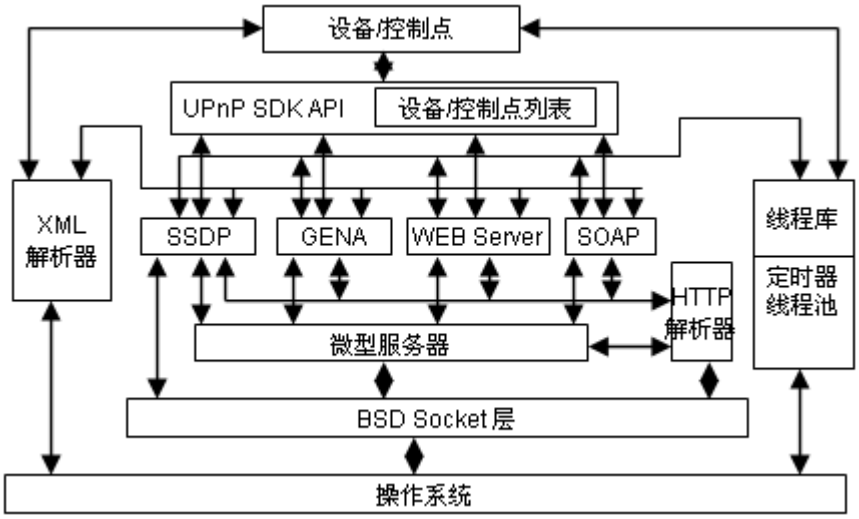

图 10 UPnP 协议实现的各个模块[1]

4.1 设备/控制点

客户端或服务器的申请软件均运行在软件开发包(SDK)之上。客户端或服务器的申请相当于执行了一个完成特殊服务的函数。例如，拿一个网关服务来说，服务器软件执行这个“Enable Internet”函数时，就相当于控制点软件开始能控制使用UPnP设备。

4.2 UPnP软件开发包API(UPnP SDK API)

UPnP软件开发包API部分将UPnP协议核的详细内容从控制点和服务器申请中抽象如来，并且给了一个统一界面的入口函数，里面包含了SSDP，GENA和SOAP等协议。

这个API层也包含了用软件开发包(SDK)注册的控制点和设备操作列表。每个对API函数的请求，软件开发包(SDK)都将会对照操作列表中的相应项，对操作列表中所包含的相应操作，在这里均会使之生效。同时，在这个过程中，只限对一个控制点或一个设备的操作进行定位。换句话说，一个申请一次要么注册一个设备，要么注册一个控制点，而一个申请不能同时注册一个设备和控制点。任何想同时注册设备和控制点的申请都会失败。

4.3 WEB Server

微型服务器模块处理一个标准的HTTP GET请求。在UPnP组成元素中，都要求使用基本的HTTP服务，所以需要使用微型服务器模块来完成基本的HTTP服务。微型服务器模块管理文档的定位，这些文档一般对GET命令有效，并且运行使用HTTP协议的数据流进行传输。

微型服务器也支持虚拟路径HTTP POST请求，但不支持其他种类的请求。

4.4 库模块

4.4.1 XML解析模块

XML在UPnP中被广泛地使用。描述文档就是XML文档。GENA使用XML来描述一个服务状态的描述变化。SOAP使用XML来格式化请求和响应。软件开发包(SDK)包含一个XML解析，这个XML解析会被UPnP协议核和客户端或服务端软件使用。

4.4.2 SDK中的线程库

UPnP协议栈的实现中大量采用了多线程技术，该模块就是为了更方便高效地使用线程技术，其中包含了一个定时器线程子模块和一个线程池子模块。线程池子模块用于管理协议栈中所有的线程；定时器线程子模块用来处理协议栈中所有的定时事件。

应用层在调用API UpnpInit()时会初始化两个缓冲池(由ThreadPoollnit()创建)：发送缓冲池，接收缓冲池。在创建每个缓冲池时都要给定一个缓冲池的属性作为参数，以确定在该缓冲池中最少，最多可以容纳的线程数，然后调用CreateWorker()创建所需的线程。

当有新的任务要执行的时候，调用TPJobInit()，ThreadPooIAddPersistent()或者ThreadPoolAdd()将任务加到缓冲池的相应队列中，等待空闲的线程调用执行。

4.4.3 HTTP解析器

设备公告消息、设备查询消息、设备查询响应消息都是用HTTP协议来封装的，使用了SSDP协议定义的头和方法，这需要开发者自己来解释这些消息。同样，事件订阅消息和事件取消订阅消息也是用HTTP协议来封装的，使用了GENA定义的头和方法，也需要开发者自己解析这些消息．这里我们把这两部分的HTTP消息解析合并成一个模块：HTTP消息的解析。通常的HTTP消息是由浏览器或www服务器来解析的，这里的HTTP消息解析只对SSDP、GENA定义的特殊的HTTP消息进行解析。

4.4.4 微型服务器（Mini Server）

Mini Server层提供GENA、SOAP、SSDP、mini web server需要的公共功能，

这一层接收所有的网络连接，并决定哪个请求可以进入上层，将HTTP头部交给HTTP模块去解析，并将这个连接转向合适的协议去处理。

5 TV控制点及设备的代码实现

本部分采用一个实例对 UPnP 的代码实现进行了阐述，该例子中包括一个 TV 控制点和一个 TV 设备，且只实现控制点对电视设备的搜索，无法搜索其他的 UPnP 设备。该例实现了对 TV 设备的服务控制及服务订阅等流程。工作平台为 Windows XP，编译环境为 Visual Studio2008，使用两台 PC，一台 PC 上运行 TV 控制点，一台 PC 运行 TV 设备，且两台 PC 处于同一局域网中，操作结果在各自的执行界面上显示。

5.1 TV控制点的代码实现

5.1.1 发现、描述的代码实现

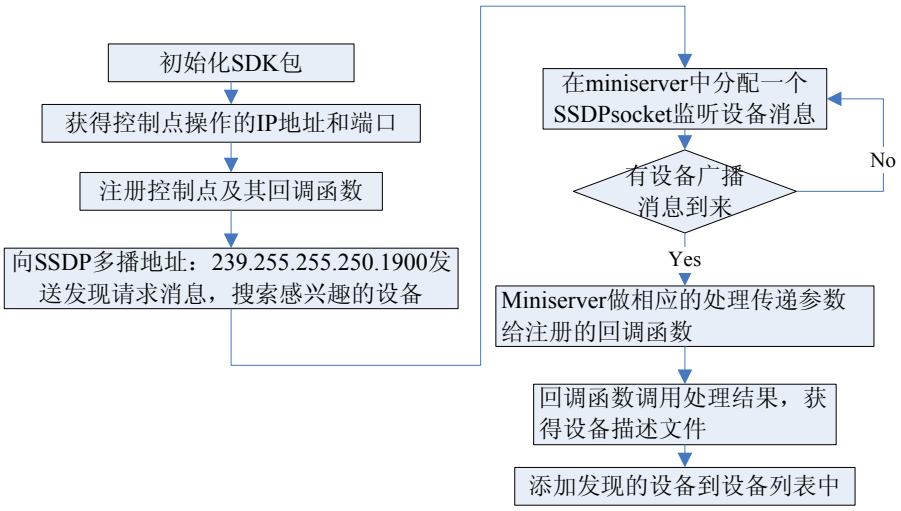

图 11　发现设备及获得设备描述的实现流程图

发现、描述流程的实现流程图如图 11 所示，需要说明的是，本例的发现请求消息没有收到 TV 设备的直接响应，因为考虑到本例只实现对一个 TV 设备的搜索，所以只需要解析 TV 设备发送的广播消息就足以获得设备的描述文件等。

5.1.1.1 启动及初始化

● UpnpInit()初始化 SDK

```
unsigned short port = 0;
 char *ip_address = NULL;

rc = UpnpInit( ip_address, port );

if( UPNP_E_SUCCESS != rc ) {
      SampleUtil_Print( "WinCEStart: UpnpInit() Error: %d", rc );
}
```

应用程序可以在初始化过程中指定IP地址和端口号。对于控制点来说，设置

IP地址和端口号用于监听事件消息。如果地址为空，那么第一个非空、非回路地址将被使用。你可以在初始化完成后通过UpnpGetServerlpAddress0和UpnpGetServerPort()来取得IP地址和端口号。TV控制点就是这样来实现的。

在UpnpInit()函数里，创建了发送线程池和接收线程池，并启动miniserver。

● UpnpRegisterClient()注册控制点

```
rc = UpnpRegisterClient( TvCtrlPointCallbackEventHandler,
                         &ctrlpt_handle, &ctrlpt_handle );

if( UPNP_E_SUCCESS != rc ) {
    SampleUtil_Print("Error registering CP: %d", rc );
    UpnpFinish();
    return TV_ERROR;
  }
```

对于函数UpnpRegisterClient(),第一个参数是控制点希望SDK使用的回调函数。TV例子中是TvCtrlPointCallbackEventHandler()。第二个参数是指向一小段信息的指针，这段信息在回调函数被触发时会传递给回调函数。TV例子中传递的是控制点句柄。最后一个参数是指向控制点句柄的指针，这个句柄在生成后续SDK调用时会用到。

TV例子使用TvCtrlPointCallbackEventHandler0处理所有异步操作,该函数处理所有的事件消息，包括SSDP，GENA，SOAP等。后续讨论在必要时会述及该函数的某些片断。

在控制点调用UpnpRegisterClient()注册回调函数后，所有网络上设备的设备宣告消息将会由回调函数传递给应用程序。应用程序应该准备好处理这些消息。

5.1.1.2 发现设备

当控制点应用程序初始化完成后，就可以开始在网络上搜索感兴趣的设备。TV例子是一个非常简单的控制点，它只搜索一种设备：Tv例子设备。

● UpnpSearchAsync（）函数发现设备

```
rc = UpnpSearchAsync( ctrlpt_handle, 5, TvDeviceType, NULL );
if( UPNP_E_SUCCESS != rc ) {
    SampleUtil_Print("Error sending search request%d", rc );
    return TV_ERROR;
  }
```

例子在函数TvCtrlPointRefresh()开始搜索TV设备。UpnpSearchAsync()开始搜索设备过程。它有以下参数：

- 控制点句柄
- 控制点等待响应的时间(以秒为单位，程序里为5)
- 搜索目标

·一段可选的cookie

对于TV设备而言，搜索目标定义为：

char TvDeviceType[]=”urn：schemas-upnp—org：device：tvdevice：1”；

● readFromSSDPSocket（）接收SSDP广播消息并进行处理

在该函数里会启用一个事件处理线程ssdp_event_handler_thread，在该线程里会对来自设备的广播消息会由函数ssdp_handle_ctrlpt_msg()进行处理，然后将相应参数送到TvCtrlPointCallbackEventHandler（）函数进行处理，从而获得设备描述文件。

● TvCtrlPointCallbackEventHandler（）函数处理所有事件信息

函数原型如下：

int CallbackFxn(Upnp_EventType EventType,void *Event, void *Cookie)；

其中第一个和第二个参数均由UPnP模块在发生回调时传入，EventType表征了待处理的UPnP请求的类型，Event是指向一个结构体的指针，该结构体中包含了关于请求消息的具体信息，不同类型的请求会对应不同的结构。参数Cookie是设备应用程序中定义的数据。

● 发现事件消息结构体

当控制点收到来自设备的响应消息后，会经由miniserver相应处理，传给回调函数一个发现事件消息，其结构体如下：

```
struct Upnp_Discovery
{
        int  ErrCode; //发现请求消息所返回的结果代码
        int  Expires; //多播消息的持续时间
        char DeviceId[LINE_SIZE]; //唯一的设备标识符
        char DeviceType[LINE_SIZE]; //设备类型
        char ServiceType[LINE_SIZE];//服务类型
        char ServiceVer[LINE_SIZE];  //服务版本
        char Location[LINE_SIZE];    //设备描述文件的URL位址
        char Os[LINE_SIZE];  //设备所运行的操作系统
        char Date[LINE_SIZE];   //响应的时间
        char Ext[LINE_SIZE];
        struct sockaddr_in DestAddr;//响应设备的IP
};
```

该结构体里面返回了很多关于设备的重要信息，包括设备类型（tvdevice）、服务类型（tvcontrol）、以及设备描述文件的URL位址等等。

5.1.1.3 获得设备描述

● UpnpDownloadUrlItem()获得设备描述XML数据

在该函数里，通过设备URL创建设备描述请求消息，再经过发送并等待响应消息，在响应消息里获得设备描述XML数据存在一个char* buffer里。

● ixmlParseBufferEx()函数解析XML文档（首行缩进2字符）

在上述函数获得XML数据后，调用该函数对其进行解析，解析的作用是将XML文件数据解析成DOMtree结构，从而可对设备描述文件进行访问。

5.1.2 订阅服务的代码实现

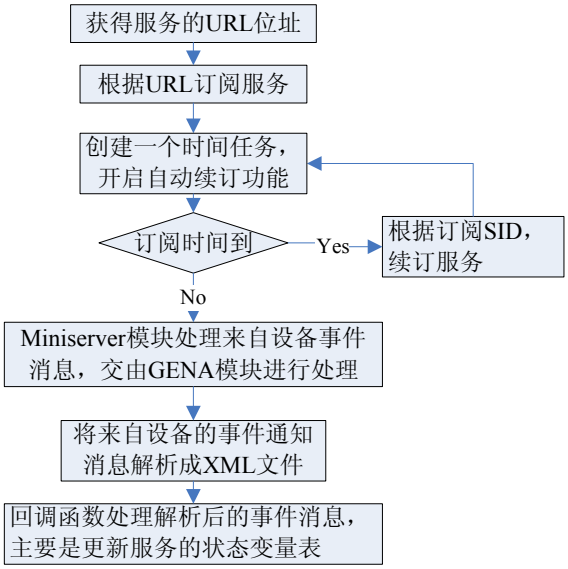

图12 订阅的代码实现流程图

5.1.2.1 订阅

● SampleUtil_FindAndParseService（）获取服务的URL

```
int SampleUtil_FindAndParseService( IN IXML_Document * DescDoc,
                                    IN char *location,
                                    IN char *serviceType,
                                    OUT char **serviceId,
                                    OUT char **eventURL,
                                    OUT char **controlURL )
```

第一个参数为以XML文件存在的设备描述文件，第二个是设备描述文件的URL，第三个是服务类型，后面三个分别是函数执行后得到的该服务的SID，事件URL和控制URL。

● gena_subscribe()完成订阅任务

通过上述函数得到事件URL后，就可以开始订阅任务了。订阅函数为：

```
gena_subscribe( IN char *url,          INOUT int *timeout,
                IN char *renewal_sid, OUT char **sid )
```

url：订阅的服务URL；

timeout：订阅的截至时间；

renewal_sid：用于续订，对于第一次订阅，此指针为空；

sid：函数返回一个用于订阅该服务的订阅标志符SID，续订会用到；

从上述参数可知，订阅一个服务，主要是通过服务的URL位址来实现。对于设备，会返回一个用于此服务订阅的SID，用于发布事件消息和续订等。

● 回调函数处理的事件消息结构体

在控制点订阅了某服务后，设备就会立即返回一个事件通知消息，反映该服务的状态变量及其当前值，其返回的事件消息经GENA模块处理后，会得到一个由回调函数处理的事件结构体，如下：

```
struct Upnp_Event
{
 Upnp_SID Sid;   //该服务订阅的SID
 int EventKey;    //事件序列号
 IXML_Document *ChangedVariables; //含有事件消息的xml文件
};
```

5.1.2.2 续订

● genaRenewSubscription()完成续订任务

实现任务的续订，是通过函数ScheduleGenaAutoRenew()在gTimerThread线程池里创建一个自动续订任务，最后通过函数genaRenewSubscription()来实现的。

```
int genaRenewSubscription( IN UpnpClient_Handle client_handle,
                           IN const Upnp_SID in_sid,
                           INOUT int *TimeOut )
```

client_handle: 控制点句柄。

in_sid: 订阅标志符SID（在第一次订阅时获得）。

TimeOut:续订的持续时间

由上述参数可知，完成续订任务并不是通过服务的URL位址，而是通过在第一次订阅时取得的SID来实现的。

5.1.3 控制服务的代码实现

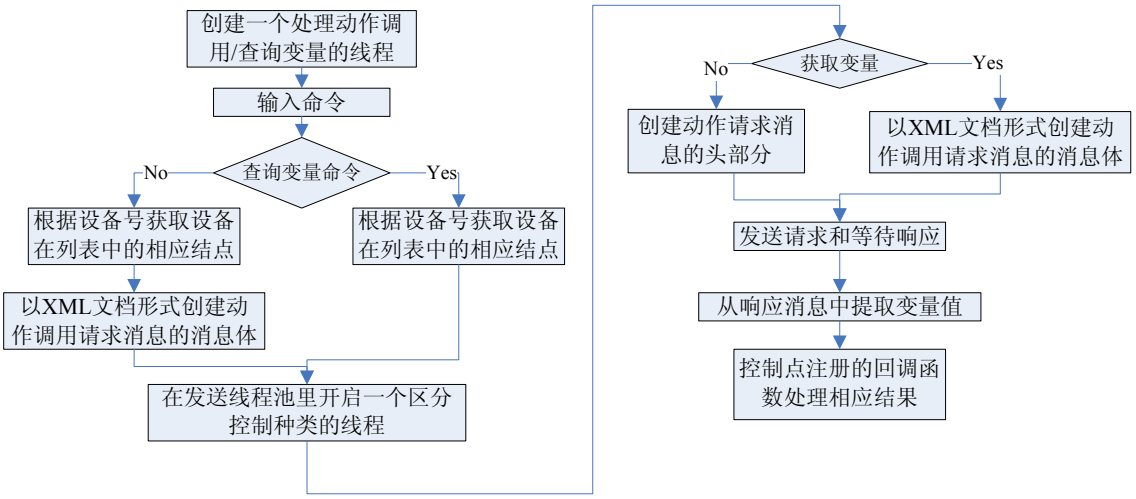

图13 控制代码实现流程图

5.1.3.1 动作调用的代码实现

● UpnpMakeAction()创建动作请求消息

在向设备发送动作消息的函数要用到描述动作的DOM文件，SDK中有组织这些DOM文件的功能函数：UpnpMakeAction()和UpnpAddToAction0。TV例子中在TvCtrlPointSendAction0充分利用了这些函数：

```
IXML_Document *actionNode = NULL;

if( 0 == param_count ) {
        actionNode =
          UpnpMakeAction( actionname, TvServiceType[service], 0,
                          NULL );
    } else {
      for( param = 0; param < param_count; param++ ) {
        if( UpnpAddToAction
            ( &actionNode, actionname, TvServiceType[service],
              param_name[param],
              param_val[param] ) != UPNP_E_SUCCESS ) {
            SampleUtil_Print
              ( "ERROR: TvCtrlPointSendAction: Trying to add action param" );
            }
      }
    }
```

如果动作不需要参数，UpnpMakeAction()就已经足够构造正确的动作节点了。否则的话，将会反复调用UpnpAddToAction()往动作节点文件中添加参数。

● UpnpSendActionAsync()发送动作消息

控制点通过改变设备的内部状态来使设备完成某些功能。这是通过向设备发送动作消息实现的。SDK有两个函数用于改变设备的内部状态：UpnpSendAction()和UpnpSendActionAsync()。两个函数都完成相同的功能，只是后面一个是异步操作。

```
rc = UpnpSendActionAsync( ctrlpt_handle,
                          devnode->device.TvService[service].
                          ControlURL, TvServiceType[service],
                          NULL, actionNode,
                          TvCtrlPointCallbackEventHandler, NULL );
```

第一个参数为控制点句柄，第二个为被控制的服务类型，第三个是服务的控制URL，第四个参数是设备的UDN，此处为空；第五个是以XML文档形式存在的动作请求消息；第六个是动作完成后的回调函数，最后一个是存放用户数据的可选项，此处为空。

当异步的动作消息发送完成后，响应消息通过回调函数返回，回调函数句柄是由UpnpSendActionAsync()指定或由UpnpRegisterClient()注册，例子中倾向于对所有的消息用同一个句柄，所有动作完成和服务状态变量查询消息均在TvCtrlPointCallbackEventHandler()结束。

5.1.3.2 查询变量的代码实现

● UpnpGetServiceVarStatusAsync()查询状态变量

另外，要查询服务的某个状态便量的值也可以通过两个函数来实现：UpnpGetServiceVarStatusAsync()和UpnpGetServiceVarStatus()。两个函数都完成相同的功能，只是前一个是异步操作。

```
if( TV_SUCCESS == rc ) {
    rc = UpnpGetServiceVarStatusAsync( ctrlpt_handle,
                          devnode->device.
                          TvService[service].ControlURL,
                          varname,
                          TvCtrlPointCallbackEventHandler,
                          NULL );
    if( rc != UPNP_E_SUCCESS ) {
      SampleUtil_Print
        ( "Error in UpnpGetServiceVarStatusAsync -- %d", rc );
      rc = TV_ERROR;
    }
  }
```

对于函数UpnpGetServiceVarStatusAsync()，和函数UpnpSendActionAsync()的参数基本上一致，最大的不同是请求查询变量不是以XML文档形式存在，而是以char型，具体的封装成XML文档形式是在函数内部解决的。

当异步的查询请求消息发送完成后，和前面的动作请求一样，响应消息通过回调函数返回。

5.1.4 退出

当控制点应用程序退出时，需要用UpnpUnRegisterClient0 SDK取消注册并用UpnpDeInit0关闭SDK。

```
int TvCtrlPointStop( void )
{
    TvCtrlPointRemoveAll();
    UpnpUnRegisterClient( ctrlpt_handle );
    UpnpFinish();
    SampleUtil_Finish();

    return TV_SUCCESS;
}
```

5.2 TV设备的代码实现

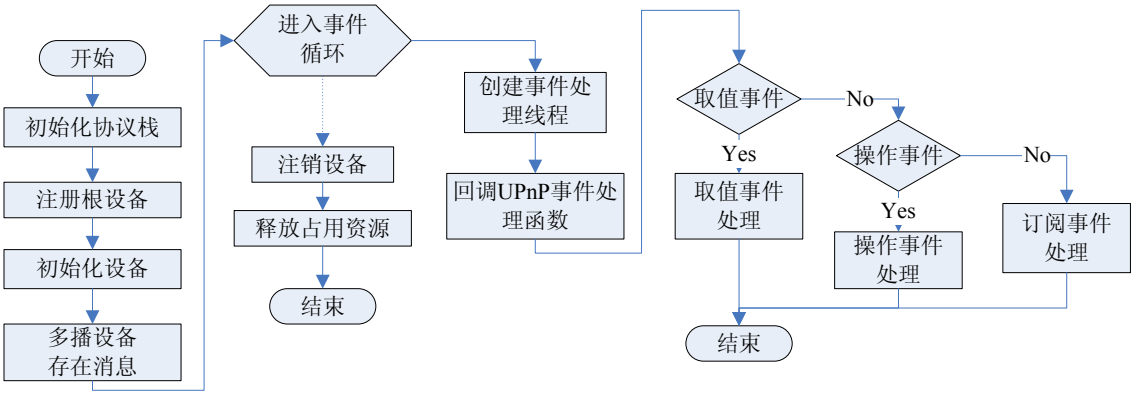

图14 TV设备实现流程图

5.2.1 设置和初始化设备

● 初始化UPnP协议栈

与控制点一样，初始化库方法是调用库函数Upnplnit()。

● 设置根目录

初始化UPnP协议栈后，有必要为UPnP模块中的web Server指定根目录，这样当网络中的控制点向服务器发出请求各种设备或服务描述文件资源的话，web server将在该目录下搜索。这一步通过UpnpSetWebServerRootDir()完成。

● 注册根设备

通过这一步，TV设备与UPnP库关联在一起。当miniserver服务器监听到有对应与TV设备的请求时，将回调该设备提供回调事件处理函数进行处理。其代码片段如下：

```
if( ( ret = UpnpRegisterRootDevice( desc_doc_url,
                                    TvDeviceCallbackEventHandler,
                                    &device_handle, &device_handle ) )
    != UPNP_E_SUCCESS ) {
    SampleUtil_Print("Error registering the rootdevice : %d\n", ret );
    UpnpFinish();
    return ret;
}
```

UpnpRegisterRootDevice()的第一个参数为设备描述文件的URL；第二个参数为设备的回调事件处理函数，为TvDeviceCallbackEventHandler()。该函数是UPnP设备的核心。最后一个参数用于接收UPnP模块为设备分配的句柄，通过该句柄可以操作设备。

● 特定设备的初始化

在向网络声明设备之前，还需要做一些特定于设备的初始化工作，比如：对嵌套于设备中的服务的状态变量的初始化。

● 声明设备

完成上述步骤后，向网络上的控制点声明设备的存在。代码如下；

```
if( ( ret =
        UpnpSendAdvertisement( device_handle, default_advr_expire ) )
      != UPNP_E_SUCCESS ) {
      SampleUtil_Print("Error sending advertisements : %d\n", ret );
      UpnpFinish();
      return ret;
    }
```

至此，设备开始等待请求消息的到来，进入事件处理循环。

5.2.2 处理异步请求

设备的生命期中，最主要的任务就是处理由控制点发送过来的各种异步请求。UPnP设备在生命期内需要处理的异步请求可分三类：订阅请求，取值请求和操作请求。

5.2.2.1 订阅请求处理

该请求用于订阅UPnP设备中嵌套服务的状态改变事件通知。当控制点向UPnP设备发送订阅请求时，处于监听状态miniserver服务器激发设备应用中定义的回调函数，该回调函数将在一个独立的线程中运行。而传入回调函数的EventType参数值为UPnP模块中定义的类型常量UFNP_EVENT_SUBSCRIFTION_REQUEST；参数Event指向的包含具体请求数据的结构体类型为Upnp_Subscription_Request，其定义如下：

```
struct Upnp_Subscription_Request
(char *ServiceId; //请求订阅的服务标识符
Char *UDN; //通用设备名
Upnp_SID Sid; //本次订阅的序列号
);
```

在请求的后续处理中将调用UpnpAcceptSubscription()或者UpnpAcceptSubscriptionExt()来响应控制点，这两个函数的区别在于两者以不同的方式传送服务状态表给UPnP模块，前者使用对应的“变量”“变量值”数组方式，后者使用DOM文档，订阅请求成功之后，每当控制点订阅的UPnP设备的服务状

态发生变化，它都会被及时通知。

5.2.2.2 取值请求处理

该请求用于返回指定的UPnP设备嵌套服务中的服务状态变量当前值。UPnP设备监听到控制点的取值请求后，激活设备中预定义的事件处理回调函数，传入的参数EventType值为UPNP_CONTROL GET_VAR_REQUEST，而参数Event则指向一个Upnp_State_Var_Request类型的结构体，其定义如下：

```
struct Upnp_State_Var_Request
{
 int ErrCode;                          //操作结果代码
 int Socket;                           //用于接收请求的socket
 char ErrStr[LINE_SIZE];               //错误代码对应的说明字符串
 char DevUDN[NAME_SIZE];               //单一设备标识符
 char ServiceID[NAME_SIZE];            //服务标识符
 char StateVarName[NAME_SIZE];         //变量名
 struct in_addr CtrlPtIPAddr;          //存有客户请求端IP
 DOMString CurrentVal;                 //变量当前值
};
```

设备对该请求的响应就是在Event指向的结构体的变量当前值域填入所求变量的当前值，然后返回，让miniserver模块提供的接口发送响应给控制点。

5.2.2.3 操作请求处理

操作请求用于改变指定的UPnP设备嵌套服务中的服务状态变量。当UPnP设备收到操作请求时，设备应用中定义的回调函数将被调用，传入的参数EventType值为UPNP_CONTROL_ACTION_REQUEST，而参数Event指针则指向一个Upnp_Action_Request类型的结构体，下面是此结构体的类型声明：

```
struct Upnp_Action_Request
{
int ErrCode；                  //操作结果代码
int Socket；                   //用于接收请求的socket
char ErrStr[LINE_SIZE]；   //错误代码对应的说明字符串
char ActionName[NAME_SIZE]；//所请求的操作名
char DevUDN[NAME_SIZE]；   //设备ID
char ServiceID [NAME_SIZE]； //包含所请求操作的服务ID
Upnp_Document ActiortRequest；//描述操作细节的DOM文档
Upnp_Document ActionResult； //描述操作结果的DOM文档
}：
```

TV设备将从操作请求文档中提取出完成操作所需的相关参数，执行请求操作，并创建一个响应文档(该文档内含有可能的操作出口参数)，最后发送事件，通知控制点设备的服务状态已经改变。

5.2.3 发送事件通知

当设备中某一服务的状态变量改变时，设备需要发送对应的更新事件给网络中已订阅这些状态变量的控制点，可通过调用UpnpNotify()完成事件通知。

5.2.4 关闭设备

当设备被关闭时，先要向网络中的各控制点发送SSDP“bye-bye”消息，声明设备将不再可用。同时释放所占用的资源，下面两条语句完成这一任务。

UpnpUnRegisterRootDevice(device_handle) ;

UpnpFinish0;

函数UpnpUnRegisterRootDevice()的参数为设备句柄。

6 UPnP协议消息分析

6.1 发现设备的消息分析

UPnP协议的发现过程是通过SSDP协议来实现的，SSDP消息至少包含以下四部分：

• 服务类型URI

用来标识一种服务，如宣告消息中的NT字段，搜索消息中的ST字段。

• 唯一的服务名字(USN)URI

USN用来区分同一服务类型的不同的具体服务。

• 有效期信息

指明SSDP客户在其缓存中保存该服务信息的时间，如宣告消息中的

CACHE. CONTROL字段。

• 位置信息

服务所在位置，如宣告消息中的LOCATION字段。

6.1.1 SSDP发现请求

SSDP发现请求用来寻找希望得到的设备及其服务。它使用SEARCH消息，用URI ssdp：discover标识。SEARCH消息必须包括一个ST(搜索类型)头，还可能含有消息体。ST头用来指明希望发现的设备及服务类型。目前只指定了在多播UDP中使用该请求，以后可能扩展到TCP。

只有那些与ST匹配的服务才可能通过SSDP端口给出响应，响应应该在AL头中表示出服务的地址，同时，响应中还要包括缓存控制信息：有效期。没有缓存控制信息的响应将被视作无效响应。由4.1.1知，本例没有设备的响应消息。

以下是TV控制点发出的发现设备请求消息：

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: 5
ST: urn:schemas-upnp-org:device:tvdevice:1
```

图15 TV控制点发送的请求发现设备消息

从ST头可以知道，TV控制点搜索的设备是tvdevice,设备号为1。其中HOST由

互联网编号分配组织（IANA）为SSDP保留的多播信道和端口。必须是239.255.255.250:1900。MAN是强制扩展声明标头，一个含有强制扩展声明的HTTP请求被称作强制请求，在这个消息中使用强制扩展的用意在于确定接受方是否理解和支持ssdp：discover，接受方如果支持ssdp：discover，将会对扩展声明进行解析，否则返回出错信息。

6.1.2 SSDP存在宣告

SSDP存在宣告可以让SSDP客户知道设备及服务的存在、更新缓存中的过期信息、服务的位置改变或即将无效。SSDP存在宣告使用NOTIFY消息，NOTIFY消息包括位置和 / 或AL头部(指示设备及服务的地址)、NT头(指示设备及服务类型)、USN(唯一服务名称，可以和其它同类型服务相区别)以及Cache-Control或者Expiration头部(缓存控制信息)。对方收到后根据NOTIFY的NT判定是否是自己感兴趣的设备及服务。如果是，若缓存中未见此USN，则增加记录，否则更新记录。对此消息不需要给出响应。

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age=100
LOCATION: http://172.28.17.235:49152/tvdevicedesc.xml
NT: upnp:rootdevice
NTS: ssdp:alive
SERVER: Linux/2.6.9-78.ELsmp, UPnP/1.0, Portable SDK for UPnP devices/1.6.6
X-User-Agent: redsonic
USN: uuid:Upnp-TVEmulator-1_0-1234567890001::upnp:rootdevice
```

图16 TV设备发送的宣告消息

从NT头知道，TV设备是根设备，由NTS子类型知道，该设备目前可用，且由USN指出了TV设备的唯一设备名称UUID。

6.2 描述数据包分析

一般情况而言，在描述部分，控制点会得到两类描述，设备描述和服务描述，本例只获得了设备描述。因为TV设备只定义了一个服务，其变量为power、channel、volume。这三个变量直接是通过在代码里定义，然后进行操作的。

6.2.1 获取设备描述的请求消息

```
GET /tvdevicedesc.xml HTTP/1.1
HOST: 172.28.17.235:49153
DATE: Thu, 13 May 2010 02:28:20 GMT
CONNECTION: close
USER-AGENT: 5.1.2600 2/Service Pack 2, UPnP/1.0, Portable SDK for UPnP devices/1.6.6
```

图17 控制点发送的GET请求消息

控制点是通过发送 HTTP GET 消息来请求获得设备描述的。GET 指出了设备描述的路径，此处为设备描述 URL（发现消息中的 LOCATION 标头）；HOST 指出了设备描述 URL 的域名或 IP 地址及可选端口部分。

6.2.2 TV设备的设备描述消息

```
HTTP/1.1 200 OK
CONTENT-LENGTH: 783
CONTENT-TYPE: text/xml
DATE: Thu, 20 May 2010 10:02:50 GMT
LAST-MODIFIED: Thu, 20 May 2010 09:54:34 GMT
SERVER: Linux/2.6.9-78.ELsmp, UPnP/1.0, Portable SDK for UPnP devices/1.6.6
X-User-Agent: redsonic
CONNECTION: close

<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <device>
    <deviceType>urn:schemas-upnp-org:device:tvdevice:1</deviceType>
    <friendlyName>UPnP Television Emulator</friendlyName>
    <UDN>uuid:Upnp-TVEmulator-1_0-1234567890001</UDN>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:tvcontrol:1</serviceType>
        <serviceId>urn:upnp-org:serviceId:tvcontrol1</serviceId>
        <controlURL>/upnp/control/tvcontrol1</controlURL>
        <eventSubURL>/upnp/event/tvcontrol1</eventSubURL>
        <SCPDURL>/tvcontrolSCPD.xml</SCPDURL>
      </service>
    </serviceList>
   <presentationURL>/tvdevicepres.html</presentationURL>
</device>
</root>
```

图18 设备返回的设备描述消息

空白处前面的数据为HTTP消息头，描述了XML描述文件的一些基本信息，空白处下面的数据就为XML文件的具体内容，具体的参数内容详见图3。

6.3 事件数据包分析

6.3.1 订阅请求的消息分析

对于设备中的每项服务，描述消息均包含一个事件触发URL（设备描述中服务元素的eventSubURL子元素）和UPnP服务标识符（设备描述中服务元素的serviceId子元素）。为订阅特定服务的事件，订阅消息将被发送到该服务的事件触发URL。（注意，事件触发URL可能相对于基本URL。）消息中含该服务的标识符以及事件消息交付URL。订阅消息还可能包含所要求的订阅持续时间。

```
SUBSCRIBE /upnp/event/tvcontrol1 HTTP/1.1
HOST: 172.28.17.235:49155
CALLBACK: <http://172.28.17.91:49152/>
NT: upnp:event
TIMEOUT: Second-100
```

图19 订阅消息

由上图可知，本订阅消息订阅的是TV的control服务，CALLBACK指出了回送的路径，即TV控制点的IP和端口，NT指出了TV控制点想接受的事件消息，HOST指出了事件触发URL（设备描述中服务元素的eventSubURL子元素）的域名或IP地址和可选端口组件。TIMEOUT指出了直到订阅期满所要求的持续时间。

```
HTTP/1.1 200 OK
DATE: Thu, 13 May 2010 09:07:47 GMT
SERVER: Linux/2.6.9-78.ELsmp, UPnP/1.0, Portable SDK for UPnP devices/1.6.6
CONTENT-LENGTH: 0
X-User-Agent: redsonic
SID: uuid:ffe2cc98-5e6e-11df-bc2d-d36115e5e0e0
TIMEOUT: Second-100
```

图20 订阅的响应消息

由上图知，响应消息返回了一个用于本次订阅的订阅号SID，此SID在订阅期间必须是唯一的。DATE指出了响应生成的具体时间。

6.3.2 续订请求的消息分析

为续订特定服务的事件，续订消息将被发往该服务的事件触发URL。然而，与最初的订阅消息不同，续订消息既不包含服务标识符也不包含事件消息交付URL，而是包含由发布者（设备）分配的订阅标识符，为续订提供明确的参考。与订阅消息相同，续订消息还可能包含所要求的订阅持续时间。

```
SUBSCRIBE /upnp/event/tvcontrol1 HTTP/1.1
HOST: 172.28.17.235:49155
SID: uuid:ffe2cc98-5e6e-11df-bc2d-d36115e5e0e0
TIMEOUT: Second-100
```

图21 续订请求数据包

由上图知，TV控制点直接通过SID完成对tvcontrol的续订任务。

6.3.3 NOTIFY事件通知消息分析

当设备的服务状态变量有变更时，设备会发布其状态变量变更的事件消息。对于控制点而言，在完成订阅或者续订了某项服务之后，它就会收到关于此服务的状态变量变更的事件消息。

以下是设备发送的关于tvcontrol的事件通知消息。

```
NOTIFY / HTTP/1.1
HOST: 172.28.17.91:49152
CONTENT-TYPE: text/xml
CONTENT-LENGTH: 214
NT: upnp:event
NTS: upnp:propchange
SID: uuid:ffe2cc98-5e6e-11df-bc2d-d36115e5e0e0
SEQ: 0

<e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-0">
<e:property>
<Power>1</Power>
</e:property>
<e:property>
<Channel>1</Channel>
</e:property>
<e:property>
<Volume>5</Volume>
</e:property>
</e:propertyset>
```

图22 设备返回的tvcontrol服务的事件通知信息

在空白处上方是事件消息的命令行和标头。标头里的SEQ是事件编号，对于事

件消息均标有事件编号，为便于进行错误检查，发布者必须保持每个订阅有一个单独的事件编号。当发布者发送初始化事件消息时，订阅事件编号被初始化为0（图22所示的便是这种情况）。对于以后的每条事件消息，发布者会增加订阅事件编号，包括事件消息更新的编号。

空白处下方是该事件消息的消息体部分，以XML文档形式存在。在消息体内，指出了关于tvcontrol服务的所有状态变量（Power、Channel、Volume）及其当前值。

6.4 控制数据包分析

6.4.1 动作调用的消息分析

● SOAP动作调用

简单对象访问协议(SOAP)定义使用XML和HTTP的远程过程调用。UPnP使用SOAP来向设备提供控制消息，并将结果或错误返回控制点。

为了向设备的服务发出一个动作，控制点必须采用以下格式的POST方法发送一个请求。以下是TV控制点向TV设备发送的SOAP控制消息（以PowerOn为例）：

```
POST /upnp/control/tvcontrol1 HTTP/1.1
HOST: 172.28.17.235:49152
CONTENT-LENGTH: 238
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-org:service:tvcontrol:1#PowerOn"
USER-AGENT: 5.1.2600 2/Service Pack 2, UPnP/1.0, Portable SDK for UPnP devices/1.6.6

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/">
<s:Body><u:PowerOn xmlns:u="urn:schemas-upnp-org:service:tvcontrol:1"></u:PowerOn>
</s:Body>
</s:Envelope>
```

```
HTTP/1.1 200 OK
CONTENT-LENGTH: 268
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: Mon, 17 May 2010 06:16:33 GMT
EXT:
SERVER: Linux/2.6.9-78.ELsmp, UPnP/1.0, Portable SDK for UPnP devices/1.6.6
X-User-Agent: redsonic

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"><s:Body>
<u:PowerOnResponse xmlns:u="urn:schemas-upnp-org:service:tvcontrol:1">
<Power>1</Power>
</u:PowerOnResponse>
</s:Body> </s:Envelope>
```

图23 请求数据包和响应数据包

前一部分是控制点发送的控制请求消息，采用 POST 方法，消息体为 XML 文件形式。在标头部分，HOST 指出了控制此服务的域名或 IP 地址、以及 URL 可选端口组件（设备描述服务元素的 controlURL 子元素）；SOAPACTION 是 SOAP 协议规定使用的标头，由调用的服务类型、散列符号和动作名称组成，均用双引号括起来，PowerOn 为要发送的动作名称。在消息体部分，Envelope 的“http://schemas.xmlsoap.org/soap/envelope”用来表示 SOAP 使用了 HTTP 扩展框架；Body 指出了具体的控制动作（PowerOn）。

后一部分是 TV 设备回送的响应消息，同样消息体以 XML 文件形式回复，回复消息表明了 Power 在动作执行后的当前值。在标头部分，DATE 指出了响应生成的具体时间；在消息体的 BODY 部分，设备返回了被操作的服务状态变量（Power）在执行动作(PowerOn)后的当前值。

● 事件通知的消息分析

在控制点发送的动作消息，设备执行了之后，除了发送相应的响应消息，设备还会采用GENA协议发送一个事件通知消息给控制点，该消息会经由miniserver模块处理，然后由回调函数TvCtrlPointCallbackEventHandler()更新相应的状态变量表。

```
NOTIFY / HTTP/1.1
HOST: 172.28.17.91:49152
CONTENT-TYPE: text/xml
CONTENT-LENGTH: 120
NT: upnp:event
NTS: upnp:propchange
SID: uuid:7f46bcb8-6317-11df-89e5-f1ca2c687b51
SEQ: 1

<e:propertyset xmlns:e="urn:schemas-upnp-org:event-1-0">
<e:property>
<Power>1</Power>
</e:property>
</e:propertyset>
```

图24 动作执行后的通知消息数据包

在数据包的消息体部分，返回了Power服务变量的当前值为1，即PowerOn的状态。

6.4.2 查询变量的消息分析

除了向设备的服务发出动作，控制点还可以对服务进行轮询，以通过发送查询消息获得状态变量值。查询消息只能查询一个状态变量，必须发送多个查询消息以查询多个状态变量。 此查询消息与该服务的事件（如果有）相分离。以下是TV例子所发送的查询消息和响应消息：

```
POST /upnp/control/tvcontrol1 HTTP/1.1
HOST: 172.28.17.235:49152
CONTENT-LENGTH: 284
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-upnp-org:control-1-0#QueryStateVariable"USER-AGENT: 5.1.2600 2/Service Pack 2,
UPnP/1.0, Portable SDK for UPnP devices/1.6.6


<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/
soap/encoding/">
<s:Body>
<u:QueryStateVariable xmlns:u="urn:schemas-upnp-org:control-1-0">
<u:varName>Power</u:varName>
</u:QueryStateVariable>
</s:Body>
</s:Envelope>
```

```
HTTP/1.1 200 OK
CONTENT-LENGTH: 283
CONTENT-TYPE: text/xml; charset="utf-8"
DATE: Mon, 17 May 2010 06:40:51 GMT
EXT:
SERVER: Linux/2.6.9-78.ELsmp, UPnP/1.0, Portable SDK for UPnP devices/1.6.6
X-User-Agent: redsonic

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/" s:encodingStyle="http://schemas.xmlsoap.org/
soap/encoding/">
<s:Body>
<u:QueryStateVariableResponse xmlns:u="urn:schemas-upnp-org:control-1-0">
<return>1</return>
</u:QueryStateVariableResponse>
</s:Body>
</s:Envelope>
```

图25 查询数据包

与前面的动作调用一样，前一部分是发送的查询请求消息，在标头部分，SOAPACTION中的QueryStateVariable表明是查询信息，这与动作调用区分开来；在消息体的BODY部分，指明了查询的服务状态变量（Power）及其所在的服务（control）。

后一部分是TV设备返回的响应消息，在消息体的BODY部分，返回了服务状态变量Power的当前值为1。

结论

作为家庭网络的核心技术之一，UPnP充分重用互联网技术标准，提供服务的跨平台自动发现和远程控制。基于UPnP技术设计应用程序将非常简单，目前许多国家在制订家庭网络体系结构时都采用了该项技术，因此UPnP有良好的应用前景，有必要积极研究开发基于UPnP的设备和应用。

论文系统地研究了UPnP及其相关协议，并结合TV控制点和设备的模拟实现，从规范描述和设备开发两个方面深入研究UPnP的实现技术。论文最后对捕获的数据包进行了详尽分析，为开发过程中的相关信息查询提供了有力参考，且为UPnP设备开发提供了事例借鉴。

当然，UPnP技术的普及任重道远，从技术层面来讲，尚有许多工作需要做。例如，需要对更多的信息设备制定UPnP标准规范；需根据规范描述开发更多的UPnP设备产品；还有必要进一步研究UPnP技术与OSGI、Jini等技术之间的联系，以实现真正的家庭联网