Linux 和 Windows 系統上的家庭網路 API

洪家鋒 尤云千 林盈達
交通大學資訊工程學系
E-MAIL：{cfhung, ycyou, ydlin}@cs.nctu.edu.tw
October 20, 2008

摘要

為了隨插即用、整合家中服務資源，家庭網路的自動化服務管理是最重要的課題。在目前的家庭網路環境中，藍牙(Bluetooth)和 IP-based 網路上的 Jini 及 UPnP(Universal Plug and Play)是用來實現自動化服務管理最重要的三個協定。Linux 及 Windows 系統上可以找到許多可供家庭網路程式開發者可使用的 API(Application Programming Interface)。

在藍牙的部份，我們介紹分別內建於 Linux 及 Windows 系統上的 BlueZ stack 及 Microsoft Bluetooth stack，再加上開放源碼(open source)且有豐富參考文件的 Affix stack。UPnP 則介紹 Window 官方的 UPnP API，以及可跨平台編譯的 C 語言 Portable UPnP SDK(Software Development Kit)，和使用上最方便的 CyberLink for Java。Jini 則是 Jini Starter Kit。開發藍牙程式時，直接使用作業系統內建的藍牙 stack 和現有的藍芽規範 API 是最快速的方法。而在 IP-based 環境下，選擇對家庭網支援度較高的 UPnP 則是較佳的選擇。

關鍵字：Home Networking, Bluetooth, Jini, UPnP, API

一、家庭網路

結束一整天辛苦的工作，才剛進家門，身心俱疲的你就直接從口袋拿出 PDA(Personal Digital Assistant)點一首節奏輕快的歌在二樓臥室的音響播放。上樓倒在床上稍微休息一下之後，突然想到樓下桌上型電腦上的最新影片已經下載好了，於是你就迫不及待的打開電視開始觀看這部你期待已久的大作 — 這麼方便的一切是怎麼做到的呢？讓我們先從家庭網路的架構看起。

家庭網路的架構

一般常見的家庭網路[1](圖 1)對外透過撥接或寬頻連接至網際網路，內部以有線的乙太網路(Ethernet)做為較快速的主幹，再搭配上無線區域網路(WLAN)[2]。在同一個房間內，許許多多的藍牙[3]裝置則構成短距離的無線藍牙網路。這些媒介連接著家中的個人電腦、筆記型電腦、家電及消費性電子產品，構成了這個遍佈著服務節點的網路環境。例子中二樓的電視機和音響透過 UPnP 除了可以得知樓下桌上型電腦裡面有哪些影片和音樂之外，更可以不必將影片或

音樂燒錄成 DVD 或 CD 之後再帶到樓上播放，而是直接建立影音串流來播放影片及音樂。

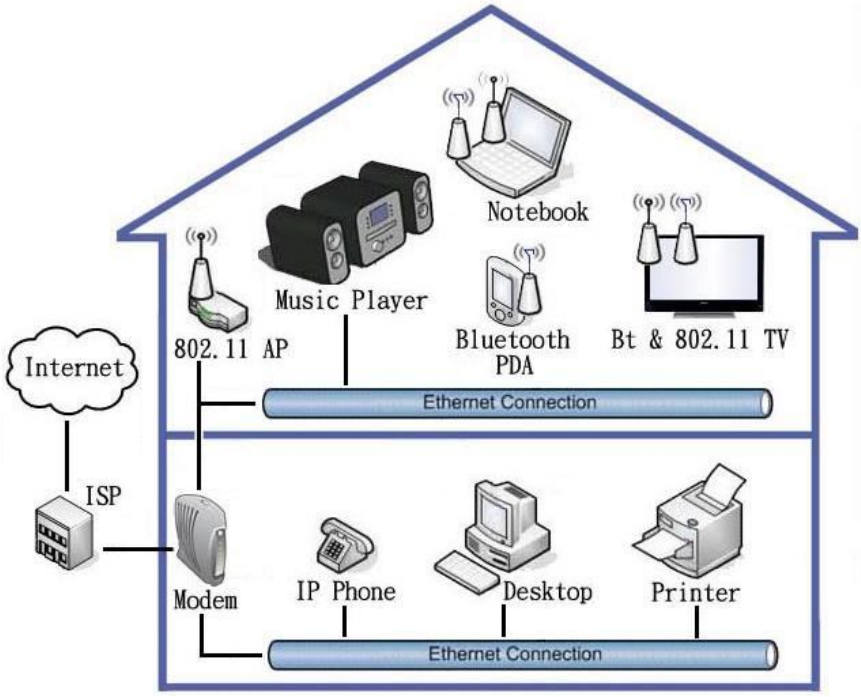

圖 1 家庭網路的環境

為了能整合家中的服務資源，且讓裝置隨插即用，家庭網路上必需要有自動化服務管理的通訊協定，來達成以下兩個重要的需求：

自動化的服務發現機制：當裝置加入或離開家庭網路時，事先只需最小程度、甚至不需要人為設定的情況之下，就知道目前家中有哪些服務可供使用，以及服務的位置。上述 PDA 的使用就是一個很好的例子。

標準化的服務使用規範：電視和音響因為所提供的影音服務具有標準化的介面，所以能和樓下遵循同樣標準的桌上型電腦互相溝通而沒有相容性的問題，服務資源能夠被整合起來。

自動化服務管理的通訊協定

在家庭網路的環境裡最常被使用的自動化服務管理的通訊協定是藍牙、UPnP[4]和 Jini[5]。藍牙在服務發現及使用上都提供了完善的機制，但因為是專門為短距無線傳輸所設計，使得藍牙通訊協定運作的範圍只侷限在藍牙網路上。

UPnP 則提供了最多元的服務使用介面，除了開發控制點(Control Point)程式進行服務的操作之外，開發者還可以提供控制網頁(Presentation Web Page)，讓使用者能透過網頁的介面來使用服務。

而 Jini 是三者中最特殊的一個。除了開發語言侷限於 Java 代表著需要更快速的運行平台之外，也需要集中式的服務目錄。而目前沒有標準的服務使用規範，這對裝置間的相容性是個大問題。不過 Jini 使用 Java RMI[6] (Remote Method Invocation)技術，讓開發者在使用遠端服務物件時可以如同使用自己的程式物件一樣的方便。表 1 列出了三者的特色及比較：

表 1 藍牙、UPnP 和 Jini 的比較

	藍牙	UPnP	Jini
開發者	Bluetooth SIG	Microsoft	Sun Microsystems
網路環境	Bluetooth	IP-based	IP-based
服務發現機制	Bluetooth SDP	SSDP	Lookup Service
集中式服務目錄	無	無	需要
標準的服務使用規範	Bluetooth Profile	Standard DCP Presentation web page	無
開發可用的程式語言	Independent	Independent	Java

從對家庭網路的支援度而言，藍牙及 UPnP 毫無疑問的是開發者目前的最佳選擇。但是 Jini 基於語言所帶來的開發便利性，如果能制定出標準的服務使用規範的話，未來仍是開發家庭網路程式的最佳選擇之一。

二、以 API 進行程式開發

本章節會著重於實務面，整理出在開發藍牙、Jini 以及 UPnP 的程式時所要注意的重點，並且介紹重要的相關 API。

藍牙程式開發

使用 socket 介面開發程式對於程式設計師是再熟悉不過的東西，所以開發藍牙程式時 socket 介面當然是我們的首選。藍牙 socket 依連線特性不同可分為類似 TCP 的 RFCOMM(Radio Frequency Communication) 與類似 UDP 的 L2CAP(Logical Link Control and Adaptation Protocol)。RFCOMM 提供較為可靠的連線性質，而使用 L2CAP 則可以降低封包的 Overhead。

開發者可以將應用程式直接建立在 RFCOMM 或 L2CAP 上，或是遵循藍牙規範(Bluetooth Profile)開發出放諸四海皆準的應用程式。有些藍牙 API 還支援部分的藍牙規範，所以選用這些 API 可以大大的減輕開發者的負擔。

當服務程式開發完後，接下來就是要考慮到如何讓使用者知道這項服務，且

知道去哪裡使用它。藍芽裝置上的發現伺服器(SDP server)就是服務的代理人，它負責回應使用者的服務查詢。要向發現伺服器註冊服務首先我們必須準備服務的相關資訊(如：socket 類型和 port 號碼)。這類的函式我們稱為 SDP Server API。

新的使用者剛進入房間打開藍牙 PDA 時，他並不知道這裡有哪些藍牙裝置，更別說是藍牙服務了。藉著搜尋裝置的功能使用者就可以找到鄰近的藍牙裝置。針對收尋裝置功能，藍牙 API 也提供一些函式，在這裡我們稱為 Inquiry API。

找到裝置的位置之後，就可以連上個別裝置的發現伺服器來進行服務的查詢，這些用來查詢服務的 API 我們稱之為 SDP Client API。根據查詢到的資訊我們就可以選定所需的服務，並且使用該服務。

Windows 與 Linux 兩大作業系統上各別內建有 Windows Bluetooth Stack 與 BlueZ stack。使用系統內建的開發環境可以減低設計師的負擔。Windows 的開發文件與相關資源都可以 Windows MSDN 中找到而 BlueZ 的開發者就沒這麼幸運了。Bluez 的官方網站中沒有提供相關的開發文件。幸運的是在 Linux 底下開發者還有另一個選擇就是 Affix。因為具備完善的開發文件，所以 Affix 非常受到開發者的歡迎。現在軟體工程師首選的語言當然是 C 和 C++。以上介紹的藍牙 API 都可以使用 C 和 C++來進行開發。除此之外 Affix 與 BlueZ 還支援較高階的 Python。Windows 中比較可惜的是只能使用 RFCOMM 而不能選用比較底層的 L2CAP socket。表 2 是筆者實際使用後的比較與整理。

表 2 藍牙 API 的比較

	Windows Bluetooth API	**Affix**	**BlueZ**
運行平台	Windows	Linux	Linux
開發可用的程式語言	C/C++	C/C++ Python	C/C++ Python
藍牙堆疊	Windows built-in	Affix-kernel package	Linux built-in
函式庫	Platform SDK	Affix package	bluez-libs package
使用許可	Microsoft	GNU GPL	GNU GPL
開發者	Microsoft	Nokia	Qualcomm
官方文件	MSDN	sourceforge	無
Socket 介面	RFCOMM	L2CAP RFCOMM	L2CAP RFCOMM
Address Family	AF_BTH	PF_AFFIX	AF_BLUETOOTH
其他介面	SDP, HCI	OBEX, PAN, HCI, SDP, HID	OBEX, HCI, SDP

Windows Bluetooth API：程式開發者在安裝 Platform SDK 之後便可以開發藍牙程式。Windows 內建的藍牙 stack 雖然支援了不少的藍牙規範，但只提供少數的藍牙規範 API，對程式開發者來說，使用起來較為麻煩。

BlueZ：在 Linux kernel 2.6 與 2.4 較新版本中內建了 BlueZ stack[8]與相關驅動程式，使用者只需安裝開發函式庫就能進行程式開發。目前在 Linux 上已經有非常多藍牙應用程式及函式庫是以 BlueZ stack 為基礎來開發。開發者可參考[9]。

Affix：擁有較豐富的藍牙規範 API 是 Affix 仍然相當受到歡迎的另一個原因。因為 Affix 不是內建在作業系統中，要使用 Affix stack 就需將其編譯進 kernel 或者編譯成模組(module)。筆者在建立 Affix 開發環境時發現，使用新版的 Linux kernel 與較新版的編譯器編譯 Affix kernel 3.2.0 會發生錯誤。其原因是新版 kernel 已不支援一些變數且新版的 gcc 編譯規則與舊版的有所出入。因此筆者使用較舊的 Linux kernel 2.6.9 與 gcc 3.3 才成功建立起環境。

Jini 程式開發

Java 中物件導向，使得程式開發者可以輕鬆透過類別(Class)的介面使用物件所提供的服務。這種透過物件來使用服務的概念可以在家庭網路中被展開來。一般的 Java 程式通常的是本地端的物件，但是提供服務的物件往往存在遠端伺服器中，所以必須透過 RMI 的機制實行遠端的呼叫。

RMI 程式模型(圖 2)目標是要讓 Client 端可以透過服務代理物件(Service Proxy)直接使用遠端物件所提供的服務。為了達成目標，首先我們必須思考如何讓 Client 擁有服務代理物件。Jini 使用的機制非常簡單，查詢伺服器(lookup service)就是其中的關鍵。每個 Jini 網路至少都會存在一個查詢伺服器，每個服務的提供者必須先找到它的位置，且向它註冊服務代理物件。當 Client 端連上網路時也會試著尋找查詢伺服器，並向它查詢所需的服務。但是只拿到了服務代理物件的 Client 端還是無法執行遠端呼叫，因為 Client 端程式編譯時用到的只是服務的介面(Interface)，而不是實做過後的服務類別，實作後的類別需以其它的方式來提供。所以在 Jini 網路上往往都會存在 HTTP 伺服器，用以提供 Client 端下載服務代理物件實作後的類別。這種遠端下載類別的方式看似笨拙，其實是非常聰明有彈性的作法。因為實作後的類別沒有一開始就編譯進 Client 中，所以開發者可以在不更動介面的情況下任意的修改服務的實作方式。根據以上的敘述我們可以歸納出 Client 端與 Server 端的程式流程，其中每一個步驟都 SDK 都有提供相關的 API 與重要的物件。

Server 端需要先根據已經訂好的服務介面實作出服務的類別，這個類別必須

放至 HTTP 伺服器上以供需要的 Client 下載。Server 程式執行流程，首先當然創造出服務的物件接著就是找到查詢伺服器，並向它註冊服務代理物件。找尋的方式可分為單播(Unicast)與群播(Multicast)兩種。若你已經知道查詢伺服器的位置，可以使用單播的方式連上伺服器，否則只好使用群播找尋周圍的查詢伺服器。接著 server 端就可以開始等待 clinet 的遠端呼叫。

Client 端首先使用與 server 端相同的服務介面開發程式。接著 client 端程式也需尋找查詢伺服器。連上伺服器後 client 端利用服務介面為依據找尋符合這個介面的服務代理物件。獲得物件後，client 端必須從 HTTP 伺服器上下載屬於這個服務代理物件的類別。只有在獲得服務代理物件與其類別後我們才可以直接使用遠端的服務。

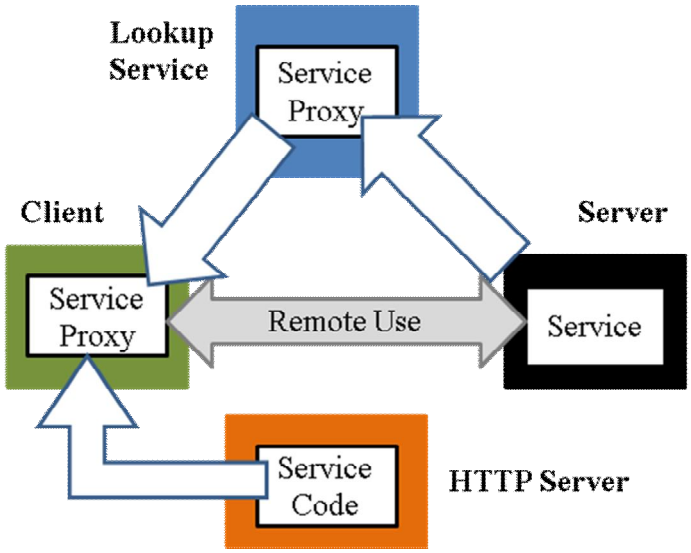

圖 2 Jini 遠端控制模型

Jini starter kit: Jini start kit 是昇陽(Sun)公司提供的 SDK，提供了 Launch-All 綜合應用程式，其中包含了 lookup service、http server 與 service browser，所以開發者不需費心另外準備環境。編譯 Jini 程式時為了簡化重複設定參數與編譯指令，可以使用自動化編譯軟體 Ant。Ant 的功能相當於是 Linux 底下的 make，它可以讀取 XML 檔案內容編譯程式。

UPnP 程式開發

UPnP 程式依功能考量可以分為兩種。一種是控制點，另一種則為提供服務的服務裝置(Service Device)。實際上的應用，一個裝置有可能同時扮演這兩個角色。

UPnP 程式一開始都需使用函式來註冊新的裝置，這可以讓你的應用程式與 UPnP SDK 建立溝通管道。溝通管道依據不同的程式語言而有不同的變化。Java

API 使用的是實作介面的方式處理底層到應用程式的事件，C 語言則是提供 SDK 一個回調(Callback)函式指標。建立起訊息傳遞的管道後，程式設計者主要的任務就是處理即將從 SDK 傳來的事件。服務裝置要處理的事件是執行使用者的請求等，而控制點則是必須注意的是網路中新裝置出現等事件。

UPnP 使用 XML 來描述裝置資訊與定義服務內容，所以當我們構思好裝置的特性與服務的內容後，必須將這些資訊轉化為 XML 檔案。要註冊服務裝置時必須將裝置描述檔案與服務描述檔案提供給 SDK，讓 SDK 散播裝置的資訊。

控制點的功能在於提供控制介面給使用者。因為控制點不須應付其他裝置的需求，所以開發程序較為單純。其程式功能重點在於掌握網路上服務裝置的資訊。這些資訊可能是來自裝置的廣播或控制點主動的搜尋。開發控制點程式時必須提供使用者瀏覽裝置、訂閱服務變數和請求動作的功能，而這些功能都可以藉由 API 來完成。

表 3 UPnP API 的比較

	Windows UPnP API	Portable UPnP SDK	CyberLink for Java
使用許可	Microsoft	BSD License	BSD License
開發者	Microsoft	Intel	Satoshi Konno
官方文件	MSDN	sourceforge	sourceforge
開發可用的程式語言	任何支援 COM 的語言	C	Java
運行平台	Windows	大多數	Java
狀態表產生方法	在原始碼編寫	在原始碼編寫	從 XML 檔讀取
XML 解析器	Built-in	Ixml	Apache Xerces kXML2

Windows UPnP API：Windows UPnP API[11] 的特色在於它是 COM(Component Object Model)介面。可以用任何能夠支援 COM 的程式語言來開發，其中裡面的控制點 API 更支援 VBScript，在實作控制網頁時非常方便。

Portable UPnP SDK (libupnp 1.6.6)：Portable UPnP SDK[12]源自於 Intel 的一套 SDK 原名為 libupnp，在 sourceforge 網站中也是下載次數最多的 UPnP SDK。早期的版本只能運行在 Linux 相容的系統，新版本 1.6.6 則是可以運行在大部分的作業系統之上。

CyberLink for Java：緊接著介紹一套 Java 的 SDK。CyberLink for Java[13]使用物件導向的繼承(Inherit)與實作介面(implements)使得程式架構非常清晰。官方提供的開發文件中除了詳細的記載函式用法，還特別提供 UML(Unified Modeling Language) Diagram 開發起來更為得心應手。

三、程式實例

介紹完 API 接下來提供三個開放源碼的應用程式，分別使用上述的 API。以下就分述這些程式碼並指出 API 在其中扮演的地位與功能。

Bluetooth – ussp-push

ussp-push[14]可以透過藍牙介面在裝置間傳輸物件，程式中使用 OpenOBEX 函式庫幫助開發。OpenOBEX 提供物件交換的協定，而 BlueZ 則是負責網路底層的傳輸。

```
if ( (niinf = hci_inquiry(devid, 32, -1, NULL, &piinf, 0)) < 0 ) {

if ( hci_remote_name(    dd, &piinf[i].bdaddr, sizeof(devname) - 1,
                         devname, 100000) >= 0 ) {
```

obex_socket.c 程式碼中使用到兩個重要的 Inquiry API。首先使用 hci_inquiry 搜尋附近藍芽裝置。接著將搜尋的結果當成 hci_remote_name 的輸入參數，用來獲得遠端藍牙裝置的名稱。

找到鄰近的裝置後，我們就可以使用 SDP Client API 來查詢遠端裝置所提供的服務。

```
sess = sdp_connect(iface, bdaddr, SDP_RETRY_IF_BUSY);
```

obex_sdp.c 中 sdp_connect 扮演的角色是建立起與遠端裝置發現伺服器的連線，藉著這條連線我們可以搜尋發現伺服器上已經註冊的服務。

```
if ( sdp_service_search_attr_req(   sess, search,
                                    SDP_ATTR_REQ_RANGE,
                                    attrid, &seq) ) {
```

ussp-push 是 OBEX 的使用者端程式，所以搜尋目標當然是 OBEX 伺服器。

當我們使用 sdp_service_search_attr_req 進行搜尋時，必須指定 OBEX 伺服器的 UUID(Universally Unique IDentifier)當作找尋的依據。程式碼中的輸入變數 search 就是用來指定目標服務的 UUID。

```
if ( (sock = socket(  AF_BLUETOOTH, SOCK_STREAM,
                      BTPROTO_RFCOMM) ) < 0) {
```

當找到 OBEX 伺服器之後，我們就可以使用 socket API 連上遠端裝置。obex_socket.c 中使用 socket API 來建立連線。因為檔案傳輸需要高度的可靠性，所以程式使用 RFCOMM socket 而不是 L2CAP。

UPnP – MediaTomb

MediaTomb[15]是架構在 libupnp 上的多媒體伺服器，它實做 UPnP MediaServer V 1.0 提供線上瀏覽與播放多媒體的功能。原始檔 server.cc 中包含了許多關於 UPnP 的程式碼，其中最重要的包括下列幾行程式。

```
ret = UpnpInit(ip.c_str(), port,
          config->getIntOption(CFG_SERVER_RETRIE_ON_TIMEOUT), cb);
ret = UpnpRegisterRootDevice2(  UPNPREG_BUF_DESC,
                                device_description.c_str(),
                                device_description.length() + 1,
                                true,
                                static_upnp_callback,
                                &device_handle,
                                &device_handle);
```

程式一開始都需要呼叫 UpnpInit 初始化 UPnP SDK。因為 MediaTomb 是 UPnP 的服務裝置，所以使用 UpnpRegisterRootDevice2 函式向 SDK 註冊新的裝置。函式輸入的參數 static_upnp_callback 是一個函式的指標。static_upnp_callback 回調函接收從 SDK 傳來的事件，並且將訊息分送到適當的處理流程。想了解事件處理的流程可以從這個回調函式開始追蹤(tracing)。

Jini – Home-automation project

在 Jini 部份我們介紹 Home-automation 的 music[16]服務。Music 利用 Jini 技術，讓你可以在本地端以樹狀目錄的方式瀏覽，並在遠端播放 mp3 音樂。

```
impl = new JukeboxImpl(musicDirectory);
```

首先在 MusicService.java 程式中建立了一個 JukeboxImpl 的物件，這個物件就是服務所要使用到的介面。musicDirectory 則是音樂目錄的根目錄。

```
proxy = exporter.export(impl);
```

接著把建立好的 JukeboxImpl 物件輸入到 exporter.export()函式，產生一個能夠在遠端用來呼叫 JukeboxImpl 物件函式的 proxy 物件。Proxy 物件的產生程序就到這裡為止。

```
LookupDiscoveryManager mgr =new LookupDiscoveryManager(
                                    LookupDiscovery.ALL_GROUPS,
                                    null, null);
```

接下來，建立 LookupDiscoveryManager 物件，目的是用來尋找 lookup service，才能把 proxy 物件及服務的相關訊息註冊上去。LookupDiscovery.ALL_GROUPS 代表所有 group 的 lookup service 都在尋找的範圍內。

```
joinManager = new JoinManager( proxy, entries, storage, mgr,
                                    new LeaseRenewalManager());
```

最後，利用 joinManager 來把服務的資訊和 proxy 註冊到 lookup service 裡。如此一來，music 服務已經能夠在 Jini 網路上被發現和使用了。

四、結論

對於家庭網路程式的開發者而言，首先要決定傳播服務所要使用的網路環境和通訊協定。若是在藍牙環境底下的話，建議使用平台上內建的藍牙 stack，並且使用現有的藍牙規範 API 來加速開發。

在 IP-based 的網路環境下，如果是以對家庭網路環境的支援程度而言，Jini 還不算是個非常成熟的技術，建議以 UPnP 來做為自動化服務管理的通訊協定。開發者可以依據標準 DCP 來開發與現有 UPnP 產品相容的程式。以開發過程的方便性和程式移植性來說，建議使用 CyberLink for Java，其次是 Portable UPnP SDK，最後才是 Windows UPnP API。

五、參考資料

[1] S. Teger and D. J. Waks, "End-user perspectives on home networking," , *IEEE,Communications Magazine* vol. 40, pp. 114-119, 2002.
[2] K. Vaxevanakis, T. Zahariadis, and N. Vogiatzis, "A review on wireless home network technologies," ACM SIGMOBILE Mobile Computing and Communications Review, vol. 7, pp. 59-68, April 2003.
[3] Bluetooth.com, http://www.bluetooth.com/bluetooth/
[4] UPnP™ Forum, http://www.upnp.org/
[5] Jini.org, http://www.jini.org/wiki/Main_Page
[6] Remote Method Invocation Home, http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp
[7] Bluetooth (Windows), http://msdn.microsoft.com/en-us/library/aa362932(VS.85).aspx
[8] BlueZ, http://www.bluez.org/
[9] A. Huang and L. Rudolph, "Bluetooth for Programmers."
[10] Affix Bluetooth Protocol Stack For Linux, http://affix.sourceforge.net/
[11] UPnP APIs(Windows), http://msdn.microsoft.com/en-us/library/aa382303(VS.85).aspx
[12] Portable SDK for UPnP Devices (libupnp 1.6.6), http://pupnp.sourceforge.net/
[13] CyberLink for Java, http://www.cybergarage.org/net/upnp/java/index.html
[14] ussp-push Home Page, http://www.xmailserver.org/ussp-push.html
[15] MediaTomb, http://mediatomb.cc/
[16] home-automation, https://home-automation.dev.java.net/docs/Music.html