異質網路服務整合架構

An Integrated Service Architecture for Heterogeneous Service Network

林慶達　魏溥辰　錢俊舟
Chin-Ta Lin, Pu-Chen Wey, Chun-Chou Chien

中文摘要

隨著網際網路的普及與家用電器的成長，對於整個生活形態的演進有相當大的影響，加上行動通訊、無線網路等下層支援，讓原先家中的電話、資訊設備都漸漸的產生改變，並且越來越聰明，具備一定程度的智慧性。本文提出一整合異質網路之系統與架構，可以讓不同服務網路之間的服務進行交互運用，以達到數位家庭服務的場景目的。

Abstract

As the internet growing popular, there are more home appliances with IT technology appearing on the manufacturing schedule. People who live with the new equipment will get a lot of benefits from the new scenario and live in a new lifestyle. Since this system will need mobile computing and a wireless network to make home phones and information appliances adapt to the new style, and to give the devices intelligence to help people handle more events, we propose a new Integrated Service Architecture to integrate heterogeneous services into the UPnP network.

關鍵詞(Key Words)

異質網路 (Heterogeneous Network)
網路服務 (Network Service)
服務協定 (Service Protocol)

1．前言

隨著網際網路的普及與家用電器的成長，對於整個生活形態的演進有相當大的影響，加上行動通訊、無線網路等下層支援，讓原先家中的電話、資訊設備都漸漸的產生改變，並且越來越聰明，具備一定程度的智慧性。

這樣的趨勢，讓原先需要人力介入的單一事件，變成一按即可的簡單生活形態，不必再擔心飯煮不好，水燒乾了，對於一般生活的方便性可以有加成的效果；另外在X10網路的控制設備上，也提供了隨手可得的自動化的服務網路，與現在潮流DLNA所訂定的規格相較，因為並非開放規格，所以投入之廠商並不多，而DLNA產品在去年九月認證方案通過之後，將會快速成長。

在本文中將以UPnP為兩個異質性網路的界接協定架構，並提出異質網路轉接與服務的重新發佈功能，並依此接取後端的條件事件引擎，依照所設定的感知數據，決定之後系統所

採取的步驟。

下文將依照系統概括介紹、異質網路橋接架構、條件事件引擎、遠端介面與最後的訊息框架進行說明。

2．概括介紹

從系統面觀看，我們可以將其分割為圖2-1：

中央的UPnP網路為整個系統交換的共通協定，所以為最大的網路區塊雲，而而左下角的部分為非IP設備的資訊交換網路，中間需要一個「Non-IP Appliance Controller」，用來將上面的服務進行重新發佈的工作，而右下角的部分除了可以透過Home Gateway連接到網際網路之外，尚可以連接一般UPnP網路上的服務，甚至是UPnP AV規格內的影音服務，也統整在規劃當中。

左上角為DLNA所規範的MEDIA HUB，用來將數位媒體傳送到特定裝置，依循UPnP的規範，可以設定從某一部機器將MEDIA STREM傳送到另一個位置，用來進行播放或是即時轉碼的過程。

在整個場景當中，尚包括了支援UPnP的各項設備，也都可以成為後面橋接與事件設定所需要環節中的一員，對於整個數位家庭環境的運作來說，這些都是非常重要的成員，因為在每個動作當中，都需要處其數個事件，每個事件當中也必須要有數個裝置的構成，這樣才有辦法完成使用場景的建立。

3．異質網路橋接架構

在異質網路間的服務傳遞，必須依靠轉接的過程，也就是整個服務的重新發佈，將非UPnP的網路服務進行前端接取，然後重新發佈

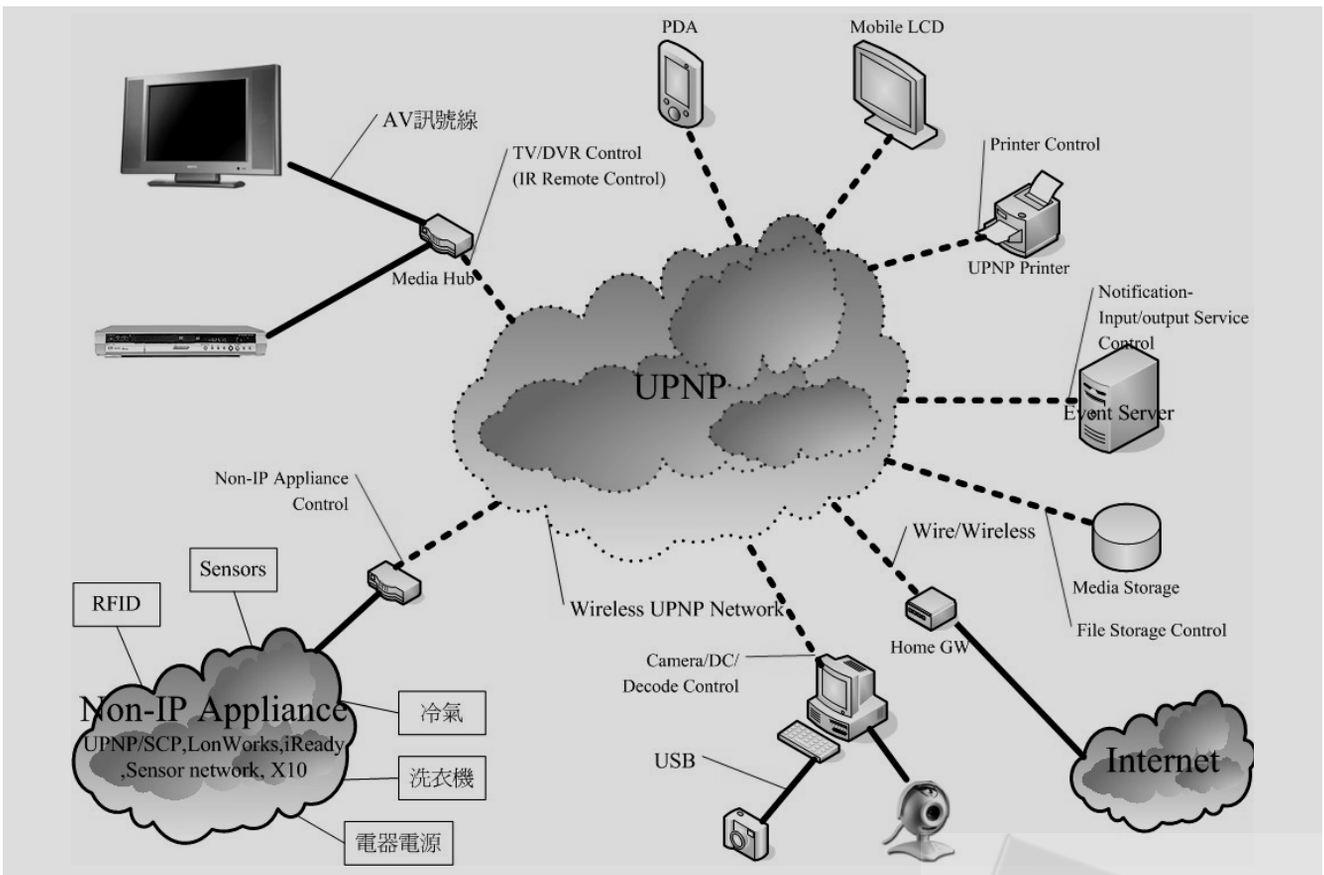

圖 2-1　系統整體架構圖

成為UPnP的標準服務，這樣的過程便是橋接的主要目的，透過橋接程序，可以讓像是X10這種現存的服務網路，將他所提供的功能，發佈到UPnP網路上，讓其他的裝置或是程式能夠透過標準的UPnP方式使用。

在Bridge的設計上，我們將其分為兩大塊，分別是左邊的掛載式異質服務接取方式，與右邊的UPnP管理模組。Non-IP Appliance Manager的機制由四個主要部分構成，分別是Service XML Resitory、UPnp Service Dispatcher、Service XML Creator、以及Service Manager；其中Service XML Reositiry是用來儲存服務的描述檔針對每個支援的描述以XML的形式記錄下來，以便接取或是轉發服務之用。UPnP Service Dispatcher則是用來轉送由UPnP Manager所發送過來的服務請求，當UPnP網路上有設備提出服務請求時，將服務送到下方的Plug-in，目前在系統的實做上是透過外掛的方式，將不同的異質服務網路掛載上來，如圖3-1所示，目前已經有X10與Phidgets兩種異質網路接取方式。

第三個元件是Service XML Creator，這是用來建立服務XML的元件，透過這個元件可以將下方的服務描述出來，讓Service Manager可以將其發佈到UPnP網路上，提供外界存取，同樣的，在搜尋到一般UPnP網路上的服務之後，也會透過這個產生器讓下方掛載上的異質裝置可以接取，服務。最後是Service Manager，這是用來發出服務事件的管理元件讓雙方的服務溝通可以透過這個特定的管理方式運作，並維持一定的運作彈性。

右邊的UPnP管理模組則是依照標準的UPnP架構運作，本文就不再多說。

4．遠端介面

除了事件之外，系統在介面上採用了UPnP標準的遠端介面方式進行設定與操控，下面將介紹整個遠端介面的運作方式。

系統內使用UPnP RemoteUI服務作為使用者介面的運作平台，RemoteUI服務分為RemoteUI Server以及RemoteUI Client兩種服務。RemoteUI Server能提供客戶端查詢目前該伺服器所提供的可用使用者介面資訊，

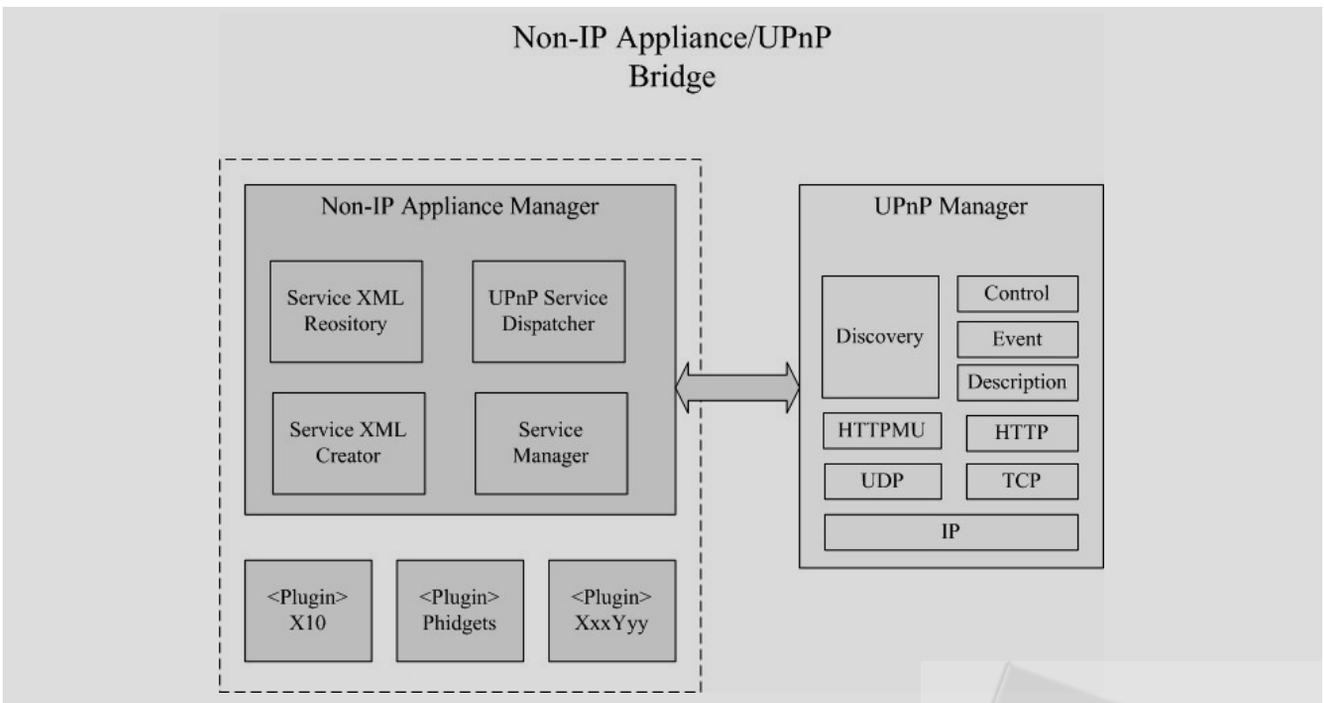

圖 3-1　異質網路橋接架構圖

RemoteUI Client則能提供伺服器主動通知客戶端可用的使用者介面資訊。

RemoteUI使用者介面資訊採用XML的方式呈現，並包含下列資訊：

名稱	說明	
uiID	代表此使用者介面的唯一識別碼	
Name	此使用者介面的名稱	
Description	此使用者介面的說明	
iconList	代表此使用者介面的圖示	
Fork	連線時是否建立新的使用者介面	
Lifetime	離線後清除使用者介面資料的時間	
Protocol	此使用者介面實地用以呈現的通訊協定，選擇如下：	
	HTTP/HTML	透過HTTP協定使用HTML呈現使用者介面
	RDP	Microsoft遠端桌面協定
	VNC	AT&T VNC協定
	XRT2	透過Intel XRT協定
	XHT	透過Samsung XHT協定
	SGXML	透過Siemens Gigaset XML協定
	UIF	透過Philips UI Fragments協定

表 4-1　RemoteUI 資訊表

使用者介面運行時，會經過下列步驟：

1. 使用者介面伺服器搜尋到網路具有支援RemoteUI的客戶端。
2. 透過RemoteUI Client ControlPoint連線至客戶端。
3. 取得該客戶端的裝置資訊。(項目如表4-2)

資訊名稱	說明
maxHoldUI	裝置最多能維持幾個介面的session
Protocol	裝置所支援的協定
protocolInfo	協定的參數

表 4-2　客戶端資訊裝置表

4. 透過裝置資訊中對所支援協定列出的參數，與伺服器上提供之介面做比對，選擇出適合該客戶端的使用者介面。本系統採用XRT2作為介面傳輸協定，表4-3中列出此協定的裝置參數及資訊。

裝置參數名稱	說明
DisplayWidth	客戶端顯示裝置的寬
DisplayHeight	客戶端顯示裝置的高
ImageEncoding	客戶端所支援的圖片類型(JPEG, PNG……)
MaxCommandSize	客戶端所允許的最大命令長度

表 4-3　裝置參數與資訊表

5. 透過RemoteUI Client ControlPoint，將選出的使用者介面以URL的方式通知該客戶端。
6. 使用者於客戶端上選擇其所需之使用者介面，RemoteUI Client Service會啟動該協定所使用的傳輸協定，連線至設定的URL位置。

本系統採用Intel XRT2作為使用者介面畫面及控制的傳輸的通訊協定，符合Intel NMPR的規範。XRT2通訊協定中，客戶端會將使用者的滑鼠及鍵盤動作傳送至伺服器端，而伺服器端則會將目前顯示的畫面，以圖片的格式傳送至客戶端顯示(如圖所示)。

本系統實作之XRT2伺服器支援透過遠端Web Server取得應用程式畫面，亦提供一套介面發展程式庫，提供介面開發者於不需要架設Web Server的情況下使用.NET平台之程式語言開發使用者介面，加速系統發展效率。此介面發展程式庫所支援的圖形介面元件如表4-4：

名稱	說明
Button	按鈕
ComboBox	清單選擇器
Image	圖片
Label	文字標籤
Media	多媒體播放器
Table	排版用表格元件
TextBox	文字輸入元件

表 4-4　XRT2介面資訊表

5 · 訊息框架

在事件發生的反應當中尚有一項重要的動

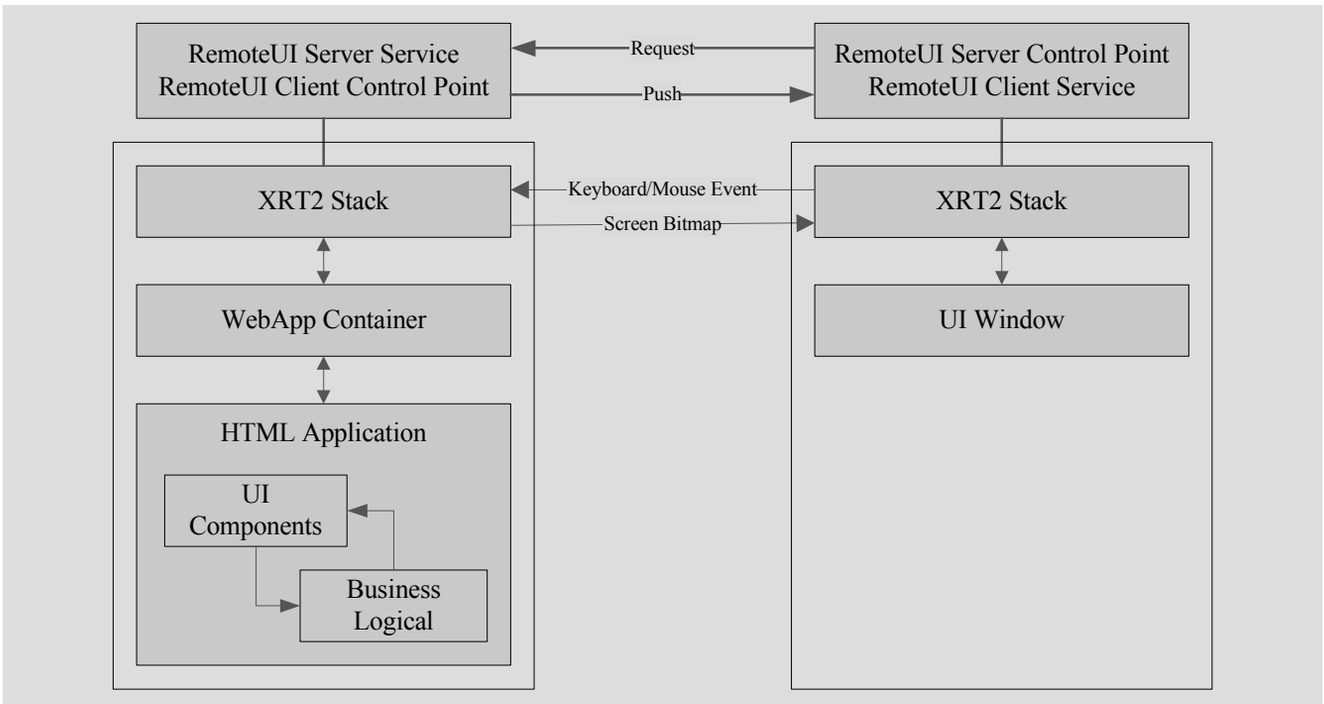

圖 5-1　訊息框架架構圖

作，就是訊息的傳遞，因次在系統中有專門處理訊息通知與傳遞的部分。

整個CCL Notification system分成兩個部分 Notification server 和 Notification Queue server，以下簡稱NFS server與NQS. server。

Notification system的設計需求大致有以下幾個：

1. 能夠設定不同的發送條件
2. 能夠根據不同媒介發送訊息
3. 能夠負荷大量的訊息發送
4. 能夠保證訊息發送的正確性，先後順序

NQS server負責接收Digital Home event engine傳送過來的訊息，並依序放入Queue中，訊息的內容是以XML定義，因此Notification server可以根據需要抓取Queue中的內容，再按照訊息內容透過SMS，SMTP，或者MSN等不同協定發送訊息，邏輯設計上NFS與NQS保持獨立設計，而系統間的溝通可以是一對一或多對多的關係，方便作cluster或者監控管理，比如NQS除了讓NFS讀取內容外，也可以讓管理系統連入並監控內部狀態，相同的，NFS在主要

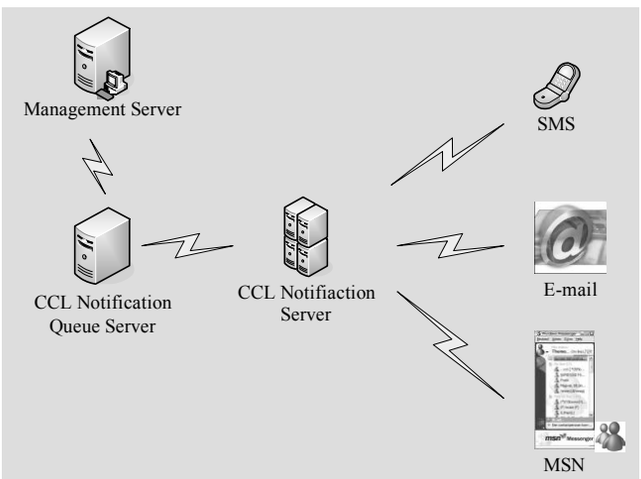

圖 5-2　系統元件關係圖

NQS斷線時，也可以存取備份的NQS，以保持訊息發送的正確。

首先介紹NQS系統

NQS設計考量到以下幾點：

1. 可以廣播訊息給任意多個客戶端
2. 可以改變尚在Queue中的訊息內容
3. 使用TCP/IP
4. 可以讓客戶端制訂運行在伺服器端的

圖5-3為CCL NQS系統設計模組包含了Queue，Message和NQS Server模組，客戶端透過TCP .NET Remoting Channel取得NQS Server Object使用此方法是因為比透過HTTP取得序列化SOAP物件的方法來得有效率。而Object Factory則是包含了Message和Queue物件的實作，分別繼承IMessage和IQueue介面，這種實作方式，是為了方便使用不同語言實作Message和Queue物件。

</message>

其中Message id是由GUID得來，目的是為了確保訊息的單一性，方便系統辨認，另外紀錄訊息進入Queue的時間，SERVICER_ID則是記錄訊息存是哪一台NQS server，而body中的內容才是NFS server真正處理的地方，將會在下面篇幅描述。

客戶端可以藉由實作QueueMessageFilter模組，使用者可以使用Message filter機制，這

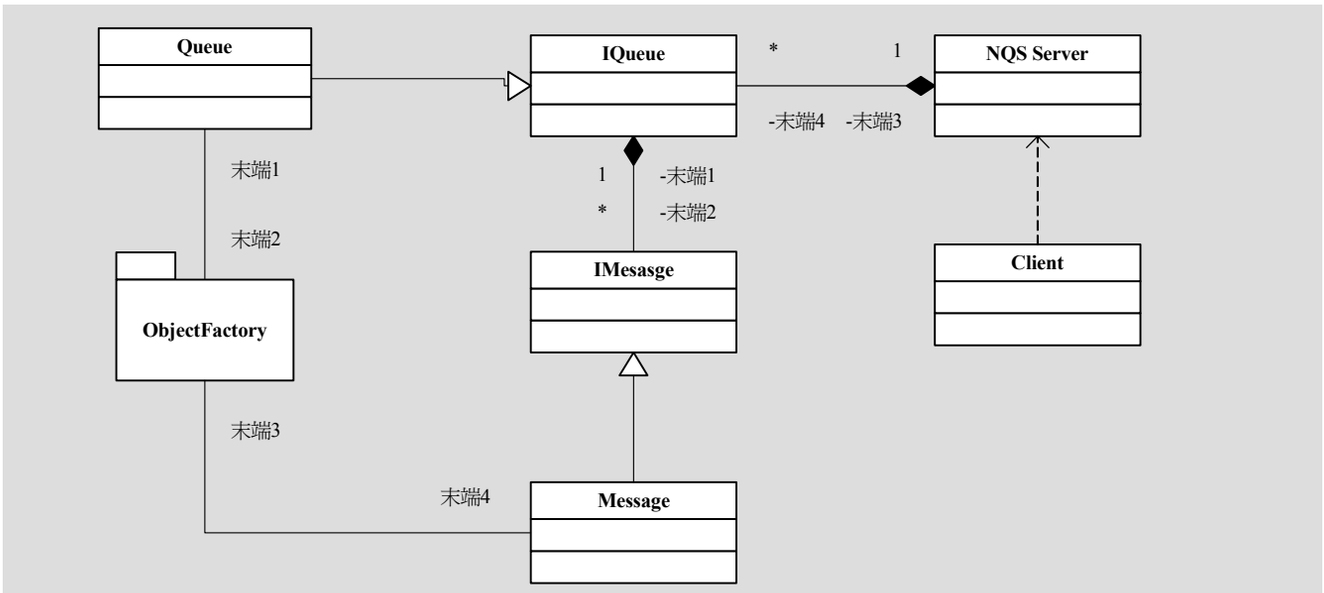

圖 5-3　訊息框架軟體分析圖

客戶端透過.Net delegate機制來取得Message物件，也就是Queue中所儲存的Notification 訊息，同時，為了防止訊息因為意外而消失，系統另外實作了儲存的機制，儲存的格式則是以XML製作，如下

```
<?xml version="1.0" encoding="utf-8"?>
<message
id="02b35cb4-ec66-4db4-9920-80a607f74d23">
    <BODY>xxxx</BODY>

<GUID>02b35cb4-ec66-4db4-9920-80a607f74d23
</GUID>
    <TIME_ENTERED_QUEUE>2005/12/16
05:27:35</TIME_ENTERED_QUEUE>
    <SERVICER_ID>xxx.xx.xx</SERVICER_ID>
```

個機制是為了方便管理Queue中訊息的狀況，使用者可以自己定義條件，當訊息被移除，加入，更新的時候，便會執行使用者所制訂的條件，比如說發出警告訊息。

以下針對NFS Server設計作簡單介紹，NFS包含幾個子模組NOTIKernels，NOTIHandlers，NOTIGlobals, NOTIExceptions, NOTIConfig, 其中NOTIKernels為系統的核心模組，它會根據呼叫NOTIConfig系統根據適當的設定檔案初始化NFS，並且在制訂的時間從NQS取得訊息的內容處理，其訊息的格式制訂如下

```
<CCLNOTIMSG>
<SENDTO>digital-home@msn.com</SENDTO>
<SENDTYPE>MSN 9</SENDTYPE>
<MIMETYPE></MIMETYPE>
```

<CONTENT>Door-11022 open</CONTENT>
<SCHEDULETime>22:03</SCHEDULETime>

<SCHEDULEINTERVAL>Everyday</SCHEDULEINTERVAL>
</CCLNOTIMSG>

內容包括傳送的目的地SENDTO，當然使用者可以是一個人會者是一整個群組，SENDTYPE表示使用者哪種傳送協定，CONTENT是描述操作哪一個device，另外還定義了執行的時間。

NOTIKernels模組會根據內容將此項作業排入批次作業中，並且根據SCHEDULETime和SCHEDULEINTERVAL執行發送的命令，發送命令時，會從系統中挑選出適當的Handler，例如此範例會透過MSN Handler來執行，MSN Handler會根據SENDTO送出訊息給使用者，另外MSNHandler設計成一個MSN Robot形式，也就是可以接收使用者執行命令，比如重新啟動系統或者更改下次通知時間等，其他對應不同的應用程式有SMSHandler，SMTPHandler等。

當然，訊息發送並不是一定能順利完成，並到Exception狀況時候，NOTIExceptions會執行相關的作業程序，儲存回NQS中，排定下次重試時間，或者改由其他協定傳送，端看使用者如何設定處理程序。

系統未來將會持續增加不同的發送應用程式，比如說MMS，Skype語音等，讓使用者有更多的選擇方式。

6 · 結論

數位家庭的趨勢目前以慢慢成形，其中DLNA組織的場景已經成為各家廠商產品規劃的目標之一，加上2005年九月的互通認證方式建立，也提高了產品之間的可用度。但僅有這些尚不足以成就整個家庭網路的便利與推展，本文所提之異質網路方式，將可以增加現存服務與裝置加入新的服務框架的可行性，並且提供未來進一步邁向無所不在的運算環境當中的架構環節。

計畫相關資訊

本文係工研院資通所執行經濟部FY94年度科專計畫「智慧型資訊系統發展計畫」子計畫「智慧型空間網路服務技術」成果之一。

參考文獻

[1] UPnP RemoteUI Server Specification, Service Template Version 1.01.
[2] UPnP RemoteUI Client Specification, Service Template Version 1.01.
[3] Intel Extended Device Remote Transfer Protocol (XRT Protocol) Specification Version 2.2.

作者簡介

林慶達

現任職於工研院資通所網際網路應用技術部工程師。畢業於成功大學工程科學研究所，專長為系統規劃分析、作業系統核心程式、軟體工程、及行動計算。
E-mail: dereklin@itri.org.tw

魏溥辰

現任職於工研院資通所網際網路應用技術部副工程師。畢業於清華大學資訊工程研究所。專長為軟體程式設計、網際網路系統技術。
E-mail: williamwey@itri.org.tw

錢俊舟

現任職於工研院資通所網際網路應用技術部副工程師。畢業於中央大學資訊工程所。專長為專長為軟體程式設計、網際網路系統技術。
E-mail: bounz@itri.org.tw